UNIVERSIDAD CATÓLICA DE LA SANTÍSIMA CONCEPCIÓN

FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

Procesamiento de imágenes en tiempo real mediante diseño en lógica programable para caracterizar el movimiento microscópico utilizando Speckle Dinámico.

Ricardo Germán Bahamonde Espinoza

Informe de Habilitación Profesional para optar al título de:
Ingeniero Civil Eléctrico

Profesor Patrocinante:
Dr. Jaime Cariñe Catrileo

Comisión:
Dr. Daniel Martínez
Dr. Ricardo Bustos

Concepción, octubre de 2023

UNIVERSIDAD CATÓLICA DE LA SANTISIMA CONCEPCION
Facultad de Ingeniería
Departamento de Ingeniería Eléctrica

Profesor Patrocinante:
Dr. Jaime Cariñe Catrileo

Procesamiento de imágenes en tiempo real mediante diseño en lógica programable para caracterizar el movimiento microscópico utilizando Speckle Dinámico.

Ricardo Germán Bahamonde Espinoza

Informe de Habilitación Profesional
para optar al Título de

Ingeniero Civil Eléctrico

Concepción, octubre de 2023

Resumen

En este trabajo se evalúa la factibilidad de aplicar algoritmos de procesamiento de imagen en tiempo real mediante diseño de lógica programable con la finalidad de caracterizar el movimiento microscópico de una muestra de pintura (corrector) utilizando un patrón de Speckle Dinámico. Para ello, se cuenta con un sistema óptico montado en el Laboratorio de Optoelectrónica de la Universidad Católica de la Santísima Concepción (UCSC) para la generación y estudio de patrones de Speckle dinámico. Las imágenes adquiridas por este sistema óptico son almacenadas para posteriormente ser procesadas mediante algoritmos en un computador. Se busca crear un sistema más compacto que permita realizar algoritmos de procesamiento de imagen en tiempo real y no en tiempo diferido, por ello esta etapa será modificada incorporando un Field-Programable Gates Array (FPGA).

Para desarrollar este sistema fue necesario implementar un hardware que permitiera realizar la conexión entre una cámara y un monitor por medio de protocolo “High-Definition Multimedia Interface” (HDMI) y entregar el valor de intensidad de un píxel como salida en un “Digital to Analogue Converter” (DAC) que enviará un voltaje analógico proporcional a la intensidad del píxel analizado hacia un osciloscopio. Parte del diseño permite al usuario observar en un monitor dos cuadrados (uno amarillo y otro morado) controlables mediante la botonería del FPGA que representan al píxel analizado. Para validar que los cuadrados representaban al píxel estudiado se realizó una comparación en el osciloscopio donde se variaba la intensidad de un LED. Esta prueba demostró que la intensidad del LED era proporcional a la intensidad del píxel en el osciloscopio, llegando a un umbral de 3.3 volts.

Se emplea dentro del desarrollo del código un algoritmo de diferencias generalizadas (GD) sobre el patrón de Speckle dinámico generado a partir de una muestra de pintura (corrector). La pintura tiene un exceso de masa húmeda que al evaporarse produce una pérdida de peso y genera un movimiento molecular que es detectado por el láser y capturado por la cámara como un patrón de Speckle dinámico. El resultado de este algoritmo es calculado en tiempo real y visible en el osciloscopio a medida que se adquieren los datos de la muestra. Para validar los resultados obtenidos se obtuvo mediante Excel la correlación del peso y la actividad de la muestra, donde se obtuvo una correlación del 98,68% ($R^2 = 0{,}9868$) para un modelo de regresión cuadrática con un Error Porcentual Absoluto Medio (MAPE) del 9,42%. Esto implica que el sistema desarrollado puede caracterizar el movimiento microscópico en tiempo real utilizando la Nexys Video A-7.

A los alumnos del pasado, presente y futuro de la carrera de Ingeniería Civil Eléctrica de la U.C.S.C.

Agradecimientos

Quiero partir agradeciendo a mi familia, por su constante apoyo. Me han ayudado durante todos estos años a motivarme y seguir adelante. Gracias a su forma de mirar la vida he tenido la oportunidad de disfrutar mis años de aprendizaje en la universidad y siempre mirar el lado positivo de las cosas.

También, quiero agradecer al Fondo Nacional de Desarrollo Científico y Tecnológico (FONDECYT N° PROYECTO 11201348) por su financiamiento parcial en el proyecto.

En mis años de universidad he hecho varios amigos con los que puedo contar de forma sincera, por lo que quiero extender mi agradecimiento a Sebastián Aedo, Rubén Padilla, Felipe Fierro, Ariel Soto, Daniel Nusdel, Matías Reyes, Cristóbal Leiva, Diego Marín, Mauricio Cartes y Fabrizzio Castiglione. Quienes siempre han estado presentes para escucharme y apoyarme, no solo en el ámbito universitario, sino que también fuera de él.

Quiero expresar mi gratitud también a mi profesor guía, el Dr. Jaime Cariñe, quien con su amplia experiencia y conocimientos me ayudó a conseguir el objetivo de este proyecto, siempre estuvo presente cuando lo necesitaba y dispuesto a resolver mis dudas.

Por último, quiero agradecer a todo el Departamento de Ingeniería Civil Eléctrica, por la invaluable contribución de conocimientos que me han brindado durante mis años como estudiante en la carrera. La dedicación y pasión demostrada por su parte han sido esenciales para mi desarrollo académico y desempeñan un papel fundamental en mi formación profesional.

Tabla de contenidos

Capítulo 1. Introducción

El fenómeno Speckle Dinámico es utilizado en múltiples aplicaciones; en biología, agronomía, medicina, entre otras [1]. En este fenómeno, luz coherente incide sobre una superficie y la reflexión produce una distribución de irradiancia aleatoria cuyo patrón de interferencia se denomina patrón de speckle, si la superficie que es iluminada presenta actividad microscópica entonces se observa un patrón de speckle dinámico. Este fenómeno se manifiesta como un cambio en la estructura del patrón en el tiempo y se relaciona con la evolución del medio dispersor [2], es capturado por medio de cámaras para su posterior análisis. Un ejemplo claro de esto es en la medicina donde esta técnica se utiliza para el estudio de tejidos biológicos, como la piel, para analizar la perfusión sanguínea, la dinámica de fluidos en vasos sanguíneos, la evaluación de lesiones o cambios en la microcirculación.

El FPGA (Field-Programmable Gate Arrays) es un sistema electrónico de lógica programable que contiene una gran matriz de bloques lógicos configurables que pueden realizar operaciones digitales como lógica booleana o almacenamiento de datos. También cuenta con I/O Block que proporcionan una interfaz para las señales que entran y salen del chip [3]. La tarjeta electrónica utilizada corresponde a la Nexys Video A-7 que ofrece una amplia colección de puertos y periféricos siendo relevante para este proyecto la entrada y salida con protocolo High-Definition Multimedia Interface (HDMI) [4]. Se propone la utilización de un FPGA debido a la factibilidad del sistema para realizar lectura en tiempo real de los puertos HDMI, permitiendo análisis píxel a píxel en tiempo real y en paralelo, sin requerir complejos driver adicionales a diferencia de plataformas como Arduino o Raspberry.

El procesamiento de imagen mediante algoritmo de diferencias generalizadas (GD) es un método cualitativo, que permite estudiar el comportamiento de la actividad de una muestra, comparando la intensidad de los píxeles en distintos instantes [5].

1.1. Revisión bibliográfica

La materia en estado líquido presenta un alto porcentaje de humedad que puede ser estudiada mediante un patrón de Speckle Dinámico, caracterizando la actividad microscópica de una muestra a medida que se evapora hasta secarse. La cantidad de actividad microscópica en una muestra es mayor si esta se encuentra más hidratada. Un artículo desarrollado en el año 2012 **"Procesado de patrones de Speckle mediante dos métodos para medir la desecación de semillas de Uchuva (Physalis peruviana L.)"** demostró que las semillas hidratadas mostraban mayor actividad de Speckle que las secas [6]. Este mismo principio se aplica para el proceso de secado de pintura, que con el tiempo pierde actividad microscópica.

Se han reportado trabajos que emplean métodos para evaluar el patrón de Speckle Dinámico que utilicen lógica programable básica y que permitan el análisis en tiempo real. El artículo **"Algorithm for Dynamic Speckle pattern processing"** de J. Cariñe, R. Guzmán y F.A. Torres-Ruiz del año 2016 [7] presenta un nuevo algoritmo para procesar imágenes de patrón de Speckle con un rendimiento superior en comparación a otros métodos y utilizando menos recursos computacionales. Esto permite un procesamiento en tiempo real de patrones de speckle cuantitativos.

En la Universidad Católica de la Santísima Concepción se desarrolló un sistema óptico portable, el cual fue analizado por Daniel Nusdel en su tesis denominada **"Desarrollo de sistema óptico portable para análisis de movimiento microscópico"**. En el proyecto se llegó a una correlación del 93% entre un estudio gravimétrico para una muestra de pintura y el método de diferencias generalizadas (GD) [8]. Por otro lado, el artículo "**Application of dynamic speckle interferometry to the drying of coatings**" publicado en el año 2001 por Javier I. Amalvy, Carlos A. Lasquibar, Ricardo Arizaga, Héctor Rabal y Marcelo Trivi, llegó a resultados en donde las curvas de la evolución temporal del Speckle tenían la forma adecuada y se comparaban favorablemente con las curvas de secado gravimétrico [9]. Por último, el artículo **"Localized analysis of paint-coat drying using dynamic speckle interferometry"** obtuvo un error promedio del 4,34% cuando compararon las mediciones ópticas propuestas con las mediciones gravimétricas [10]. A partir de las tres referencias se concluye que la comparación entre un estudio gravimétrico y la actividad de una muestra si presentan una relación, la cual es coherente con el proceso de secado y se puede utilizar como un factor determinante para saber si nuestro sistema permite caracterizar el movimiento microscópico.

La Tesis desarrollada por Martin Hultman en el año 2021 denominada **"Real-time multi-exposure laser speckle contrast imaging of skin microcirculatory perfusio"**, utiliza un FPGA para

procesar datos en tiempo real mostrando una mejora en la velocidad y precisión de la medición de la perfusión cutánea [11]. Por otro lado, un objeto de conferencia presentado el año 2011 titulado **"Speckle signal processing through FPGA"** en el evento II Congreso de Microelectrónica Aplicada, desarrollado por E. Todorovich, M. Vázquez, E. Cozzolino, F. Ferrara, G. J. A. Bioul, A. L. Dai Para, L. I. Passoni, concluyó que la implementación de un FPGA Xilinx Virtex-6 para procesar múltiples instancias de muestras de "Time History Speckle Pattern" (THSP) tiene un gran potencial para aplicaciones en tiempo real con recursos usados de manera eficiente en comparación a software de alto nivel como MATLAB [12]. A partir de los últimos dos artículos revisados, se llega a la conclusión de que implementar un sistema para el procesamiento de imágenes en tiempo real por medio de un FPGA es una tarea factible.

1.2. Motivación

En el Laboratorio de Optoelectrónica de la Universidad Católica de la Santísima Concepción (UCSC) se está desarrollando un sistema óptico portátil y compacto que pueda llevar a cabo análisis de movimiento microscópico. El modelo actual utiliza una cámara con sensor "Complementary metal-oxide-semiconductor" (CMOS) y una interfaz "Universal Serial Bus" (USB) para la adquisición de datos. Estos datos seguido de un procesamiento por lotes, esto significa que se adquiere una cierta cantidad de imágenes durante un periodo de tiempo determinado. Estos datos son almacenados y procesados posteriormente en MATLAB, aplicando algoritmos que ayuden a estudiar la dinámica del Speckle.

Se busca modificar este modelo portátil para que pueda realizar un algoritmo en tiempo real. Para lograrlo se plantea la utilización de un FPGA que adquiera la señal de la cámara y la procese aplicando algoritmos a medida que los datos ingresan.

También se plantea el reemplazo de la cámara con sensor CMOS por una con sensor "Charge-coupled device" (CCD) disponible en el laboratorio. Las cámaras con sensor CMOS tienden a tener peor calidad de imagen en situaciones de poca luz [13]. Además, el modelo de cámara con sensor CCD disponible nos permite trabajar directamente con el protocolo HDMI del FPGA.

Las mejoras planteadas para este sistema permitirán elaborar nuevos desafíos que impliquen la optimización del diseño presentado.

1.3. Objetivo General

Por lo planteado anteriormente, este trabajo tiene por objetivo general: Procesar en tiempo real imágenes capturadas por una cámara CCD con salida HDMI utilizando un FPGA y caracterizar la dinámica de la muestra mediante el procesamiento de diferencia generalizada (GD) sobre el patrón de Speckle.

1.4. Objetivos específicos

- Implementar un hardware que permita la conexión de una cámara CCD y un monitor por medio de protocolo HDMI en FPGA Nexys video A-7.
- Evaluar respuesta del píxel ante variaciones de intensidad lumínica.
- Aplicar diferencia generalizada para el procesamiento de imagen de Speckle en tiempo real.
- Analizar la dinámica de una muestra según su peso en base al secado de pintura.

1.5. Metodología

Para generar el Speckle Dinámico se utiliza un láser conectado a una fibra óptica la cual propaga una luz coherente que ilumina la superficie donde será colocada la muestra de pintura. Junto al láser se dispone una cámara CCD encargada de capturar este patrón de actividad y enviar los datos a un FPGA mediante una conexión HDMI. La tarjeta FPGA se encargará de recibir la señal y de aplicarle un algoritmo de procesamiento de imagen que podrá ser analizado en tiempo real mediante sus salidas “Peripheral Module” (PMOD) y un osciloscopio, además de mostrar en un monitor la imagen capturada por la cámara CCD.

La primera prueba consistirá en evaluar la variación de intensidad de dos pixeles ubicados en dos áreas distintas. Para ello, la cámara estará enfocando un aro de luz que irá aumentando su intensidad lumínica progresivamente mediante un potenciómetro. Por otro lado, mediante la botonería del FPGA se desplazarán dos cuadrados observables en el monitor donde cada uno representa el píxel que se está estudiando. De este modo, uno de los pixeles se encontrará directamente sobre un LED del aro de luz mientras que el otro píxel será ubicado lejos, los datos de intensidad de cada píxel serán monitoreados mediante un osciloscopio que recibe la información por medio de un “Digital-Analog converter” (DAC) conectado al FPGA.

La segunda prueba consiste en medir la dinámica de una muestra de pintura que se encontrará sobre una balanza con el fin de medir tanto la actividad microscópica como la variación de peso en el

tiempo. Se aplicará con el FPGA la diferencia generalizada del Speckle dinámico en tiempo real y se comparará su comportamiento en el tiempo con la variación de peso de la muestra, obteniendo así una correlación entre los datos que determinarán la factibilidad del FPGA como procesamiento de imagen para un patrón de Speckle Dinámico.

1.6. Alcances y limitaciones

- Todos los materiales necesarios para el desarrollo de este proyecto se encuentran en el Laboratorio de Optoelectrónica de la Universidad Católica de la Santísima Concepción.
- Para el código de captura y entrega de datos HDMI mediante FPGA se utilizó y modificó el diseño entregado por el usuario Hamsternz del portal Github, los derechos de autor corresponden a Michael Alan Campo [14].

Capítulo 2. Marco Teórico

En este capítulo se describirán las características principales de un FPGA, del protocolo HDMI y del fenómeno de Speckle, con la finalidad de explicar las etapas y consideraciones necesarias que se deben llevar a cabo para poder establecer una conexión entre la cámara CCD, el FPGA, un monitor y el sistema óptico. También se explicará en que consiste el procesamiento de imagen digital para la aplicación de algoritmos en el estudio del patrón de Speckle dinámico mediante un análisis cualitativo realizado con el método de diferencias generalizadas.

2.1. Procesamiento de una imagen digital.

El procesamiento digital de imágenes se puede definir como un conjunto de procesos y técnicas que tienen el objetivo de descubrir o resaltar información de una imagen utilizando herramientas como una computadora [15]. Esto significa la manipulación de los pixeles de una imagen tanto para, almacenar, modificar, recortar, posicionar, etc.

En un modelo RGB cada color se descompone en sus componentes primarios: rojo, verde y azul. Las imágenes representadas en este modelo se estructuran en tres canales distintos, uno para cada color primario [16]. Podemos referirnos a esta imagen como una matriz de tres dimensiones definidas como Ancho, Alto y Canales. Las dimensiones de Ancho y Alto representan el tamaño de la imagen, es decir, el número total de píxeles, mientras que la dimensión Canales representa los colores. Esta estructura genera la matriz $f(x, y, z)$ que se muestra en la Fig.2.1.1.

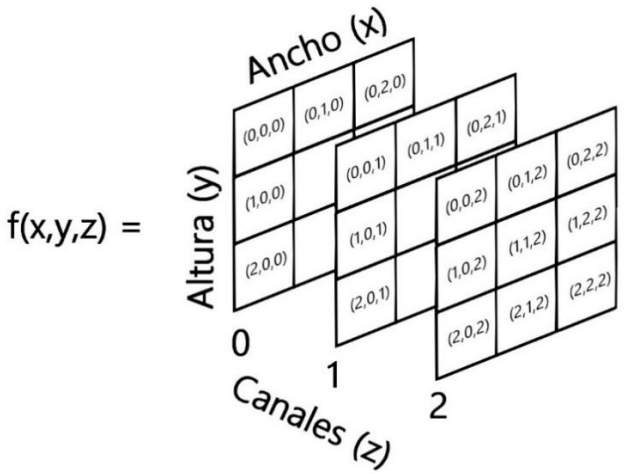


Fig.2.1.1: Matriz de los canales Rojo, Verde y Azul.

Cuando las matrices RGB se introducen en un monitor estas se combinan generando una imagen de color compuesta. El número de bits usados para representar la intensidad del color del píxel en el espacio RGB es llamado profundidad siendo 8 bits los más comunes entregando 256 tonalidades distintas. Al utilizar el color rojo, verde y azul, los pixeles deben tener una profundidad de 24 bits, esto significa un total de $(2^8)^3 = 16.777.216$ tonalidades de color distintas [16].

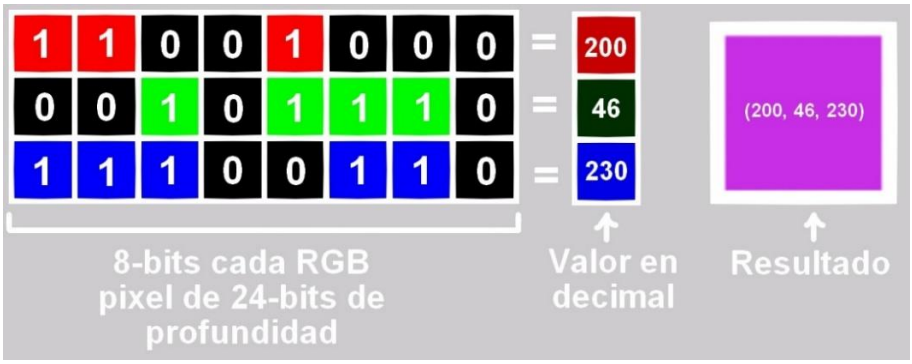


Fig.2.1.2: Representación del color para RGB con 8-bits.

La Fig.2.1.2 muestra una combinación binaria de 8 bits para cada color y su resultado en base decimal. El píxel combina estos tres resultados y entrega un color con una profundidad de 24 bits. Esto visto en el modelo de la matriz anterior, quedaría como la Fig.2.1.3.

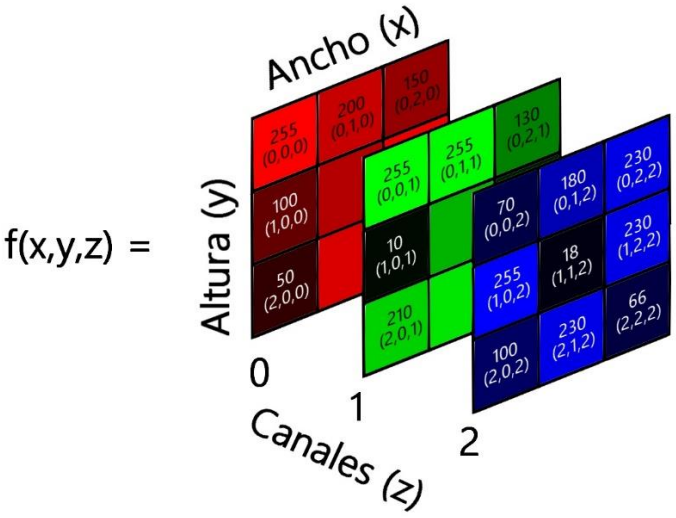


Fig.2.1.3: Matriz RGB con sus distintas tonalidades, profundidad de 8-bits para cada canal.

2.2. Cámara con sensor CCD.

Un sensor CCD (Dispositivo de carga acoplada, Charge-Coupled Device) se compone en general de fotodiodos (píxeles), carga (electrones), registros de desplazamiento verticales y un registro de almacenamiento horizontal como se observa en la Fig.2.2.1. El proceso inicia con la captura de imagen por medio de los píxeles seguido por la transmisión de los valores hacia el registro vertical donde se copian de manera secuencial, línea por línea, hacia el registro horizontal quien los va desplazando hacia una salida donde son amplificados y posteriormente convertidos en datos digitales [17].

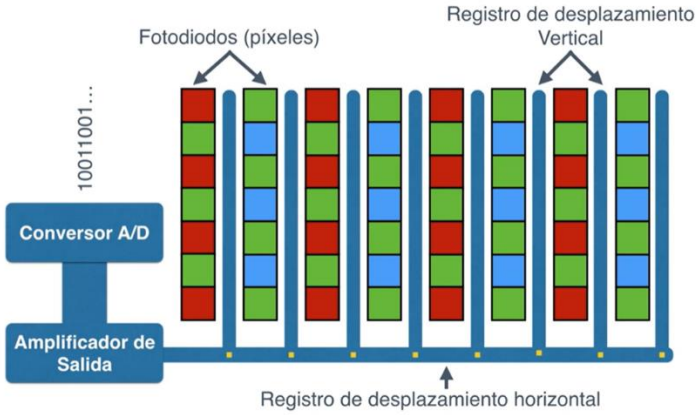


Fig.2.1.1: Sensor CCD [17].

Los sensores CCD pueden clasificarse por su estructura, siendo los más conocidos “Full-Frame”, “Frame-Transfer” e “Interline-Transfer” Fig.2.2.2.

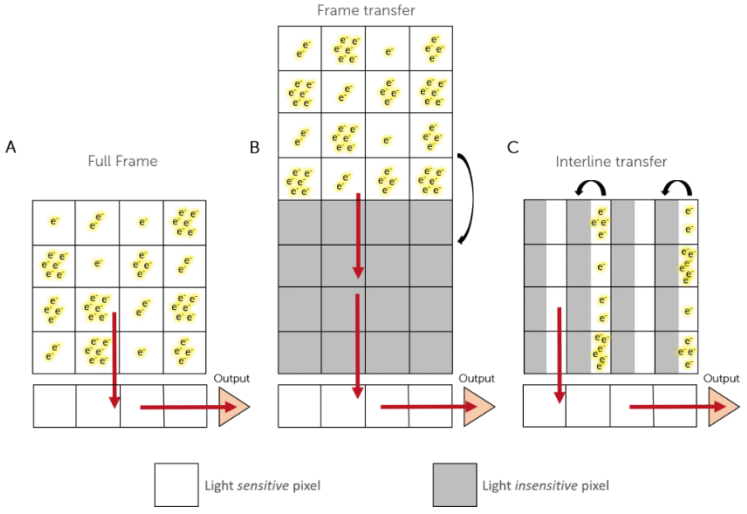


Fig.2.1.2: Distintas estructuras de un sensor CCD [17].

Los sensores CCD con arquitectura "**Full-Frame**" (Fig.2.2.2. A) capturan fotones en toda la matriz durante la exposición con un factor de llenado del 100%. Requieren de un obturador sincronizado para controlar el periodo de exposición y transferencia de carga. La arquitectura "**Frame-Transfer**" (Fig.2.2.2 b) divide la matriz en áreas fotoactivas y de almacenamiento, lo que permite la lectura simultanea de la trama actual y la siguiente sin necesidad de obturador durante la transferencia aumentando la velocidad de los fotogramas. Por último, en la arquitectura "**Interline-Transfer**" (Fig.2.2.2 c) cada fila activa de píxeles tiene una fila enmascarada que permite la adquisición del siguiente fotograma mientras los datos se transfieren por lo que no requieren de obturadores externos [18].

Utilizar este tipo de sensores nos entrega ventajas como una mayor calidad de imagen, un mayor rango dinámico y mejor sensibilidad a la luz, por lo que son ideales para su aplicación en lugares con poca luz. Sin embargo, como desventaja existe un incremento en el consumo eléctrico y la aparición del efecto "Smear" y "Lag" [17].

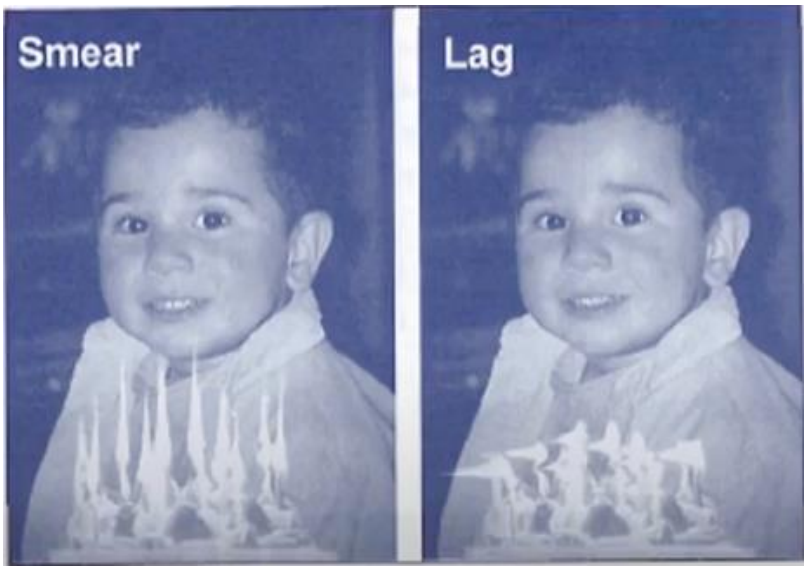
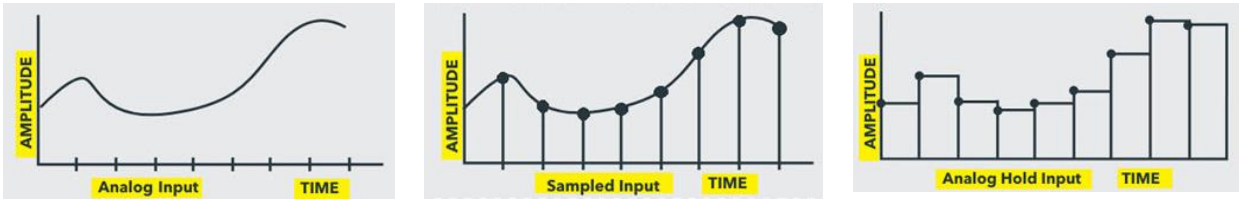


Fig.2.1.3: Efecto "Smear" y "Lag" [17].

El "Smear" se ve como un halo de luz vertical que sale de puntos de luz muy brillante, mientras que el "Lag" se produce entre la contaminación de los pixeles en objetos con movimiento, la dirección del "Lag" es a favor en la dirección del movimiento [17].

2.3. Convertidores Analógico-Digital y Digital-Analógico.

El propósito principal de un Convertidor Analógico-Digital (ADC) es tomar una señal eléctrica y convertirla en un número digital. Esto ocurre en dos etapas clave: primero, la señal física se convierte en una señal eléctrica análoga a través de un dispositivo llamado transductor. Luego, esta señal análoga se introduce en el ADC, donde pasa por un proceso de cuantificación y codificación para finalmente ser representada en formato digital. Sin embargo, antes de que la señal entre en el ADC, debe pasar por un proceso de muestreo y retención conocido como “sampling and holding” [19].



(a) Señal analógica (b) Sampling (Muestreo) (c) Holding (Retención)

Fig.2.3.1: Etapa de muestreo y retención [20].

- Muestreo: Consiste en la obtención de una serie de muestras o porciones de una señal continua en el tiempo para determinados instantes.
- Retención: Retiene durante un tiempo suficiente los valores muestreados para poder cuantificarlos.
- Cuantificación: Es el proceso que consiste en tomar la señal muestreada y convertirla en un conjunto de valores finitos.
- Codificación: Consiste en representar mediante código binario los valores obtenidos en la cuantificación [19].

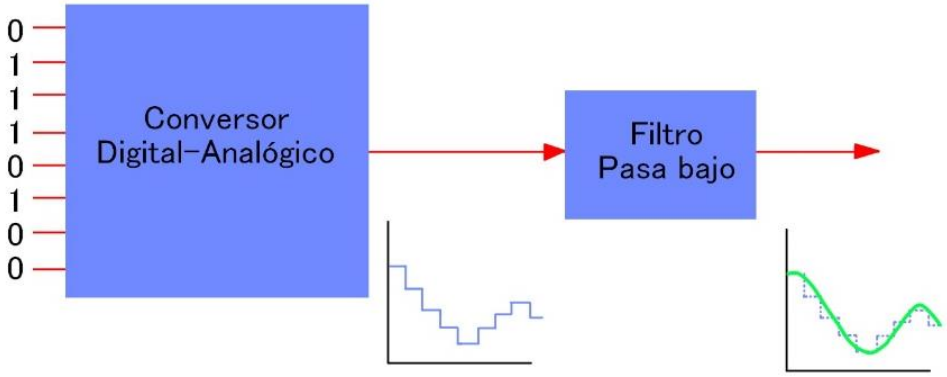


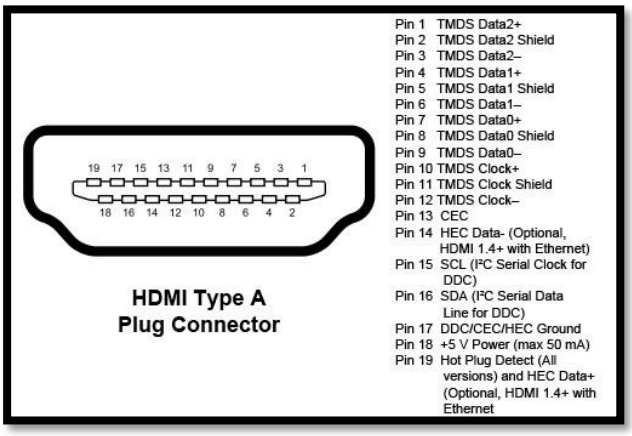
Fig.2.3.2: Etapa de muestreo y retención [20].

Por otro lado, tenemos los convertidores Digital-Analógico de la Fig.2.3.2. donde la señal digital se convierte en una señal analógica expresada en escalones que se suavizan utilizando un filtro pasa bajo [19]. En la Fig.2.3.2. podemos observar una entrada binaria al conversor que sale de forma análoga (voltaje o corriente) pero escalonada, luego ingresa al filtro pasa bajo y la señal es suavizada.

2.4. High-Definition Multimedia Interface (HDMI)

La "Interfaz Multimedia de Alta Definición" es utilizada en equipos de audio/video para la transmisión en alta definición y sin pérdida de calidad debido a la conversión o compresión de la señal. Además, permiten hacer coincidir la resolución nativa de la pantalla con la fuente de video o audio conectada píxel por píxel [21].

(a) Cable HDMI tipo A.



(b) Pines de un cable HDMI tipo A [22].

Fig.2.4.1: Cable HDMI tipo A con sus pines de conexión.

En la Fig.2.4.1, se muestran los pines de un HDMI tipo A. Los pines que se encuentran desde el pin 13 hasta el pin 19 corresponden a CEC (Consumer Electronics Control, Pin 13) para controlar o conectar otros dispositivos compatibles desde una sola fuente. El DDC (Display Data Channel) que es un protocolo de comunicación bidireccional entre un dispositivo que sea compatible con HDMI y una fuente permitiendo que esta lea los datos de identificación del periférico, consta de una señal de reloj serial o SCL (Pin 15), una línea de datos serial o SDA (Pin 16) y una conexión a tierra (Pin 17). El Pin 18 es una línea de alimentación de +5V utilizada para suministrar energía al bus DDC del dispositivo receptor cuando este no está encendido. Por último, se encuentra el HPD (Hot Plug Detection) y el HEC Data+ (HDMI Ethernet Channel) que comparten la misma línea. El HPD permite que una fuente detecte si se conecta o desconecta cualquier dispositivo habilitado para HDMI mientras

está en funcionamiento y el HEC se utiliza para la comunicación de datos Ethernet entre dispositivos compatibles [23]. Los pines del 1 al 12 corresponden a los TMDS (Transition Minimized Differential Signaling / Transición Minimizada de Señales Diferenciales) que se utilizan para transmitir las señales de video y audio desde el transmisor hasta el receptor, consta de tres canales de datos TMDS y un canal de reloj TMDS, cada canal de datos está compuesto por datos +, datos – y un blindaje de datos. El canal reloj TMDS está compuesto por un reloj +, reloj – y un blindaje de reloj. Los canales TMDS de datos se sincronizan con referencia al canal TMDS de reloj [23].

El protocolo HDMI utiliza tecnología TMDS para transmitir información desde un lugar hacia otro. TMDS es una forma de codificar la señal con la finalidad de protegerla ante la degradación a medida que viaja por el cable y utiliza tecnología para la transmisión de datos seriales de alta velocidad [21].

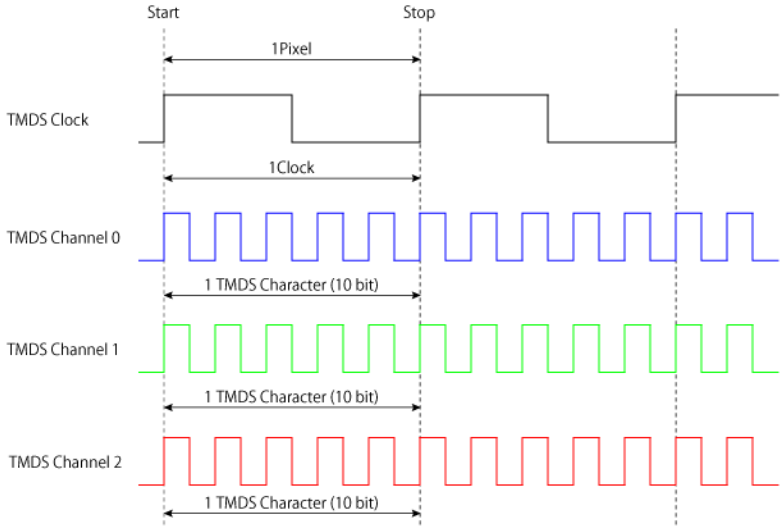


Fig.2.4.2: Ejemplo de TMDS Clock como referencia de frecuencia para los tres canales [24].

En la Fig.2.4.2. Observamos que un ciclo de reloj del canal TMDS Clock corresponde a un carácter TMDS (10 bits). El Píxel Clock viaja por el TMDS Clock y es utilizado por el receptor como referencia de frecuencia para la recuperación de datos en los tres canales de datos TMDS [25].

2.5. 8b/10b Codificación/Decodificación.

La codificación 8b/10b consiste en convertir palabras de 8 bits en símbolos de 10 bits donde los 5 bits menos significativos los codifica en un grupo de 6 bits y los 3 bits más significativos en un grupo de 4 bits. Estos dos grupos se concatenan en un símbolo de datos de 10 bits. Para delimitar los paquetes enviados de manera serial y marcar su inicio y/o final se utilizan 12 caracteres especiales denominados comas [26].

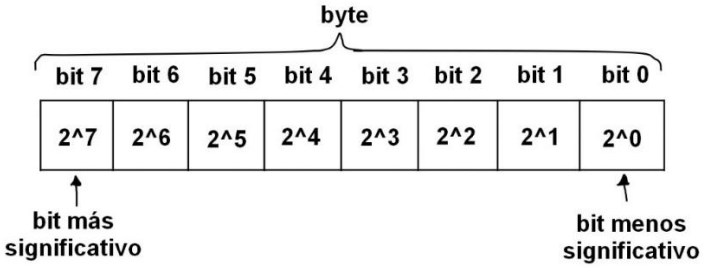


Fig.2.5.1: Byte y sus bits más y menos significativos.

La codificación 8b/10b garantiza un equilibrio de unos y ceros en la transmisión de datos mediante un método llamado “disparidad en ejecución” (Running Disparity) donde se emplean dos símbolos (+ y −) asignados a cada valor de datos. Un símbolo tiene 6 ceros y 4 unos mientras que el otro tiene 4 ceros y 6 unos. El siguiente símbolo se selecciona en base al equilibrio actual de unos y ceros para mantener la corriente directa [27].

Name	Hex	8 Bits	RD -	RD +
D10.7	EA	11101010	0101011110	0101010001
D31.7	FF	11111111	1010110001	0101001110
D4.5	A4	10100100	1101011010	0010101010
D0.0	00	00000000	1001110100	0110001011
D23.0	17	00010111	1110100100	0001011011

Fig.2.5.2: Tabla con ejemplos de símbolos 8b/10b [27].

En la Fig.2.5.2, los nombres de los símbolos de datos se representan como Dx.y, donde “x” tiene un rango de 0-31 e “y” un rango de 0-7 [26] (en total 256 combinaciones). En la tercera columna tenemos la representación binaria en 8 bits de la señal que será codificada, “RD −” corresponde a Disparidad en ejecución “Negativa” y “RD +” es Disparidad en ejecución “Positiva”.

Name	Hex	8 Bits	RD -	RD +
K28.0	1C	00011100	0011110100	1100001011
K28.1	3C	00111100	0011111001	1100000110
K28.2	5C	01011100	0011110101	1100001010
K28.3	7C	01111100	0011110011	1100001100
K28.4	9C	10011100	0011110010	1100001101
K28.5	BC	10111100	0011111010	1100000101
K28.6	DC	11011100	0011110110	1100001001
K28.7	FC	11111100	0011111000	1100000111
K23.7	F7	11110111	1110101000	0001010111
K27.7	FB	11111011	1101101000	0010010111
K29.7	FD	11111101	1011101000	0100010111
K30.7	FE	11111110	0111101000	1000010111

Fig.2.5.3: Tabla de ejemplo para caracteres K de control válidos [27].

Los 12 símbolos especiales que reciben el nombre de “comas” los podemos ver representados en la Fig.2.5.3. con el carácter K. Al igual que los carácter D se representan con el nombre Kx.y donde “x” puede tomar los valores 23, 27, 28, 29 y 30 e “y” tiene un rango de 0-7 [26]. Los símbolos (bits transmitidos) se delimitan con estas “coma” donde la secuencia suele ser ajustable por el transceptor y en algunos casos puede estar predefinida [27]. En la Fig.2.5.4 observamos un ejemplo de una secuencia de símbolos Dx.y delimitados con una coma Kx.y (K28.7).

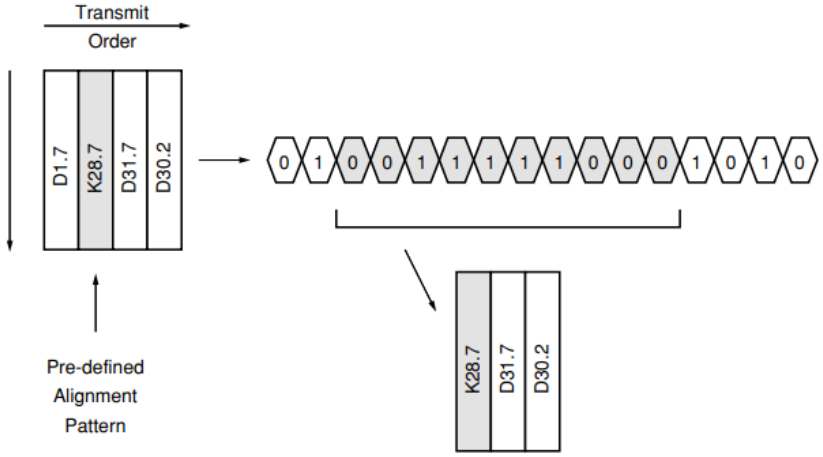


Fig.2.5.4: Tabla de ejemplo para caracteres K de control válidos [27].

2.6. Serializador y Deserializador

Un serializador es una forma de convertir los datos que entran de forma paralela a un formato en serie. El formato paralelo se refiere a que los bits de datos están presentes simultáneamente en cables individuales mientras que el formato en serie los datos se presentan de manera secuencial en el tiempo que pasan por un solo cable o circuito [28] (Fig.2.6.1).

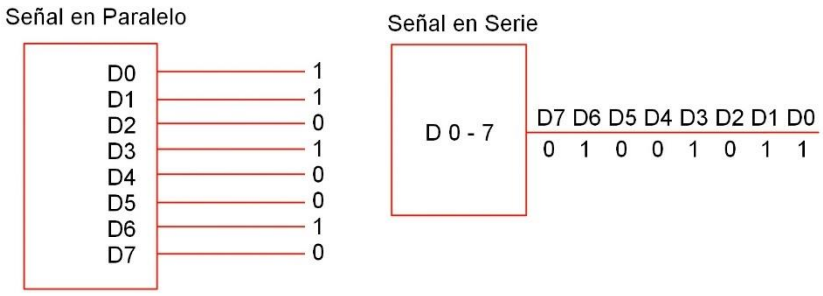


Fig.2.6.1: Comparación de una señal en paralelo y una señal en serie.

Si bien existen distintos dispositivos que pueden utilizarse para realizar la serialización, este proyecto tomará la explicación general utilizando un registro de desplazamiento “PISO” (parallel-in/serial-out). La etapa consta de un Flip-Flop tipo D (FF-D) para almacenamiento, dos selectores AND (uno con una entrada NOT) y un selector OR para determinar si los datos paralelos se cargaran en el FF-D o si los datos cargados se desplazaran hacia la derecha (salida serial) [28] (Fig.2.6.2). Estos elementos se van replicando, dependiendo de la cantidad de etapas requeridas [28]. Por ejemplo, si tenemos una codificación 8b/10b en la salida vamos a tener 10 bits que deben ser serializados, por ende, se utilizarían 10 FF-D, 20 AND, 10 OR y 10 NOT.

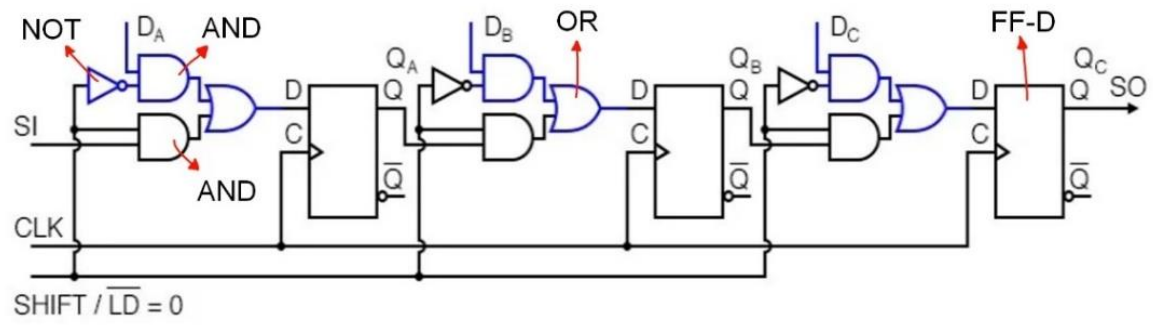


Fig.2.6.2: Registro de desplazamiento para PISO, etapa de carga paralela [28].

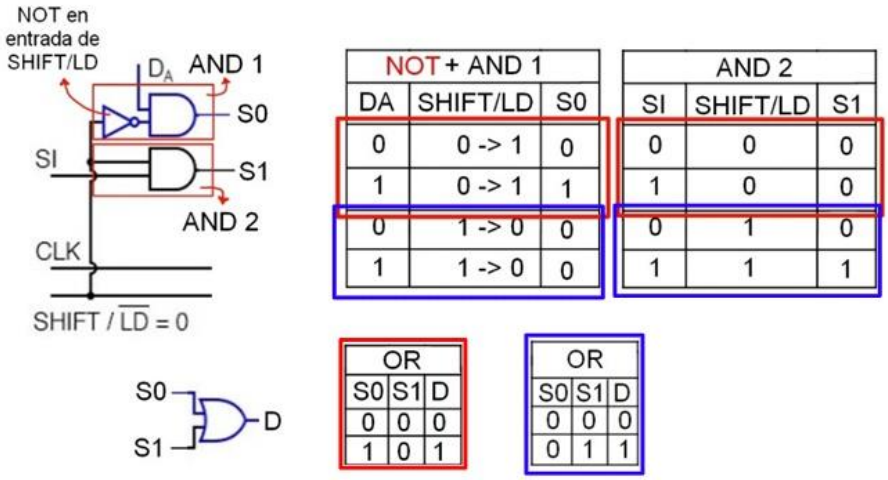


Fig.2.6.3: Tabla de verdad para distintos estados de $\mathrm{SHITF}/\overline{\mathrm{LD}}$ y OR.

Respecto a la Fig.2.6.3 los cuadrados rojos en las tablas señalan los casos en que $\mathrm{SHITF}/\overline{\mathrm{LD}} = 0$ y los cuadrados azules cuando $\mathrm{SHITF}/\overline{\mathrm{LD}} = 1$.

Cuando $\mathrm{SHITF}/\overline{\mathrm{LD}} = 0$ la salida S_0 del "AND1" puede variar entre 0 y 1 mientras que la salida S_1 del "AND2" siempre será 0, ambas ingresan a un OR que realiza una suma lógica con salida "D". El resultado de esta suma lógica va a ser igual al valor de la salida del "AND1" y pasará a ser almacenado en un FF-D que es la entrada de un "AND2" vecino.

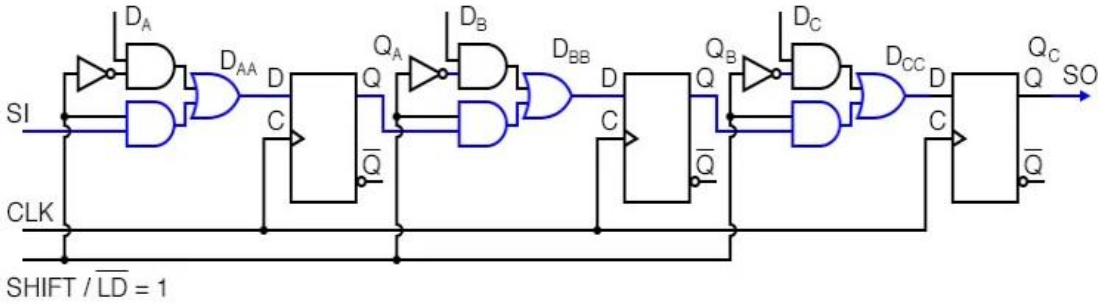


Fig.2.6.4: Registro de desplazamiento para PISO, etapa de desplazamiento [28].

Cuando $\mathrm{SHITF}/\overline{\mathrm{LD}} = 1$ la secuencia que tiene mas peso sería la de color azul (Fig.2.6.4). En este caso la salida S_0 del "AND1" siempre será 0 y el valor de salida S_1 del "AND2" ahora tendrá peso en la sumatoria lógica del OR haciendo que se desplacen los bits almacenados en los FF-D hacia la salida "SO".

En resumen, todos los “AND1” reciben un bit de información al mismo tiempo cuando $\mathrm{SHITF}/\overline{\mathrm{LD}} = 0$ que se almacena en los FF-D. Cuando $\mathrm{SHITF}/\overline{\mathrm{LD}} = 1$ los bits almacenados en los FF-D son desplazados gracias a los “AND2” y “OR” hasta la salida SO.

Por otro lado, un SIPO (Serial-input / Parallel-output) corresponde al proceso contario, es decir, convertir señales seriales a su forma paralela.

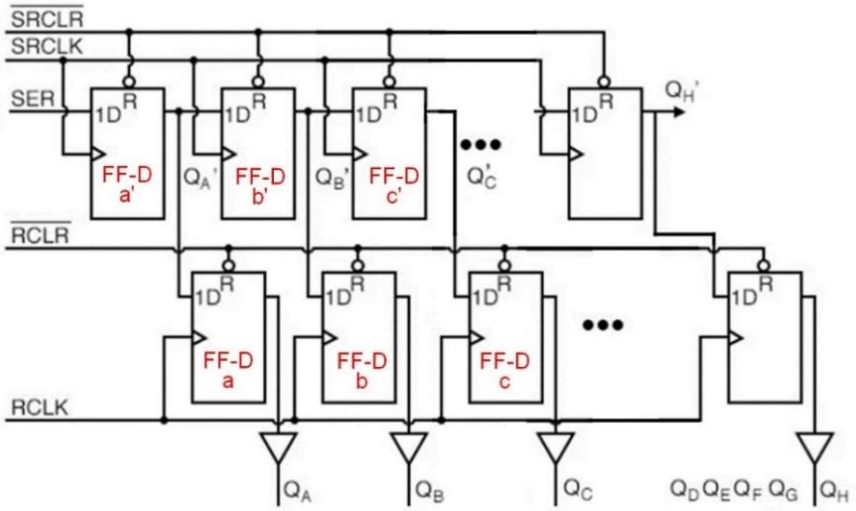


Fig.2.6.5: Estructura para la conversión SIPO (serial in / parallel out) [28].

Uno de los diseños presentados es el de la Fig.2.6.5 se identifica un juego de Flip-Flop tipo D (FF-D) conectados entre ellos. Los FF-D ubicados en la parte superior reciben la señal serial bit a bit por la entrada “SER” y son controlados por un reloj SRCLK. Cada vez que SRCLK realiza un flanco de subida se desplaza el bit que entra en el primer FF-D hacia el siguiente FF-D de modo que cuando se hayan cargado todos los bits estos se encontrarán almacenados en los FF-D inferiores. Los FF-D inferiores son controlados con el RCLK y en cada flanco de subida enviarán los bits de forma paralela y sincronizada por cada una de las salidas.

2.7. Field Programmable Gate Array (FPGA).

Un FPGA (Matriz de puertas programables en campo) es un sistema electrónico de lógica programable que utiliza bloques lógicos preconstruidos configurables y programables mediante software [29]. Una de las características importantes del FPGA es su capacidad de adaptarse a las necesidades del usuario permitiendo programarlos cada vez que sea necesario. La arquitectura de un FPGA puede variar según el fabricante, pero su composición básica consta de Bloques lógicos configurables (CLB), Bloques de entrada/salida (I/O Blokcs) y Recursos de interconexión [30].

- Bloque de lógica programable (CLB): implementan funciones lógicas.
- Recursos de interconexión: Enrutamiento programable que conecta estas funciones lógicas.
- Bloques de entrada y salida (I/O Blocks): Conectados a los CLB a través de interconexiones de enrutamiento y que hacen conexiones fuera del chip [31].

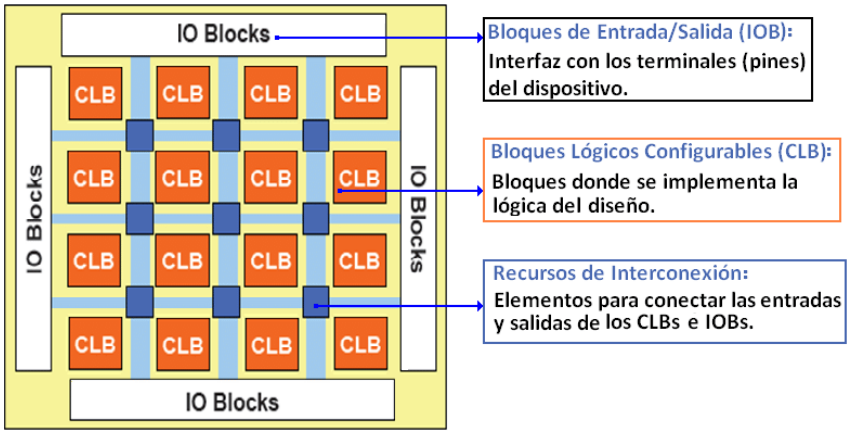


Fig.2.7.1: Estructura básica de un FPGA [30]

Dentro de un FPGA existe un sincronismo entre los elementos conectados basado en una señal de reloj que debe llegar a todos estos elementos en fase. La distribución del reloj en el FPGA es denominada como “Clock tree” que se repite en varias regiones asegurando el sincronismo entre los elementos secuenciales [29].

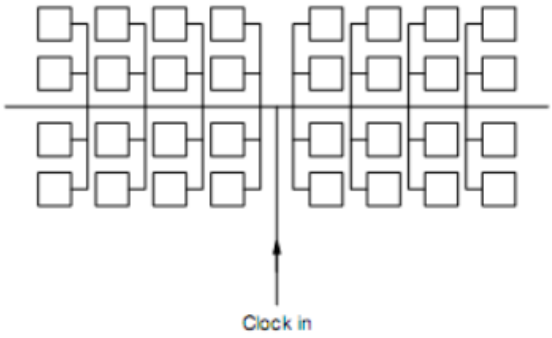


Fig.2.7.2: “Clock tree” distribución de un reloj en una región el FPGA [29].

Hoy en día la estructura de un FPGA varía según el fabricante y nos podemos encontrar con funciones específicas como memorias (BRAM), procesamiento digital de señales, control de reloj, etc [32]. Estas funciones específicas permiten trabajar de forma más sencilla y eficiente al momento de programarlas mediante software.

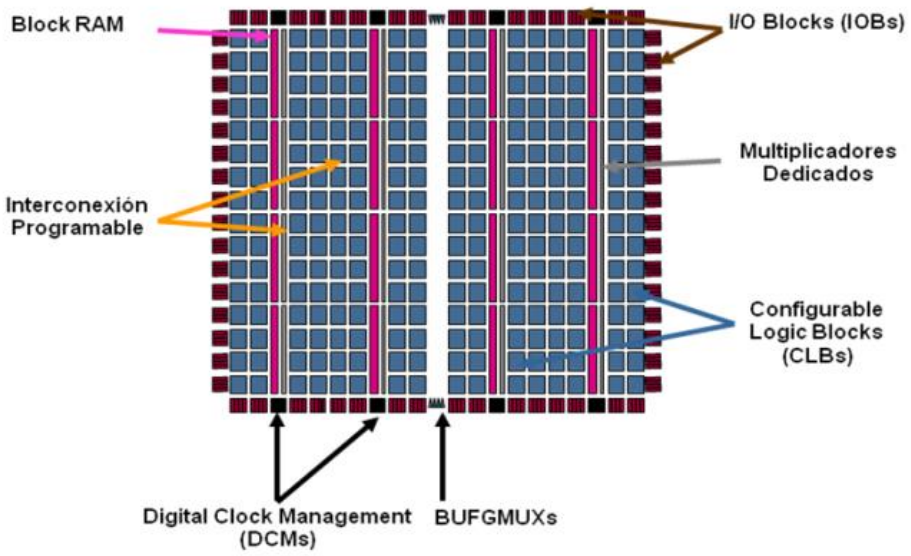


Fig.2.7.3: Estructura de un FPGA Xilinx [32].

Para programar un FPGA se utiliza un lenguaje conocido como “Hardware Description Language” (HDL). Dentro de los más utilizados se encuentran “VHDL” y “Verilog” [33]. Ambos lenguajes tienen sus propias peculiaridades en cuanto a sintaxis y enfoque en el diseño de circuitos digitales, ofreciendo diferentes ventajas en términos de estilo de programación y resolución de problemas de diseño digital. VHDL se basa en derivados de Ada y Pascal mientras que Verilog se basa en C [34].

2.8. Vivado Design Suite.

Vivado Design Suite es un entorno de desarrollo integrado (IDE / Integrated development environment) que facilita el diseño y la interacción con dispositivos como los FPGA mediante un lenguaje de descripción de hardware como VHDL y Verilog. Este conjunto de herramientas proporciona una interfaz gráfica que simplifica la creación de diseños complejos de manera visual [35].

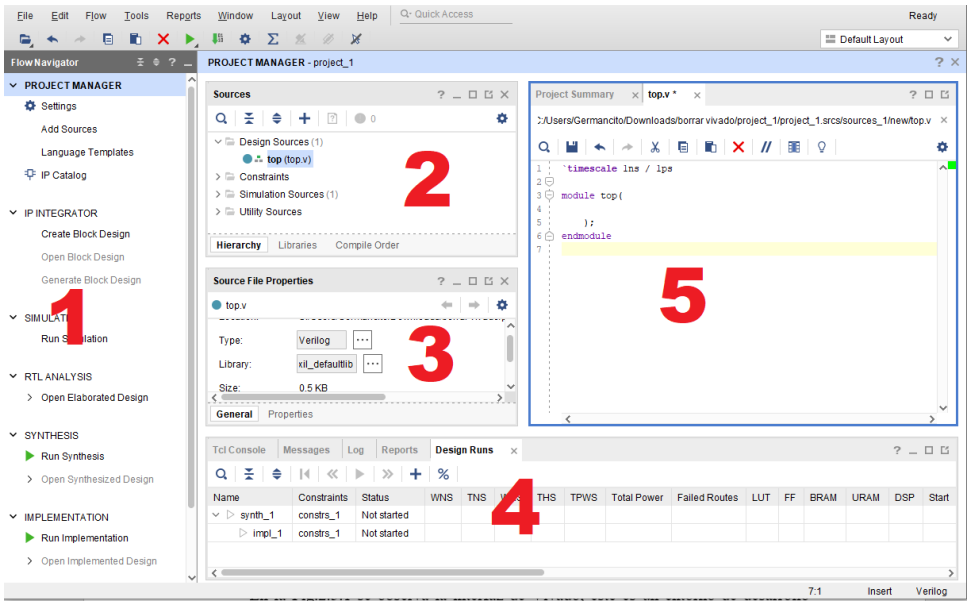


Fig.2.8.1: IDE Vivado para diseño y síntesis de hardware.

1. Navegador: Acceso rápido a los pasos del flujo de diseño y opciones de configuración.
2. Archivos de fuente: Vista gráfica de las fuentes y otros aspectos para las fuentes de diseño.
3. Propiedades de los ítems seleccionados.
4. Muestra la información actual de la consola, como reportes y detalles relacionados a las operaciones.
5. Editor: Contiene información de la edición del proyecto.

En la Fig.2.8.1 se observa la interfaz de Vivado que se utiliza en parte para diseñar y sintetizar el hardware que se implementa en la lógica programable de las FPGA's. Permite trabajar con diseños "Register-Transfer Level" (RTL) de los que destacan "Very High Speed Integrated Circuit" (VHDL) y Verilog [35].

En el apartado de navegador se encuentra la pestaña IP Catalog (Intellectual Property Catalog) que permite ingresar al catálogo IP de Vivado, en este apartado encontramos módulos predefinidos y reutilizables que se pueden integrar en un diseño FPGA para realizan funciones específicas como controlar memorias, periféricos, y otros bloques de lógica comúnmente utilizados.

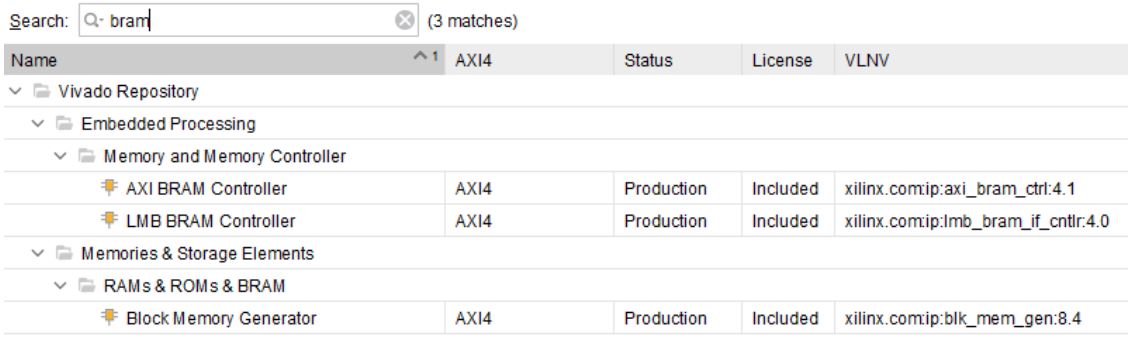


Fig.2.8.2: Interfaz IP Catalog de Vivado.

Como ejemplo, podemos observar la Fig.2.8.2 donde se busca un módulo que facilite la utilización de una Block RAM. Cuando se selecciona la opción con la que se quiere trabajar se abre una nueva pestaña (Fig.2.8.3.) que permite ajustar los parámetros del módulo. En el ejemplo se ajusta el ancho de bits y la profundidad de una BRAM.

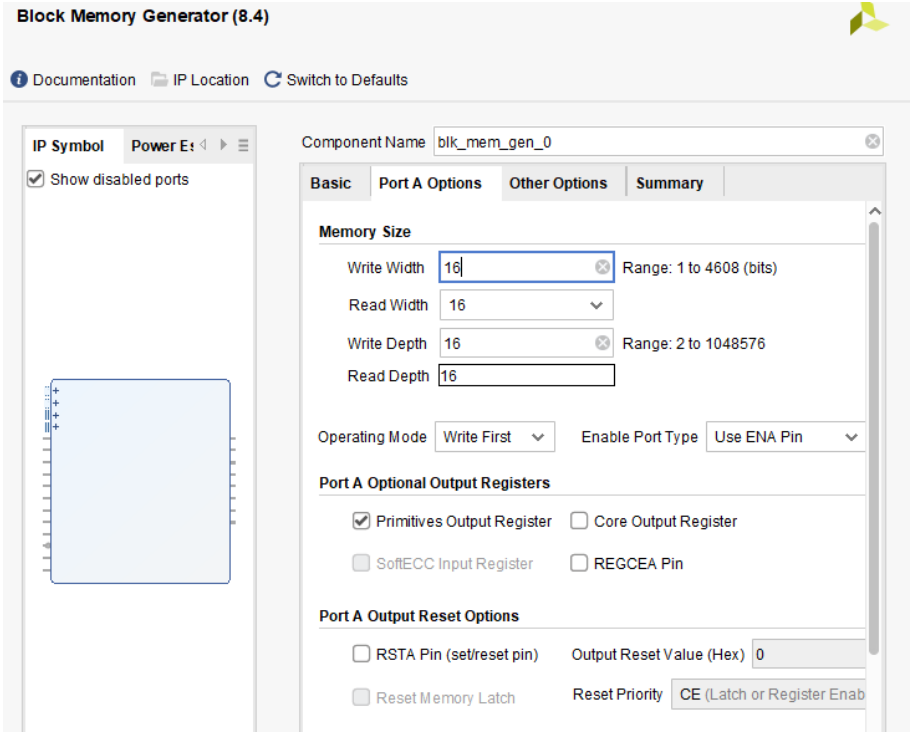


Fig.2.8.3: Pestaña para especificar parámetros y funciones del módulo.

2.9. Video Timings.

Durante la transmisión de una imagen en un monitor se cumplen dos etapas, la etapa de dibujo del píxel (Píxeles Activos) y la fase en blanco (Blank), ambas cuentan con periodos de tiempo estandarizados que consideran la resolución y la tasa de refresco [36].

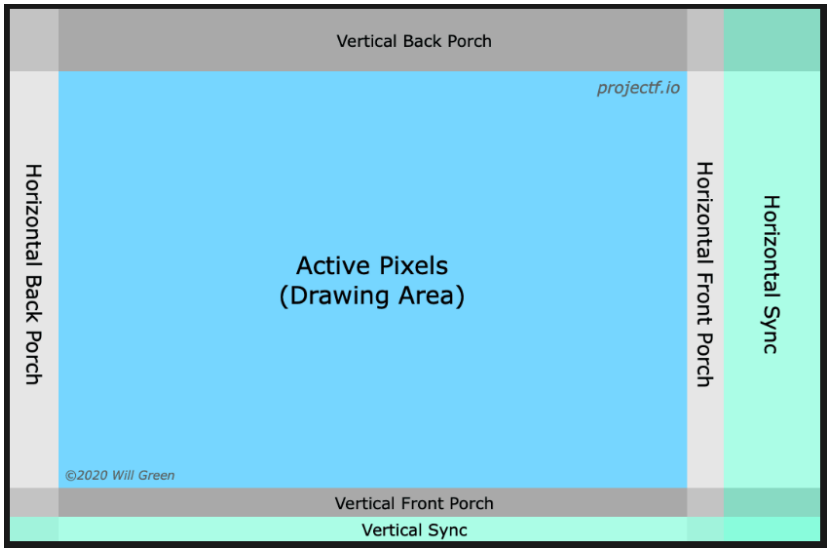
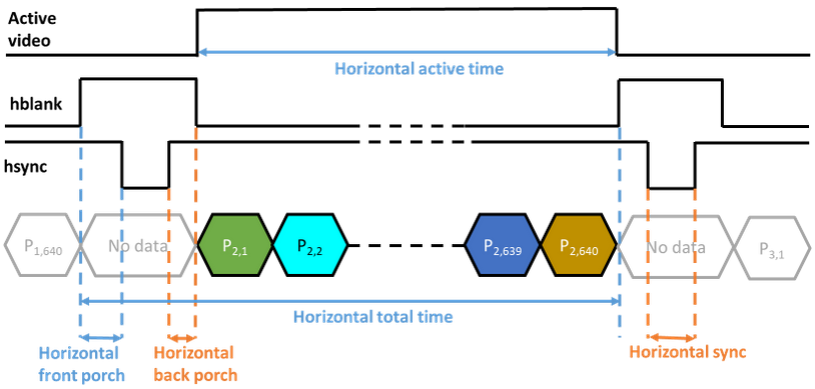
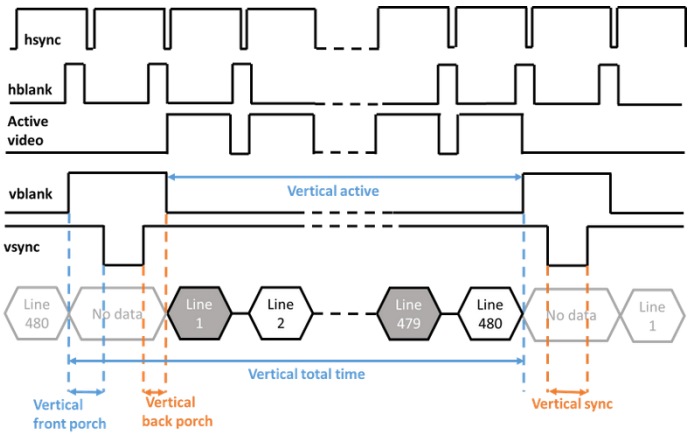


Fig.2.9.1: Resumen de las señales de video [36].

En la Fig.2.9.1 la fase en blanco se encuentra tanto para los pixeles horizontales como verticales y se subdividen en tres componentes: front porch, back porch y el área de sincronismo.



(a) Tiempos para los píxeles en dirección horizontal [37].



(b) Tiempo para las líneas verticales [37].

Fig.2.9.2: Timing de video para resolución 640x480 píxeles [37].

Si observamos la Fig.2.9.2. se muestra un ejemplo para resolución de 640x480 píxeles. Para el caso (a) se inicia en un píxel horizontal $P_{2,1}$ esto significa que el píxel está en la línea vertical 2 pero en la posición horizontal 1. Cuando se llega al $P_{2,640}$ comienza la fase en blanco horizontal con las etapas "horizontal front porch", "horizontal sync" y "horizontal back porch" respectivamente. Terminada esta etapa comienza el píxel $P_{3,1}$ que corresponde a la línea vertical 3 y posición horizontal 1. En (b) observamos que el proceso de (a) se repite hasta llegar a la línea 480 donde inicia la fase en blanco vertical y se prepara al monitor para el ingreso de una nueva imagen.

1920x1080p 60Hz			
Reloj	Frecuencia		
Píxel Clock	148,5 MHz		
TMDS Clock	1485,0 MHz		
Píxeles horizontales		Líneas verticales	
Parámetro	Píxeles	Parámetro	Líneas
Píxeles activos	1920	Píxeles activos	1080
Porch Frontal	88	Porch Frontal	4
Banda de sincronismo	44	Banda de sincronismo	5
Porch Trasero	148	Porch Trasero	36
Total en blanco	280	Total en blanco	45
Píxeles totales	2200	Píxeles totales	1125

Fig.2.9.3: Parámetros para resolución 1920x1080 60 Hz.

Los parámetros de la Fig.2.9.3 son para resolución de 1920x1080p con una tasa de refresco de 60 Hz obtenidos de Project F [36].

2.10. Patrón de Speckle Dinámico.

El Speckle se produce cuando una superficie rugosa es iluminada con un haz de luz coherente. Al reflejarse la luz se produce una dispersión que genera interferencia entre ondas resultando en un patrón de intensidad característico compuesto por puntos brillantes y oscuros distribuidos aleatoriamente [38]. Este fenómeno lo podemos ver como una imagen granulada. Fig.2.10.1.

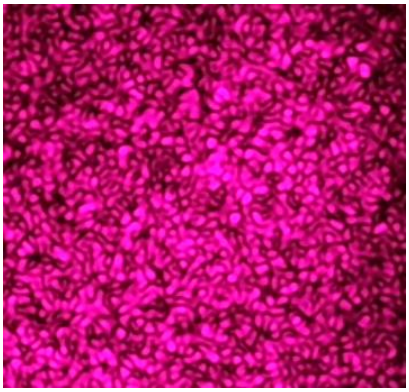


Fig.2.10.1: Patrón de Speckle para superficie metálica.

Los patrones de Speckle se pueden clasificar en dos tipos; estáticos y dinámicos. Los patrones estáticos se crean a partir de estructuras que no varían en el tiempo generando un patrón invariable. Por otro lado, los patrones dinámicos se originan cuando la luz interactúa con una estructura que experimenta cambios a lo largo del tiempo, por ejemplo, el secado de pintura [38]. Este trabajo se centrará en el Speckle dinámico.

Los laser son dispositivos que emiten luz coherente y monocromática. Luz coherente es cuando los fotones que la componen se encuentran en fase y luz monocromática cuando comparten el mismo color [39]. Podemos observar en la Fig.2.10.2. (a) un patrón de ondas de luz coherente y en (b) un patrón de ondas de luz incoherente.

(a) Luz coherente (b) Luz incoherente

Fig.2.10.2: Patrones de onda de luz [39].

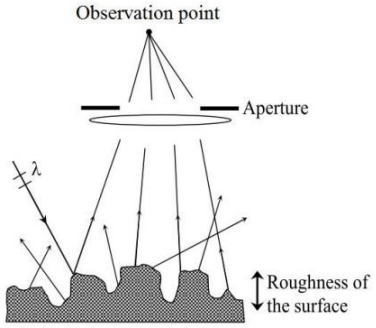


Fig.2.10.3: Esquema representativo de cómo se genera un Speckle [40].

La luz que incide en cada punto del receptor (como una cámara) proviene de varios puntos en el objeto y la longitud del recorrido de la luz desde cada punto de la superficie hasta el plano del observador depende de la forma de la superficie en ese punto específico. Cuando existen variaciones significativas en la superficie donde incide la luz se generan efectos de interferencia aleatoria que conocemos como Speckle [40].

Los algoritmos para procesar las imágenes de Speckle dinámico pueden ser cuantitativos o cualitativos. Los algoritmos cuantitativos permiten analizar el movimiento total de una muestra a partir de un valor único mientras que el algoritmo cualitativo permite generar una imagen de actividad que se basa en la variación de intensidad del patrón. Cuantitativamente el algoritmo más exitoso es el "Time History Speckle Pattern" (THSP) y cualitativamente existen varios algoritmos, siendo los más utilizados "Análisis de Contraste de Speckle con Láser" (LASCA), "Diferencias Generalizadas" (GD), "Diferencias Generalizadas Pesadas" (WGD) y Fujii [7].

El método utilizado en este proyecto se basa en la diferencia generalizada, obtenido mediante la ecuación (1) proporcionada por el artículo "**Dynamic laser speckle to detect motile bacterial response of Pseudomonas aeruginosa**" [5].

$$GD(x,y) = \sum_{k}\sum_{l}|I_k(x,y) - I_{k+l}(x,y)| \qquad (1)$$

Donde $I(x,y)$ es la intensidad en las coordenadas x e y de la matriz imagen, k y l son los índices de la imagen (cantidad de imágenes utilizadas).

Este algoritmo permite realizar un análisis cuantitativo y cualitativo, ya que, si bien entrega un valor en magnitud que representa la cantidad de actividad microscópica de una muestra, también se

puede realizar un mapa de movilidad generado a partir de las diferencias de intensidades de cada píxel e identificar patrones visuales.

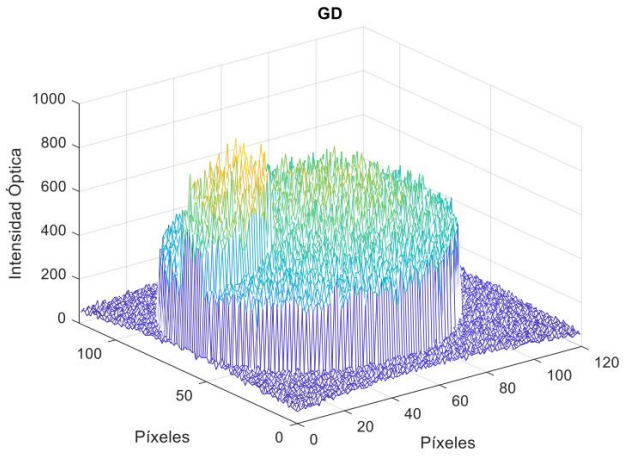


Fig.2.10.4: Mapa de movilidad para las diferencias generalizadas [8].

En la Fig.2.10.4 podemos observar un análisis cualitativo a partir de la diferencia generalizada para una imagen de 120x120 píxeles, donde se puede identificar de forma visual una región que presentó mayor intensidad óptica para una cierta cantidad de imágenes.

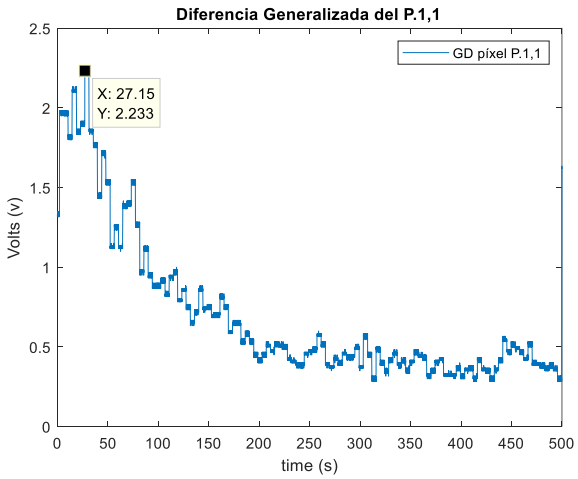


Fig.2.10.5: Diferencia de actividad de una muestra en el tiempo.

En la Fig.2.10.5 también se utiliza la diferencia generalizada, pero esta vez se analiza la magnitud de la muestra en el tiempo, de este modo se puede decir que para valores sobre los 1,5 volts la muestra presenta alta actividad, en el rango de los 0,5-1,5 volts la muestra tiene presenta actividad normal y para valores por debajo de los 0,5 volts la actividad microscópica es baja.

Capítulo 3. Desarrollo de sistema de procesamiento de imagen con FPGA para Speckle Dinámico.

3.1. Esquema y componentes del sistema de procesamiento de imagen con FPGA para Speckle dinámico.

Se modifica el sistema óptico portátil disponible en el Laboratorio de Optoelectrónica de la Universidad Católica de la Santísima Concepción (UCSC) reemplazando y agregando componentes como la cámara CMOS por la cámara CCD, un monitor para la visualización de la cámara, un FPGA para procesamiento de imagen y un osciloscopio para posterior análisis de datos. El montaje final puede ser visualizado en la Fig.3.1.1. y su forma esquemática en la Fig.3.1.2.

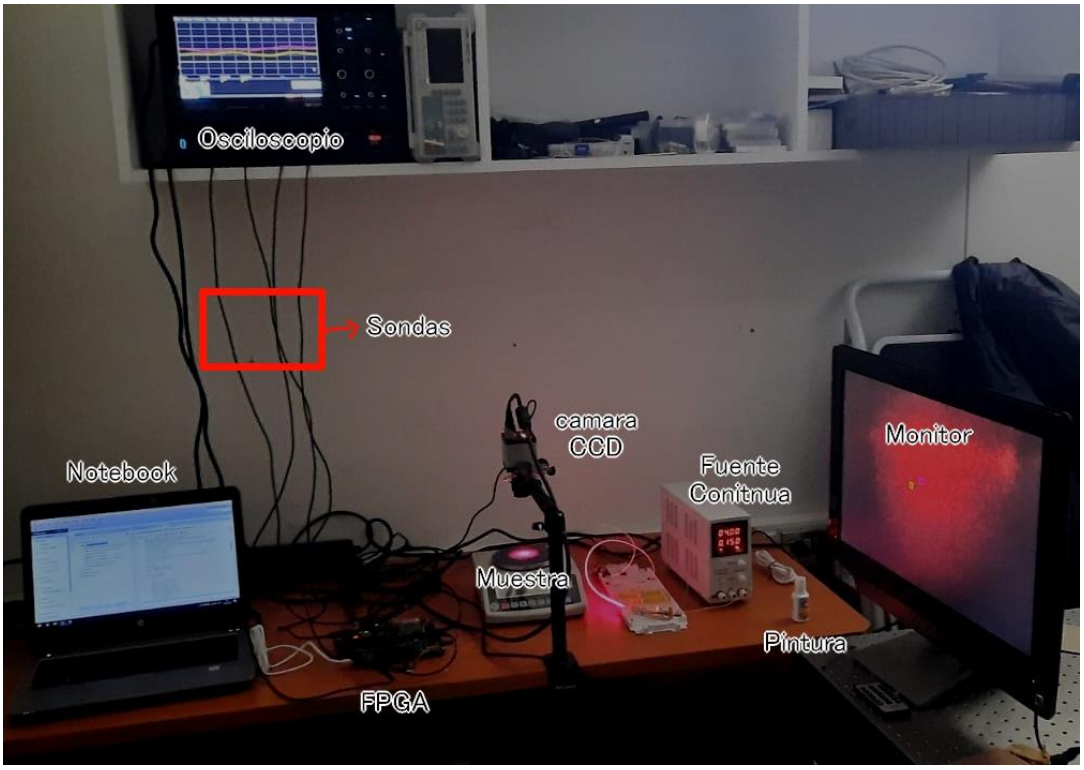


Fig.3.1.1: Montaje final para estudio de actividad microscópica con sistema óptico.

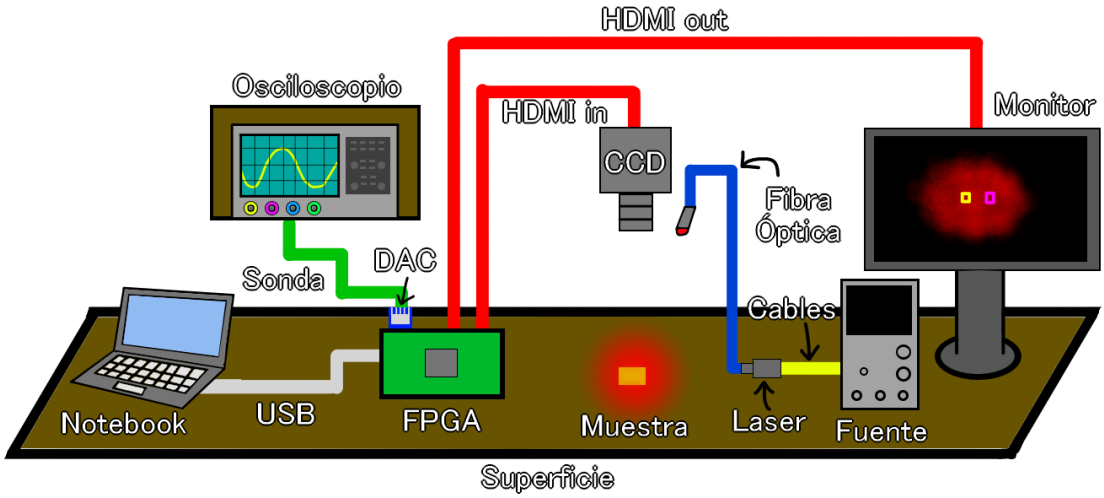


Fig.3.1.2: Montaje para procesamiento de imagen en tiempo real con FPGA.

Para el montaje se utilizó un Notebook modelo HP ProBook 640 G2, en él se programó mediante IDE Vivado y se analizaron los datos en MATLAB. El Notebook se conectado al FPGA mediante micro USB para cargar el archivo Bitsream generado.

La tarjeta electrónica utilizada FPGA corresponde a la Nexys Video A-7 que cumple la función de procesar los datos entregados por el cable HDMI y entregar resultados mediante los PMOD Header y la salida HDMI. El modelo lo podemos ver en la Fig.3.1.3, junto a su tabla de componentes.

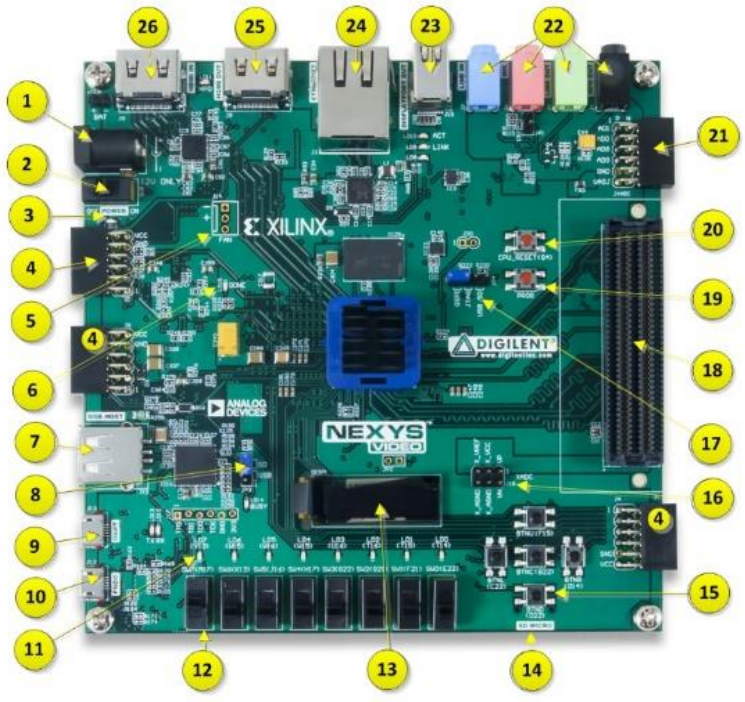
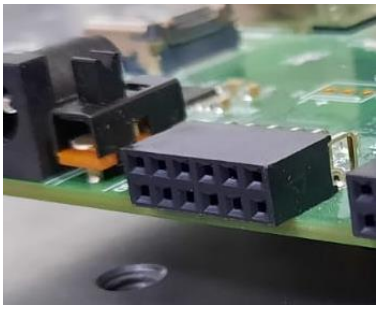


Fig.3.1.3: Estructura del FPGA Nexys Video A-7 [4].

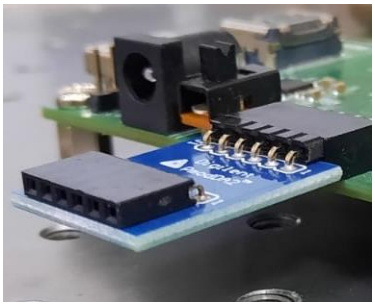
Tabla 1: Componentes del FPGA Nexys Video A-7 [4]**.**

Callout	Component Description	Callout	Component Description
1	Power Jack	14	MicroSD card slot (underneath)
2	Power Switch	15	Five pushbuttons
3	Power indicator LED	16	Miscellaneous XADC pins
4	Pmod Header(s)	17	Programming mode jumper
5	External fan control solder points	18	FMC header
6	FPGA programming done LED	19	FPGA configuration reset button
7	USB host connector	20	CPU reset button (for soft cores)
8	External configuration jumper (SD/USB)	21	Analog signal Pmod header (JXADC)
9	Micro USB (UART) connector	22	Microphone/Audio connector
10	Micro USB (JTAG) connector for USB programming	23	DisplayPort output connector
11	LEDs (8)	24	Ethernet connector
12	Slide Switches (8)	25	HDMI output connector
13	QLED display	26	HDMI input connector

En el modelo utilizado se destacan las entradas y salidas HDMI fundamentales para la conexión de la cámara y salida al monitor. La Nexys Video A-7 utiliza una alimentación externa de 12 V ±5% que ingresa por el Power Jack (1) y a su lado se encuentra el Power Switch para encender o apagar el FPGA. La programación es cargada mediante el Micro USB (JTAG) connector for USB programming señalado en (10). Por último, Para adquirir los datos del algoritmo aplicado en el procesamiento de imagen se utilizan los PMOD Header (4) [4].



(a) PMOD Header del Nexys Video

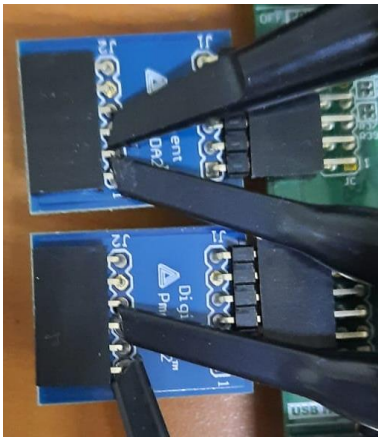


(b) PmodDA2 de Digilent acoplado

(c) PmodDA2 de Digilent.

Fig.3.1.4: Conexión del PMOD Header con un DAC.

Podemos ver en la Fig.3.1.4 (a) un PMOD Header, en (b) se le acopla un DAC que realiza la conversión digital a analógica que será enviada al osciloscopio. El modelo utilizado es el (c) Digilent Pmod DA2: Two 12-bit D/A Outputs (Diagrama en la Fig.3.1.6).



(a) Sondas en el DAC

(b) Osciloscopio WS3104Z

Fig.3.1.5: Medición de datos en Osciloscopio por medio de DACs.

Mediante la utilización de 4 sondas Fig.3.1.5 (a) se envía la señal del DAC hacia el osciloscopio (b). El osciloscopio utilizado corresponde al WaveSurfer 3104Z PROMO1.

Para realizar las mediciones con la sonda es necesario guiarse con el diagrama de la Fig.3.1.6 que proporciona información respecto a los pines que debe ir conectado en el FPGA y los pines de salida que entregan los datos. Las sondas están midiendo el voltaje que pasa por “P2: Data1” y “P3: Data2”.

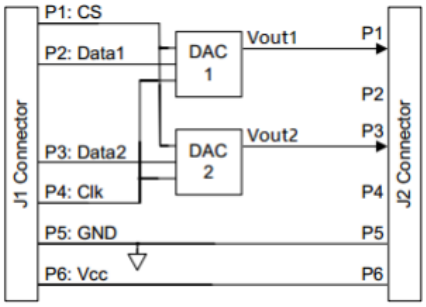
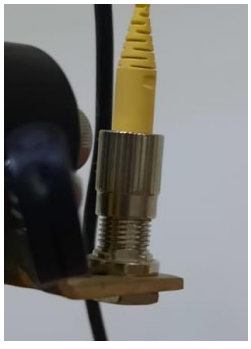


Fig.3.1.6: Diagrama del circuito PmodDA2 [41].

Para generar el Speckle dinámico se utiliza un sistema ya montado en el Laboratorio de Optoelectrónica que consta de un detector de fallas para fibras ópticas modificado con una longitud de onda de 650 nm. La alimentación es por medio de una fuente de poder ELE-TECH modelo HY3005B configurada para entregar 4 volts y 0,15 amperes. Esto lo podemos observar en la Fig.3.1.7 (a), la fibra óptica se conecta al detector de fallas y transporta la luz hacia un costado de la cámara.

(a) Alimentación del laser

(b) Fibra óptica entrada



(c) Fibra óptica salida

(d) Fibra óptica junto a la cámara

Fig.3.1.7: Transporte de la luz hacia el área de trabajo

En la Fig.3.1.8 podemos observar la cámara BAKU con sensor CCD utilizada para el proyecto, esta entrega una alta calidad y resolución de imagen, además, cuenta con una salida HDMI. Originalmente pertenecía a un microscopio de la misma marca el cual fue desmontado.

(a) Vista Frontal (b) Vista Trasera (c) Control

Fig.3.1.8: Cámara CCD de la marca BAKU

En la Fig.3.1.8. (b) se puede observar la salida HDMI de la cámara y junto a él una entrada para Tarjeta TF (TransFlash). En (c) tenemos el control utilizado para grabar, reproducir y configurar parámetros de la cámara.

La tarjeta de memoria ocupada es una microSD de la marca SanDISK (Fig.3.1.9) con 64 GB de almacenamiento. Su función es guardar la grabación de la muestra para su posterior reproducción y análisis.

Fig.3.1.9: Tarjeta microSD.

La salida HDMI de la cámara se conecta en la entrada HDMI del FPGA visible en la Fig.3.1.10. Los datos que ingresan son procesados y enviados por la salida HDMI para su visualización en un monitor.

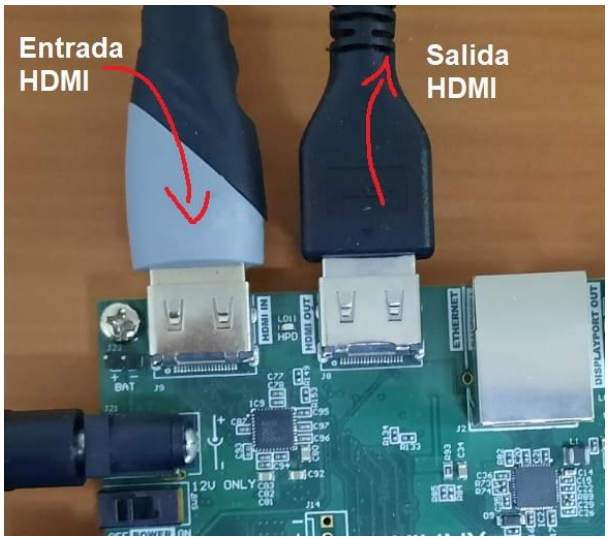


Fig.3.1.10: Conexión por protocolo HDMI del FPGA.

Como monitor se utilizó un computador “todo en uno” de la marca ASUS modelo AIO ET2210e 22" Pentium 2,6 GHz - HDD 1 TB – 4GB con una resolución nativa de 1920x1080 píxeles y una entrada HDMI. Fig.3.1.11.

Fig.3.1.11: Computador “todo en uno” de la marca ASUS.

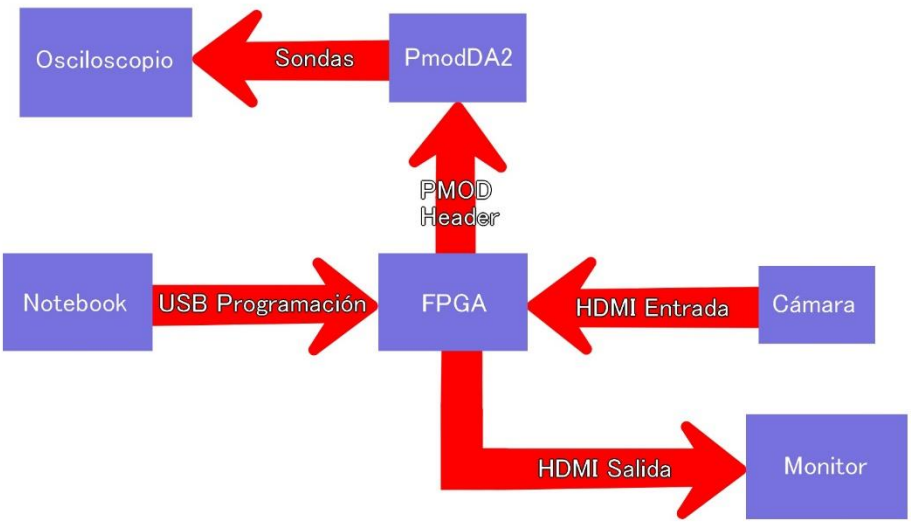


Fig.3.1.12: Diagrama simplificado de las conexiones con el FPGA.

El diagrama simplificado nos muestra las principales entradas y salidas que se realizan en el FPGA. Recibe la información por protocolo HDMI desde la cámara para procesarla y entregar dos salidas, una hacia el monitor por medio del HDMI de salida y otra hacia el osciloscopio a través del PMOD Header.

Para la programación del FPGA se utiliza IDE Vivado y se llega a un diseño en el que se distinguen cuatro etapas fundamentales:

- Entrada HDMI: Permite la adquisición de datos de la cámara CCD con protocolo HDMI
- Posición Sx y Sy del píxel: Detecta la posición del píxel para parámetros establecidos de 1920x1080 píxeles con 60 Hz
- Procesamiento de la imagen: Distintos ajustes a los datos recibidos para su posterior análisis.
- Salidas: Salida por los PMOD Header y protocolo HDMI.

3.2. Descripción de estructura y diseño implementado en el FPGA por medio de IDE Vivado.

Para poder leer los datos que entrega la cámara CCD mediante protocolo HDMI procesar los datos y entregarlos hacia un monitor con entrada HDMI, fue necesaria la implementación de un modelo compuesto por 25 módulos identificados en la Fig.3.2.1.

Fig.3.2.1: Módulos implementados para el sistema en Vivado.

Cada módulo contiene una parte del diseño digital y se utiliza para describir el comportamiento y funcionalidad del FPGA utilizando el lenguaje de programacion Verilog (.v) y VHDL (.vhd). Además, se pueden identificar 4 módulos principales que corresponden a (2) “Entrada”, (10) “SxSy”, (11) “Procesamiento” y (12) “Salida”. Un módulo top (1) “hdmi_top” se encarga de conectar los 4 módulos principales.

Los 4 módulos principales representan las etapas fundamentales del proceso y se encargan de agrupar los módulos con funciones específicas necesarios para el funcionamiento de cada etapa. El esquema fue exportado por medio de Vivado Disgin Suit y lo podemos ver en la Fig.3.2.2.

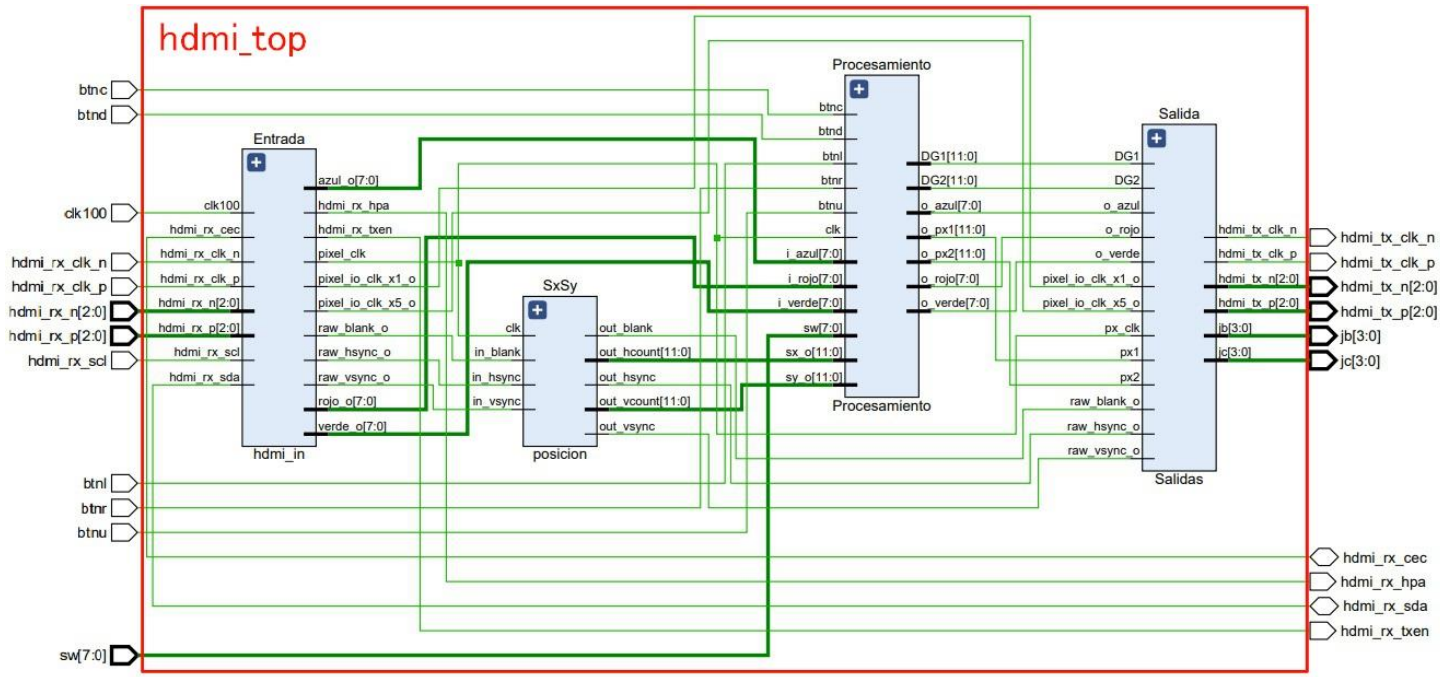


Fig.3.2.2: Conexión entre los módulos que caracterizan las etapas principales.

En la Fig.3.2.2. podemos observar los 4 módulos y la conexión que hay entre ellos que se realiza mediante el módulo “hdmi_top” (Recuadro rojo que contiene a los 4 módulos). Cada uno de estos módulos contiene en su interior más módulos que ayudan a cumplir con el objetivo de cada etapa. A continuación, se hace una breve explicación de su funcionamiento y objetivo.

El (1) “hdmi_top”, tiene como función realizar la conexión entre los módulos “Entrada”, “SxSy”, “Procesamiento” y “Salida”. También llama a las entradas y salidas fundamentales del FPGA que conectan a los periféricos con los módulos de forma coherente.

```
module hdmi_top(
        input wire clk100,
    // Entrada HDMI
        inout wire hdmi_rx_cec,
        output wire hdmi_rx_hpa,
        input wire hdmi_rx_scl,
        inout wire hdmi_rx_sda,
        input wire hdmi_rx_clk_n,
        input wire hdmi_rx_clk_p,
        input wire [2:0]hdmi_rx_n,
        input wire [2:0]hdmi_rx_p,
    // Salida HDMI
        output wire hdmi_rx_txen,
        output wire hdmi_tx_clk_n,
        output wire hdmi_tx_clk_p,
        output wire [2:0]hdmi_tx_p,
        output wire [2:0]hdmi_tx_n,
    // Perifericos
        input wire btnc, btnl, btnr,btnu,btnd,
        input wire [7:0] sw,
        output wire [3:0]jc,
        output wire [3:0]jb
    );
```

Fig.3.2.3: entradas y salidas del hdmi_top.

En la Fig.3.2.3 podemos observar que las entradas del “hdmi_top” corresponden a “clk100” que es el clock del sistema de 100 MHz, los “hdmi_rx”, los botones y los switches, en cuanto a las salidas se tienen los “hdmi_tx”, el “hdmi_rx_txen” y los PDMO (jc y jb especificamente).

```
##HDMI out
set_property -dict { PACKAGE_PIN AA4   IOSTANDARD LVCMOS33 } [get_ports { hdmi_tx_cec }]; #IO_L11N_T1_SRCC_34 Sch=hdmi_tx_cec
set_property -dict { PACKAGE_PIN U1    IOSTANDARD TMDS_33  } [get_ports { hdmi_tx_clk_n }]; #IO_L1N_T0_34 Sch=hdmi_tx_clk_n
set_property -dict { PACKAGE_PIN T1    IOSTANDARD TMDS_33  } [get_ports { hdmi_tx_clk_p }]; #IO_L1P_T0_34 Sch=hdmi_tx_clk_p
set_property -dict { PACKAGE_PIN AB13  IOSTANDARD LVCMOS25 } [get_ports { hdmi_tx_hpd }]; #IO_L3N_T0_DQS_13 Sch=hdmi_tx_hpd
set_property -dict { PACKAGE_PIN U3    IOSTANDARD LVCMOS33 } [get_ports { hdmi_tx_rscl }]; #IO_L6P_T0_34 Sch=hdmi_tx_rscl
set_property -dict { PACKAGE_PIN V3    IOSTANDARD LVCMOS33 } [get_ports { hdmi_tx_rsda }]; #IO_L6N_T0_VREF_34 Sch=hdmi_tx_rsda
set_property -dict { PACKAGE_PIN Y1    IOSTANDARD TMDS_33  } [get_ports { hdmi_tx_n[0] }]; #IO_L5N_T0_34 Sch=hdmi_tx_n[0]
set_property -dict { PACKAGE_PIN W1    IOSTANDARD TMDS_33  } [get_ports { hdmi_tx_p[0] }]; #IO_L5P_T0_34 Sch=hdmi_tx_p[0]
set_property -dict { PACKAGE_PIN AB1   IOSTANDARD TMDS_33  } [get_ports { hdmi_tx_n[1] }]; #IO_L7N_T1_34 Sch=hdmi_tx_n[1]
set_property -dict { PACKAGE_PIN AA1   IOSTANDARD TMDS_33  } [get_ports { hdmi_tx_p[1] }]; #IO_L7P_T1_34 Sch=hdmi_tx_p[1]
set_property -dict { PACKAGE_PIN AB2   IOSTANDARD TMDS_33  } [get_ports { hdmi_tx_n[2] }]; #IO_L8N_T1_34 Sch=hdmi_tx_n[2]
set_property -dict { PACKAGE_PIN AB3   IOSTANDARD TMDS_33  } [get_ports { hdmi_tx_p[2] }]; #IO_L8P_T1_34 Sch=hdmi_tx_p[2]
```

Fig.3.2.4: Archivos .XDC para ejemplo con salida HMDI.

Estas entradas y salidas van asociadas al chip como tal y sus nombres derivan del “Constraint” (Archivo .xdc) del FPGA Artix-7 (este archivo se puede descargar desde Digilent). Como ejemplo podemos observar la Fig.3.2.4 donde se hacen las restricciones para las salidas HDMI, aquí se asocian los pines del FPGA con los pines de la placa junto a los estándares de propiedades eléctricas admitidas, además de asignar un nombre al final de la línea que se utilizará para llamarlos en el apartado de programación (Fig.3.2.3).

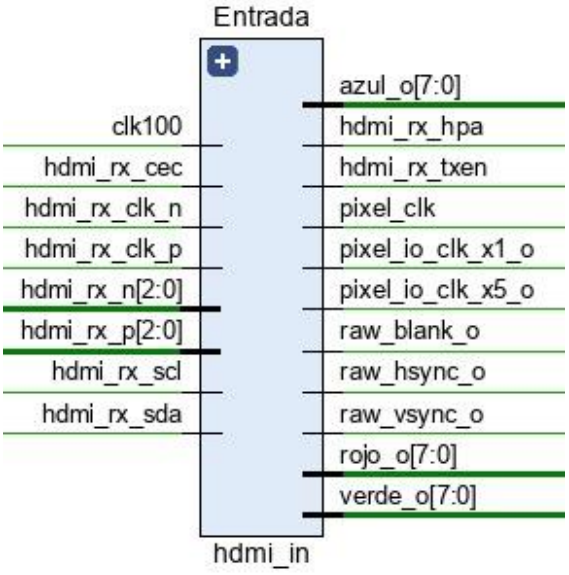


Fig.3.2.5: Entradas y salidas del módulo Entrada: hdmi_in.

En la Fig.3.2.5 se observa el módulo (2) “Entrada: hdmi_in” que corresponde a la etapa de recepción de los datos, como entradas se encuentran el clock del sistema (“clk100”) y los “hdmi_rx” (señal de entrada en serie proveniente de los TMDS del HDMI), por la parte de las salidas se tienen los colores rojo, verde y azul como valores de 8 bits en paralelo, los relojes “pixel_clk”, “pixel_io_clkx1_o” y “pixel_io_clkx5_o” que representan diferentes señales de reloj impulsadas por BUFIO (elemento lógico de los FPGA, utilizado para gestionar señales de reloj) y son utilizados para la sincronización y temporizado en el procesamiento de señales de video. Además, encontramos las salidas “raw_blank_o”, “raw_hsync_o” y “raw_vsync_o” que son señales de sincronización en la secuencia de datos de video en bruto (**2.9. Video Timings** “fase en blanco”) donde “raw_blank_o” indica la señal de borramiento, “raw_hsync_o” es la señal de sincronización horizontal y “raw_vsync_o” es la señal de sincronización vertical. Dentro de este módulo encontraremos cuatro IBUFDS y un módulo denominado “i_hdmi_input”.

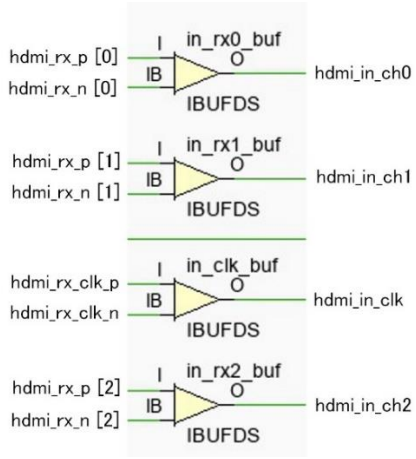


Fig.3.2.6: IBUFDS para convertir las señales diferenciales en señales de un solo extremo.

Los IBUFDS son elementos lógicos que se encargan de tomar una señal diferencial, en este caso los “hdmi_rx_p [2:0]”, “hdmi_rx_n [2:0]”, “hdmi_rx_clk_p”, “hdmi_rx_clk_n” y los convierte a cada uno en una señal de un solo extremo (Fig.3.2.6 como ejemplo). Estas señales ingresan al módulo (3) “i_hdmi_input” con el nombre “hdmi_in_ch0”, “hdmi_in_ch1”, “hdmi_in_ch2” y “hdmi_in_clk”.

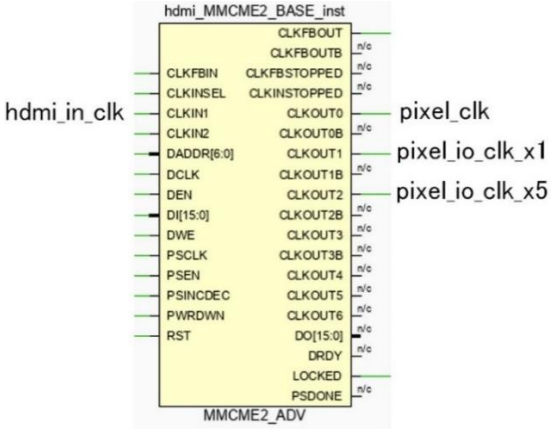


Fig.3.2.7: MMCME generador de relojes del sistema.

Dentro del módulo (3) se utiliza un MMCME2 (Mixed-Mode Clock Manager) el cual toma la entrada “hdmi_in_clk” y genera las salidas “pixeles_clk”, “pixel_io_clk_x1” y “pixel_io_clk_x5” (Fig.3.2.7) que son relojes utilizados para el sincronismo de las distintas etapas. Cada “hdmi_in_ch0/ch1/ch2” transporta la información de los colores Rojo, Verde y Azul de forma serial y binaria e ingresan respectivamente a los módulos “ch0”, “ch1” y “ch2” que podemos observar en la Fig.3.2.8 (encerrado en un cuadrado rojo y que corresponden a la posición (4), (8) y (9) respectivamente).

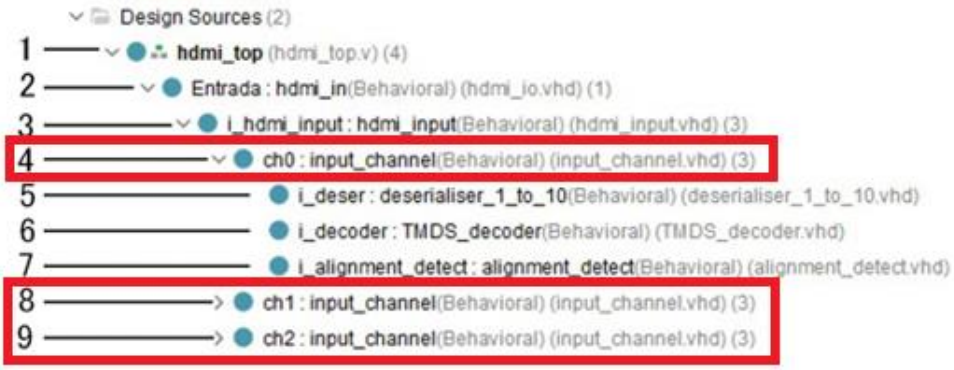


Fig.3.2.8: Módulos input_channel y sus elementos.

Los módulos “ch0”, “ch1” y “ch2” no solo reciben como entrada los datos de color, sino que también reciben cuatro señales de reloj, tres corresponden a los generados por el MMCME2 y el otro corresponde al reloj de 100 MHZ del FPGA. Además, cuenta internamente con un módulo de deserialización “i_deser”, un módulo de decodificación “i_decoder” y un módulo de detección de alineación “i_alignment_detect”.

El “i_deser” se utiliza para deserializar la señal de entrada serial de 10 bits y convertirla en una señal paralela también de 10 bits llamada “data_o”. Para la deserialización se utiliza un elemento lógico programable llamado “ISERDESE2” que forma parte de los dispositivos FPGA.

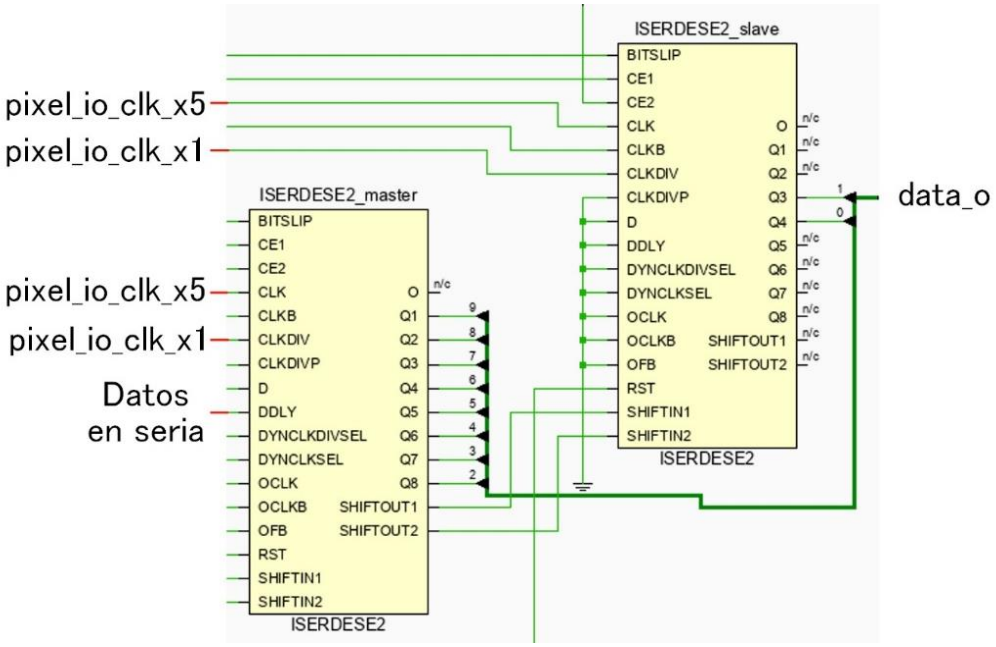


Fig.3.2.9: ISERDESE, deserialización de datos de 10 bits en serie.

Los ISERDESE2 tiene un máximo de 8 salidas “Q1-Q8” por lo que es necesario utilizar una configuración “master” y “slave”. El reloj “CLK” del deserializador utilizado para muestrear la señal serial de entrada y controlar las operaciones internas corresponde al “pixel_io_clk_x5”, por otro lado, el reloj “CLKDIV” para ajustar la tasa de datos de la señal serial corresponde al “pixel_io_clk_x1”. Las señales de datos provenientes de los canales “hdmi_in_ch0/ch1/ch2” son ingresados por la entrada “DDLY” del ISERDESE y se retrasa utilizando un bloque “IDELAYE2” (este bloque es controlado por el módulo “alignment_detect”) con el propósito de ajustar el tiempo de llegada de la señal para la deserialización.

La señal “data_o” de 10 bits paralelos que se generó en el ISERDESE es enviada al módulo “i_decoder” quien recibe la señal y la decodifica en diferentes tipos de datos siendo los más relevantes “ctl”, “data” e “invalid_symbol”. El “ctl” transporta la información de control relacionada con la configuración y el dispositivo de visualización, mientras que “data” corresponde a la señal de 8 bits paralelos que transporta los datos reales que se muestran en la pantalla, por último, “invalid_symbol” se utiliza para indicar si el símbolo decodificado es válido o no. Este último junto al “pixel_clk” son enviados como entradas al módulo “alignment_detect” que genera las salidas “delay_count”, “delay_ce” y “bitslip” con el objetivo de supervisar y controlar estas señales para asegurar una correcta alineación de datos deserializados.

Luego de toda esta etapa de adquisición de datos viene el módulo (10) “SxSy” de la Fig.3.2.10 que se utiliza para identificar y generar las posiciones “out_hcount” (“sx” horizontal) y “out_vcount” (“sy” vertical) basadas en las señales de sincronización de entrada.

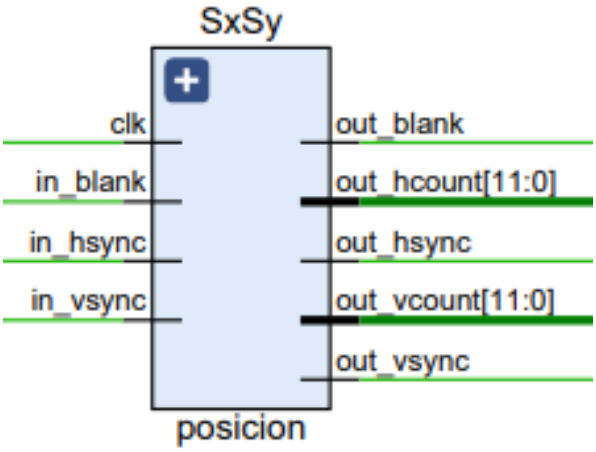


Fig.3.2.10: Módulos SxSy para detectar la posición del píxel.

“out_count” y “out_vcount” se calculan en función a las señales de entrada “hcount” y “vcount” respectivamente, ambos son contadores internos del módulo para los píxeles horizontales (“hcount”) y píxeles verticales (“vcount”).

En el proceso si “in_blank” cambia de 0 a 1 entonces se incrementa el contador vertical “vcount” en 1 con el fin de contar las líneas verticales. Si “in_vsync” cambia de 1 a 0 entonces “vcount” se restablece en 0 y el valor anterior de “vcount” es almacenado en un registro interno llamado “v_size” para mantener un seguimiento del número de líneas por cuadrado. Si “in_blank” es 0 entonces “hcount” incrementa en 1. Cuando “in_blank” es 1 y “hcount” es distinto de 0 entonces el valor de “hcount” se guarda en “h_size” para mantener un seguimiento del número de pixeles horizontales en la línea de video activa, cuando el valor ya está guardado “hcount” se hace 0. Los valores “hcount” y “vcount” son asignados a las salidas “out_hcount” y “out_vcount” respectivamente y permiten realizar un seguimiento de la posición de los píxeles. Por otro lado, las entradas “in_blank”, “in_hsync” e “in_vsync” son puenteadas a las salidas “out_blank”, “out_hsync” y “out_vsync” para su utilización en otros módulos. Las salidas “out_hcount” y “out_vcount” se conectan a las entradas “sx_o” y “sy_o” respectivamente del módulo “Procesamiento” que se puede observar en la Fig.3.2.11.

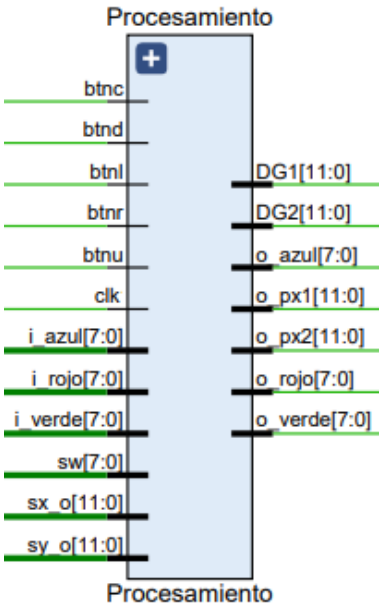


Fig.3.2.11: Módulos Procesamiento para aplicar algoritmo en tiempo real.

Como se puede observar este módulo tiene varias entradas de control denominadas “btnc”, “btnd”, “btnl”, “btnr” y “btnu” que significan botón central, botón abajo, botón izquierdo, botón derecho y botón arriba respectivamente, estos son botones del Nexys Video y pueden ser controlados manualmente. También se observa la entrada de 8 switches que al igual que los botones cumplen la

función de ejecutar acciones sobre el módulo. Respecto a las entradas de datos se pueden distinguir las señales "i_rojo", "i_verde" e "i_azul" que representan los colores Rojo, Verde y Azul con una profundidad de 8 bits, las señales "sx_o" y "sy_o" que se utilizan para determinar la posición del píxel y la señal "clk" que opera a una frecuencia de 148,5 MHz y se encarga de coordinar todos los procesos del módulo.

Internamente el módulo tiene dos registros denominados “pixel_1” y “pixel_2” de 8 bits que se encargan de almacenar el valor de un píxel seleccionado por el usuario mediante la botonería. Este píxel es restringido por las coordenadas “sx_o” y “sy_o” las que además tienen una variable llamada “incremento” que les suma o resta valores dependiendo de que botón presionamos para desplazarse por el monitor y seleccionar el píxel deseado. Esto permite que el píxel que se está almacenando varíe desplazándose por los distintos pixeles capturados por la cámara hasta llegar al píxel del área que se quiere analizar. Para una visualización más clara de la región en la que nos encontramos se realizó un arreglo que genera un cuadrado amarillo para el “pixel_1” y un cuadrado morado para el “pixel_2” Fig.3.2.12.

Fig.3.2.12: Monitor con dos cuadrados de referencia para estudio de píxel.

Para el procesamiento del píxel en tiempo real se utiliza un bloque “case”. Estos bloques seleccionan una ruta a partir de una variable de decisión que permite el correcto funcionamiento de nuestro sistema de procesamiento. La variable de decisión corresponde a un “$index$” de 3 bits que permite un máximo de 7 destinos, cada uno con una determinada acción.

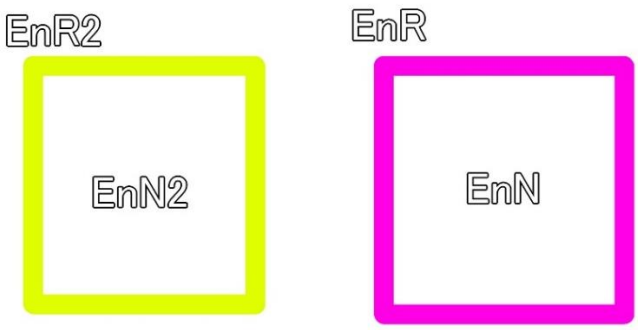


Fig.3.2.13: Cuadrados generados para determinar posición del píxel.

En la explicación del bloque case se habla de unas variables denominadas “EnR”, “EnR2”, “EnN” y “EnN2” (Fig.3.2.13). “EnR” corresponde a los momentos en que los píxeles se encuentran en el borde del cuadrado morado y “EnN” a los momentos cuando los píxeles se encuentran dentro de él. La misma lógica se utiliza para el otro cuadrado donde “EnR2” corresponde al borde del cuadrado amarillo y “EnN2” a la zona interior.

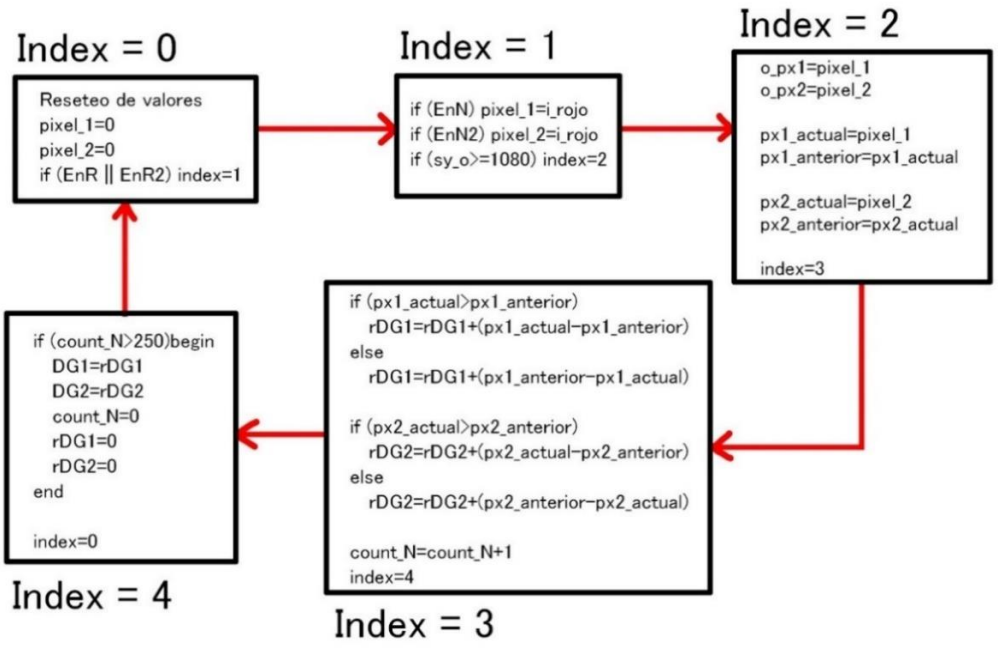


Fig.3.2.14: Diagrama para el bloque case del módulo “¨Procesamiento”.

En la Fig.3.2.14 observamos un diagrama de bloques que representa al bloque “case” que actúa en cada flanco de subida del reloj “clk” que corresponde al píxel clock de 148,5 MHz. El bloque inicia en “$index = 0$” donde se ajustan los parámetros iniciales de “pixel_1” y “pixel_2” en 0.

Cuando nos encontremos en el borde del cuadrado amarillo “EnR” o en el cuadrado morado “EnR2” nos desplazaremos al “$index = 1$” donde mediante un “if” se guardan los píxeles cuando nos

encontremos dentro de los cuadrados (“EnN” o “EnN2”), en “pixel_1” se guarda el píxel rojo que ingresa y se encuentra dentro del cuadrado morado, mientras que en “pixel_2” se guardara el píxel rojo que ingresa y se encuentra dentro del cuadrado amarillo.

Si el valor de “sy_o” supera los 1080 pixeles nos moveremos al “$index = 2$”. En este bloque los registros “o_px1” y “o_px2” (que son salidas del módulo “Procesamiento”) guardan el valor del píxel final del área interna de los cuadrados morada y amarilla respectivamente. Además, se crean 4 “memorias” que guardan el mismo píxel en distintos periodos de tiempo, siendo “px1_actual” el que almacena el “pixel_1” que va ingresando y “px1_anterior” el que almacena el valor anterior del mismo pixel. El proceso se repite para “px2_actual” y “px2_anterior” pero con “pixel_2”. Cada vez que se guardan los píxeles el “index” cambia a “$index = 3$” donde se calcula el algoritmo de diferencia generalizada (GD).

Para obtener el valor absoluto del “GD” se crea un “if” que evalúa cuál píxel (“px_actual” y “px_anterior”) tiene mayor intensidad y posteriormente realizar la resta entre ellos. El resultado es guardado en un registro denominado “rDG1” para el píxel del área morada y “rDG2” para el píxel del área amarilla. Cuando se guarda un resultado en “rDG1” y “rDG2” un registro denominado “count_N” aumenta una unidad para luego pasar al “$index = 4$”.

En este “index” se evalúa el valor de “count_N”, si este no es mayor que 250 entonces se retorna a “$index = 0$” y se repite todo el proceso de modo que se va realizando una sumatoria en “rDG1” y “rDG2” hasta obtener 250 muestras, una vez hayan ocurrido 250 fotogramas (muestras) el “count_N” habrá llegado al valor 250, superado este valor se guarda la sumatoria de la diferencia generalizada para 250 muestras en un registro denominado “DG1” y “DG2” en “$index = 4$” a su vez se reestablecen los valores iniciales de “rDG1”, “rDG2” y “count_N” para la sumatoria de las siguiente muestras. Los registros “DG1” y “DG2” son salidas del módulo “Procesamiento” y representan la diferencia generalizada para 250 muestras de dos píxeles en distintos puntos.

Como resultado se tiene el módulo “Procesamiento” con las salidas “o_px1”, “o_px2”, “DG1”, “DG2”, “o_rojo”, “o_verde” y “o_azul”, estos últimos tres son puenteados directamente con la entrada “i_rojo”, “i_verde” e “i_azul” que transportan la información de los 1920x1080 píxeles para su posterior visualización en un monitor.

El módulo denominado “Salida” que podemos observar en la Fig.3.2.15 se encarga de recibir las señales de los módulos anteriores y entregarlos en puertos de salida, en este caso hacia dos DACs y un HDMI transmisor.

Fig.3.2.15: Módulo “Salida” para la última etapa.

El módulo “dac02” (Fig.3.2.15) recibe las dos señales de datos “o_px1” y “o_px2” por sus entradas denominadas “Din1” y “Din2”. El proceso internamente realiza un bloque “case” con tres etapas:

- Etapa 0: Carga los datos de entrada en los registros “B” y “B2” para luego avanzar a la “Etapa 1”
- Etapa 1: Se generan las señales analógicas “PinDAC1” y “PinDAC2” basadas en los datos de entrada. Se establece “syncDAC” en 0 para indicar que se está generando una señal analógica y se inicia un contador “CN” que cuando realiza 16 ciclos de reloj avanza a la “Etapa 2”.
- Etapa 2: En esta etapa “syncDAC” se establece en 1 para señalar que el DAC no está generando una señal analógica en ese momento con la finalidad de controlar y coordinar el funcionamiento del DAC, esta etapa dura 3 ciclos de reloj.

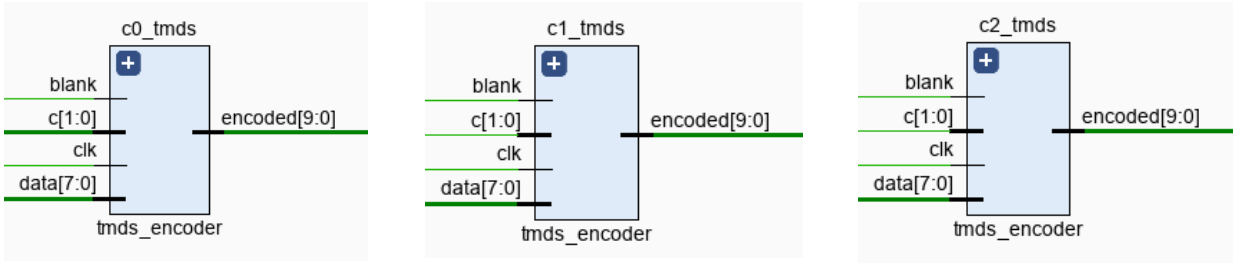
Para el módulo “dac03” se repite esto mismo pero los datos de entrada corresponden a la diferencia generalizada “DG1” y “DG2”.

Fig.3.2.16: Módulo “salida: hdmi_out” para visualización en monitor por medio de HDMI.

En la Fig.3.2.16 se observa el módulo “SalidaHDMI” encargado de enrutar las señales que entran y salen del módulo “GeneradorHDMI”, además, se encarga de convertir la señal de un solo extremo en señales diferenciales por medio de los “OBUFDS”.

El módulo “GeneradorHDMI” tiene en su interior las etapas específicas “c0_tmds”, “c1_tmds”, “c2_tmds” para la codificación y los “ser_ch0”, “ser_ch1”, “ser_ch2” y “ser_clk” para la serialización.

Esta etapa de transmisión de datos HDMI es similar a la de recepción de datos HDMI, solo que realiza el objetivo contrario. En la “Entrada” era deserialización, decodificación y al final un detector de alineación, en cambio, para el “GeneradorHDMI” las etapas son codificación y serialización en ese orden y quitando la detección de alineación ya que esta etapa es para que el receptor de datos detecte que la información que llega lo haga de la forma correcta.



(a) “c0_tmds” (b) “c1_tmds” (c) “c2_tmds”

Fig.3.2.17: Módulos para la codificación de las señales RGB.

Los módulos “c0_tmds”, “c1_tmds”, “c2_tmds” de la Fig.3.2.17 ocupan el módulo base “tmds_encoder” que se encarga de realizar la codificación de las señales que ingresan con los datos Rojo, Verde y Azul que corresponden a “o_rojo”, “o_verde” y “o_azul” provenientes del módulo “Procesamiento” y traen la información de los pixeles, cada una de estas señales ingresa por la entrada “data” de los módulos siendo “c0_tmds” para el color rojo, “c1_tmds” el color verde y “c2_tmds” el color azul. La entrada “c” es un control para la codificación que recibe información de “out_hsync” y “out_vsync”, la entrada “blank” recibe información de “out_blanks” para identificar las etapas en blanco, estas tres señales provienen del módulo “SxSy”, para el sincronismo de las etapas se utiliza como entrada el reloj “clk” que corresponde al “pixel_clk” de 148,5 MHz. Los tres módulos entregan solo una salida denominada “encoded” que es una señal de 10 bits en paralelo que debe ser convertida a una señal de 10 bits en serie, para ello, se utiliza el elemento lógico programable “OSERDESE2” que forma parte de los dispositivos FPGA.

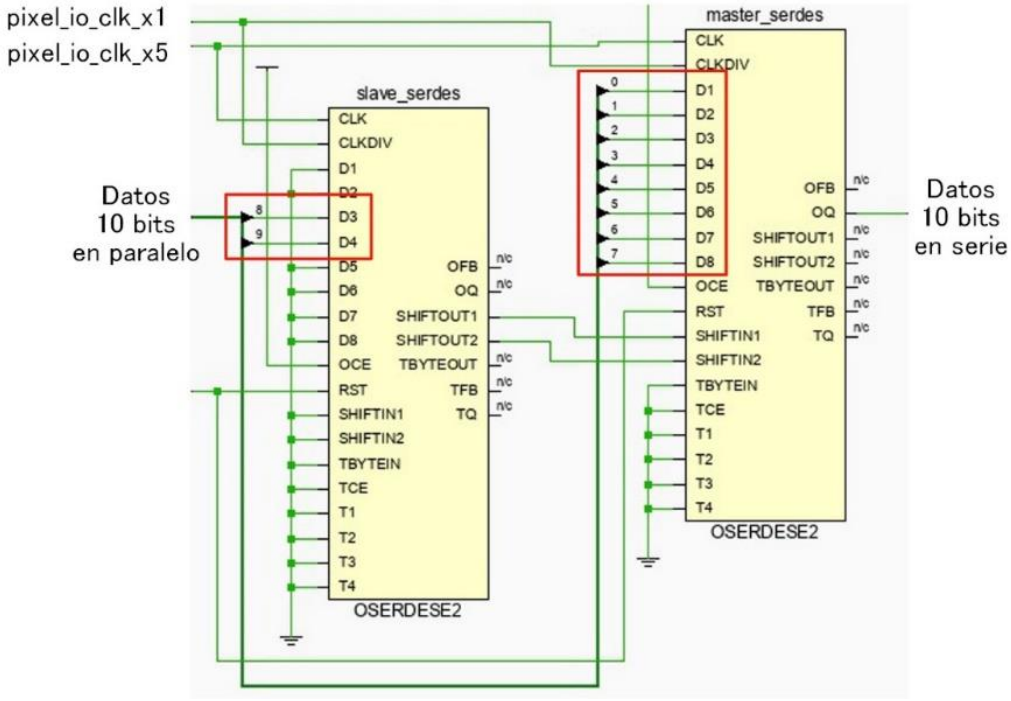


Fig.3.2.18: OSERDESE, serialización de datos de 10 bits paralelos.

Al igual que en el “ISERDESE2” el “OSERDESE2” utiliza en su “CLK” el reloj “pixel_io_clk_x5” y en el “CLKDIV” el “´pixel_io_clk_x1” para sincronizar correctamente los datos que entran y salen, también, podemos observar la configuración “master” y “slave”. El “master” puede recibir 8 bits de la señal codificada mientras que el “slave” se encarga de los 2 bits faltantes, ambos se coordinan utilizando los “SHIFTOUT/IN 1” y “SHIFTOUT/IN 2” para entregar como resultado una señal en serie de 10 bits. Este proceso se repite para cada canal y se obtienen las salidas “tmds_out_ch0”, “tmds_out_ch1”, “tmds_out_ch2” y “tmds_out_clk”. Hay que recordar que estas cuatro señales son salidas del módulo “GeneradorHDMI” que se encuentra dentro del módulo “SalidaHDMI”.

“SalidaHDMI” toma las señales “tmds_out_ch0”, “tmds_out_ch1”, “tmds_out_ch2” y “tmds_out_clk” para conectarlas cada una a un OBUFDS los cuales son elementos lógicos que se encargan de tomar una señal de un solo extremo y la convierten en una señal diferencial, en este caso las salidas “hdmi_tx_p [2:0]”, “hdmi_tx_n [2:0]”, “hdmi_tx_clk_p” y “hdmi_tx_clk_n” (Fig.3.2.19).

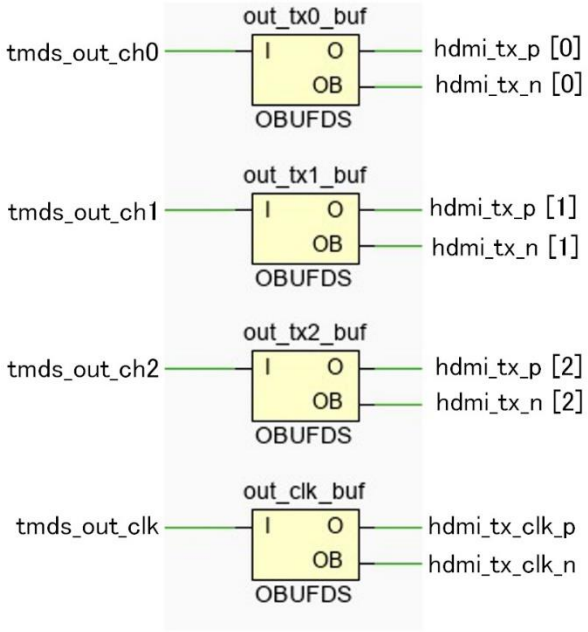


Fig.3.2.19: OBUFDS para convertir las señales de un solo extremo en señales diferenciales.

Las señales diferenciales salen de estos módulos hacia el “hdmi_top” quien toma estas señales y las conecta directamente con los pines de la placa hacia el periférico de salida HDMI. Lo mismo ocurre con las señales “jb” y “jc” producidas por los DACs, las cuales son enrutadas hacia los PMOD de la Nexys Video A-7.

3.3. Especificaciones del diseño y determinación de parámetros.

Para el proyecto es importante señalar que solo se trabajará con el canal rojo de los datos adquiridos por la cámara ya que la luz laser utilizado tiene una onda longitudinal de 650 nm, por lo que agregar en el análisis los canales verde y azul no generarían un aporte.

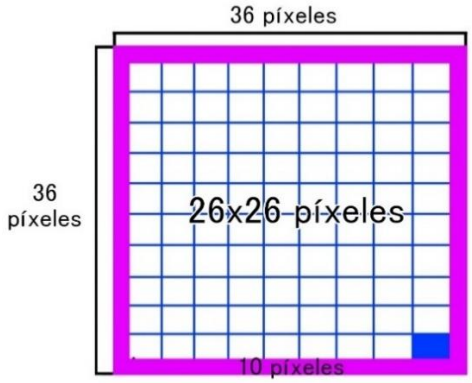
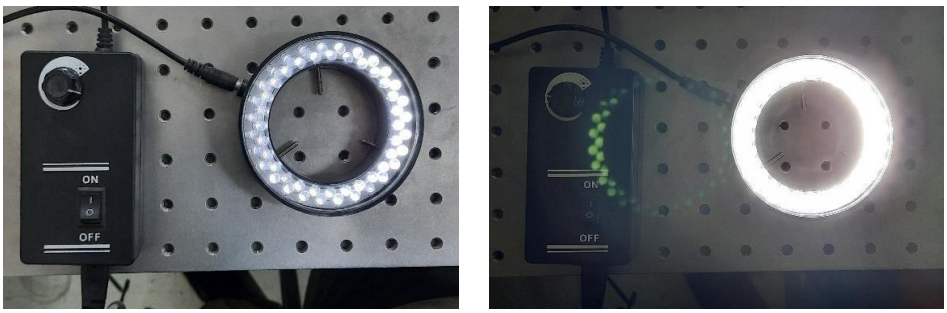


Fig.3.3.1: Dimensiones de los cuadrados diseñados y píxel estudiado.

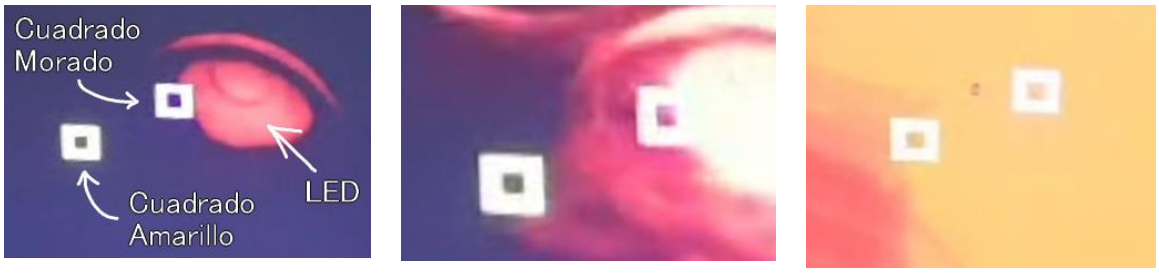
Las dimensiones de los cuadrados generados por el FPGA son de 36x36 píxeles para la parte exterior, 26x26 píxeles para la parte interior y 10 píxeles como borde. Podemos ver el cuadrado interior como una matriz de 26x26 donde el píxel estudiado corresponde al último valor de la matriz señalado como un cuadrado pintado en azul. Las dimensiones del cuadrado fueron diseñadas con el propósito de tener una fácil visualización del píxel seleccionado en el monitor. Los colores amarillo y morado fueron escogidos pues son los colores típicos de las señales en los osciloscopios, de modo que la señal morada y amarilla del osciloscopio equivalen al cuadrado morado y amarillo respectivamente. Para determinar si la información que llega al osciloscopio está siendo monitoreada con los cuadrados se utilizó un aro de luz Led conectado a un potenciómetro.



(a) Aro de luz en intensidad mínima (b) Aro de luz en intensidad máxima

Fig.3.3.2: Aro de luz con potenciómetro.

La Fig.3.3.2 corresponde al aro de luz utilizado para evaluar tanto la posición como la intensidad máxima del píxel. Utilizando el potenciómetro se fue aumentando la intensidad del led de forma progresiva hasta llegar al máximo, posterior a esto, se fue disminuyendo la intensidad hasta volver a su estado inicial.



(a) Led estado inicial (b) Led mediana intensidad (c) Led máxima intensidad

Fig.3.3.3: Distintos estados del Led para dos posiciones de píxel.

Las fotos de la Fig.3.3.3 fueron tomadas directamente con el teléfono apuntando al monitor por eso no se logra distinguir con claridad los colores y dimensiones de los cuadrados. Sin embargo, se logra distinguir que uno (amarillo) está más alejado del Led que el otro (morado). Las señales con la intensidad del píxel salen por un DAC hacia el osciloscopio y son almacenados en una memoria externa para su análisis en MATLAB.

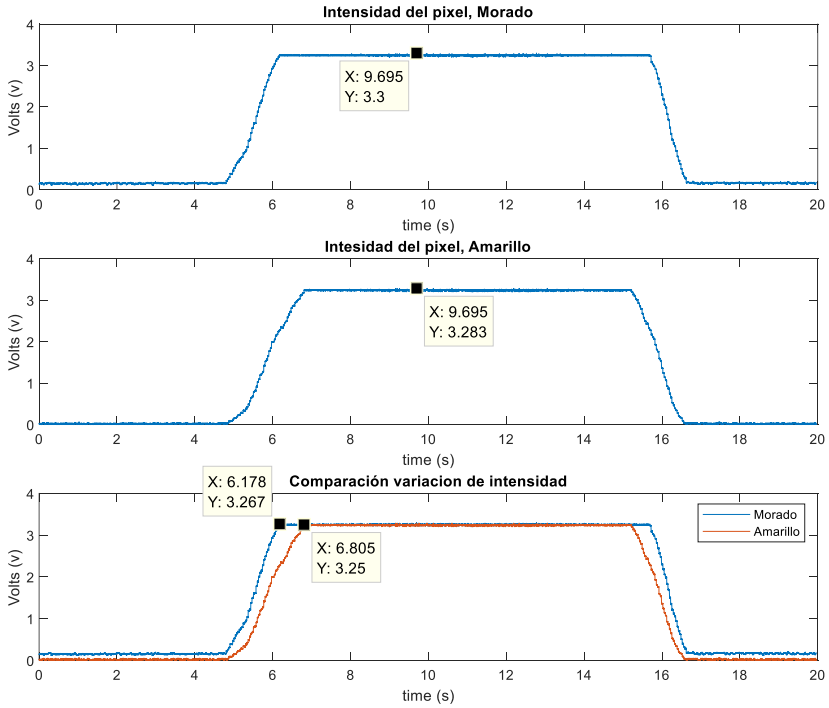


Fig.3.3.4: Representación de la intensidad del píxel como voltaje en tiempo.

Podemos observar respecto a la Fig.3.3.4 que el píxel ubicado más cerca del led (cuadrado morado) no parte con intensidad cero debido a que el led en su estado inicial ya presenta una pequeña intensidad de luz. Sin embargo, el otro píxel (cuadrado amarillo) inicia con un valor más cercano a cero debido a que se encuentra más alejado. Cuando se comienza a incrementar la intensidad del led, el píxel del cuadrado morado llega al umbral 0,627 segundos antes que el píxel del cuadrado amarillo. Se concluye que los píxeles monitoreados con los cuadrados son los correctos.

También a partir de la misma figura podemos determinar cuál es el punto de saturación del píxel en su valor de voltaje. Un píxel tiene sus valores definidos en 8 bits para cada color por lo que se distinguen 256 tonalidades de rojo, por lo tanto, aproximadamente 3,3 volts sería el equivale a la tonalidad roja 256.

La cámara CCD presenta un parámetro denominado “Automatic White Balance” (AWB) para cada canal y se utiliza para equilibrar los niveles de blanco en los colores. Dejar este parámetro en automático puede provocar la saturación en algunos pixeles y generar pérdida en cuanto a la información de la actividad del píxel. La cámara cuenta con un menú que permite dejar este parámetro en forma manual, para determinar en qué valor dejar el parámetro AWB se ubicó el cuadrado morado en el centro del área generada por el láser (punto con mayor intensidad) y se fue variando el AWB desde 255 (máximo) hasta 0 (mínimo).

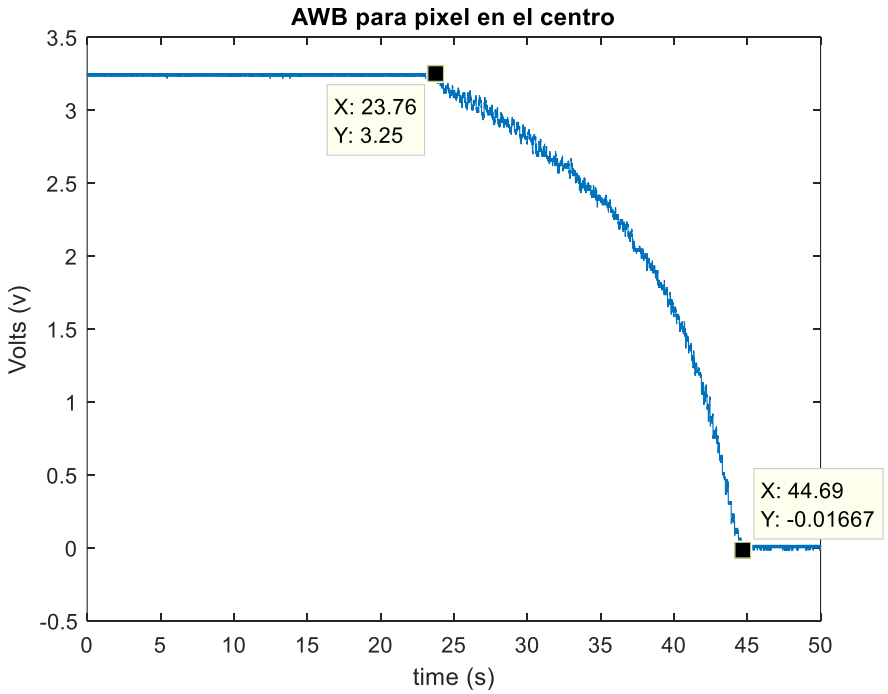


Fig.3.3.5: Relación voltaje en el tiempo para distintos AWB.

Observando el resultado de la Fig.3.3.5 en los primero 23 segundos se encuentran los valores de AWB desde 255 hasta 105, aquí existe una saturación del píxel constante de aproximadamente 3,3 volts, pasados los 23 segundos la saturación comienza a disminuir llegando a aproximadamente 0 volts en el segundo 45 donde el AWB ya es 0. Por lo tanto, el valor de AWB óptimo para trabajar se encuentra dentro de los 105 y 0 AWB. Para este trabajo se decidió utilizar el valor que se encontraba aproximadamente en el 50%, es decir un AWG de 50.

(a) Distancia muestra y lente (b) Apertura del diafragma del lente [42].

Fig.3.3.6: Configuración de la cámara para el experimento.

La distancia entre el lente y la muestra la podemos ver en la Fig.3.3.6 (a) que corresponde a 200 mm y es la distancia mínima en la que puede trabajar el lente. La apertura del diafragma utilizado es F22 y es la apertura mínima del lente utilizado, esta apertura permite detectar poca luz, pero aumenta la profundidad de campo.

Respecto a la Tesis desarrollada por Daniel Nusdel **"Desarrollo de sistema óptico portable para análisis de movimiento microscópico"** [8] se determina que la cámara debe trabajar con un ángulo de 90° respecto a la horizontal para preservar la forma gaussiana del haz de luz generador por la fibra óptica.

3.4. Comparación de variación del píxel para área con y sin actividad microscópica.

Una vez ya se tienen los parámetros definidos se deposita una muestra de pintura (corrector) sobre una placa metálica y comparamos dos pixeles, uno ubicado sobre la muestra y otro en la placa metálica.

(a) Pintura de corrector (b) Placa metálica con pintura

Fig.3.4.1: Materiales utilizados para la experiencia.

En la Fig.3.4.1. (b) se señalan las áreas comparadas con un cuadrado morado y amarillo haciendo referencia a los cuadrados generados con el FPGA y que dan como resultado el Speckle dinámico de la Fig.3.4.2.

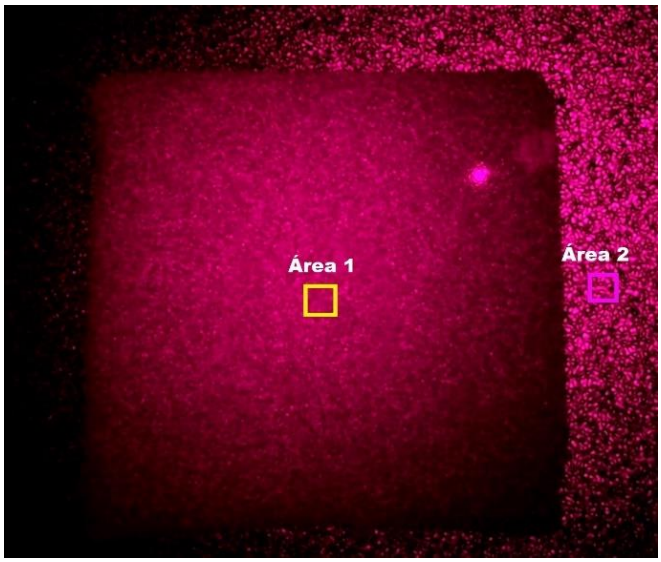


Fig.3.4.2: Speckle dinámico para muestra de pintura.

Los datos obtenidos en el osciloscopio a partir de la muestra fueron almacenados en una memoria externa para su análisis en MATLAB obteniendo los siguientes resultados.

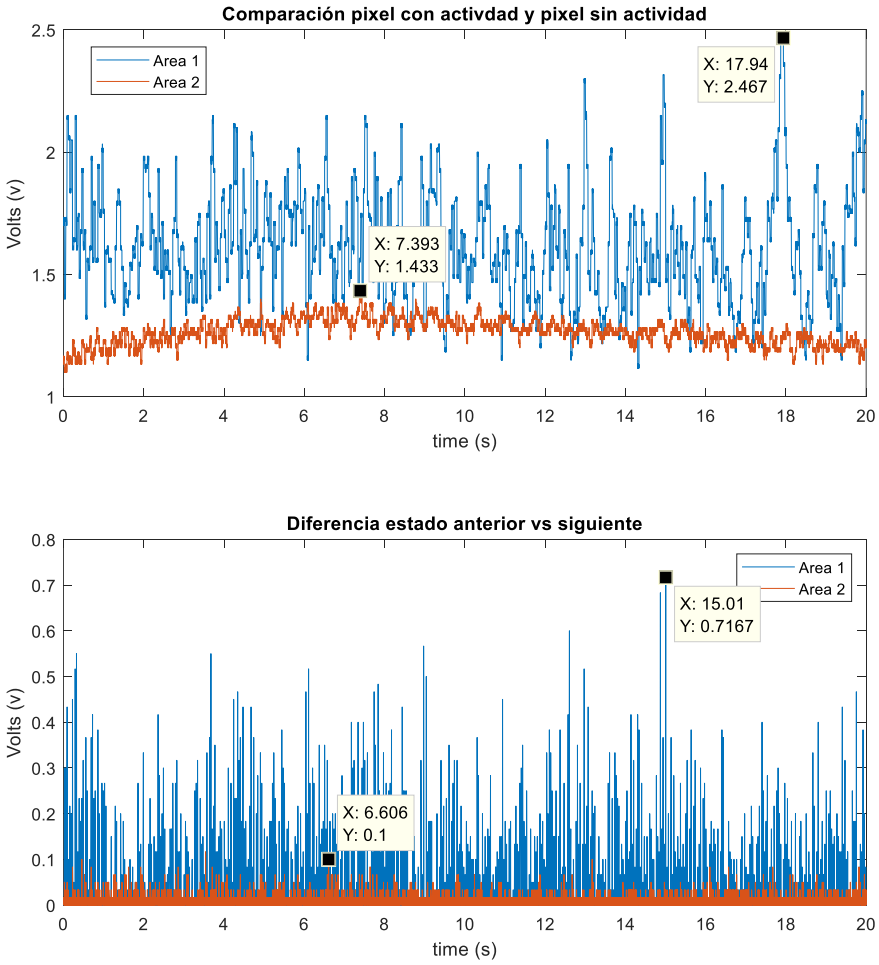


Fig.4.3.3: Comportamiento del píxel para distintas áreas.

En el primer gráfico de la Fig.3.4.3 el área 1 que corresponde al píxel sobre la muestra presenta una mayor actividad en comparación con el área 2. Además, se demuestra que el área con actividad no llega al umbral (3,3 volts) durante un periodo de 20 segundos siendo su valor máximo 2,467 volts, esto significa que los parámetros establecidos con anterioridad nos permiten trabajar con valores dentro de un rango aceptable.

En el segundo gráfico se tiene la diferencia del píxel para distintos instantes de tiempo con la finalidad de comparar su valor actual con el anterior, esto permite determinar que tanto varía la intensidad del píxel cuando se encuentra sobre una superficie con actividad microscópica. Podemos observar que cuando hay actividad la diferencia puede llegar a ser de 0,7 volts y cuando no hay actividad la diferencia máxima es de 0,1 volts.

Capítulo 4. Evaluación de micro actividad utilizando un FPGA.

En este capítulo se evalúa la diferencia generalizada de una muestra de pintura durante 500 segundos y sus mediciones gravimétricas. La pintura tiene un exceso de masa húmeda que al evaporarse genera un movimiento molecular detectado por el láser y capturado por la cámara como un patrón de Speckle dinámico. Además, esta evaporación produce una disminución en la masa de la muestra que es analizado mediante el estudio gravimétrico.

Artículos como "**Speckle time evolution characterization by the co-occurrence matrix analysis**" de Ricardo Arizaga, Marcelo Trivi, Héctor Rabal [43] ya han realizado estudios de mediciones ópticas en función de las mediciones gravimétricas obteniendo una alta correlación.

El algoritmo utilizado es una adaptación que permite evaluar la diferencia generalizada para un píxel y queda definido por la ecuación (2).

$$GD_{P(x,y)} = \sum_{k=1}^{N} |I_k(x,y) - I_{k-1}(x,y)| \qquad (2)$$

Donde $GD_{P(x,y)}$ es la diferencia generalizada para el píxel en las coordenadas (x, y), N es la cantidad de imágenes secuenciales utilizadas e I_k es la intensidad del píxel para la k-ésima imagen. En esta ecuación se realiza la diferencia en valor absoluto entre la intensidad del píxel actual y la intensidad del píxel anterior.

La cámara utilizada tiene una tasa de refresco de 60 Hz que se traduce en 60 imágenes por segundo enviadas al FPGA, como la muestra de pintura es estudiada durante 500 segundos se tiene un total de 30000 imágenes. El algoritmo GD implementado utiliza 250 imágenes de forma secuencial, de modo que si dividimos el total de 30000 imágenes por la cantidad de imágenes que utiliza el algoritmo se tendrá un total de 120 resultados de diferencia generalizada cada uno con una duración de 4,16 segundos.

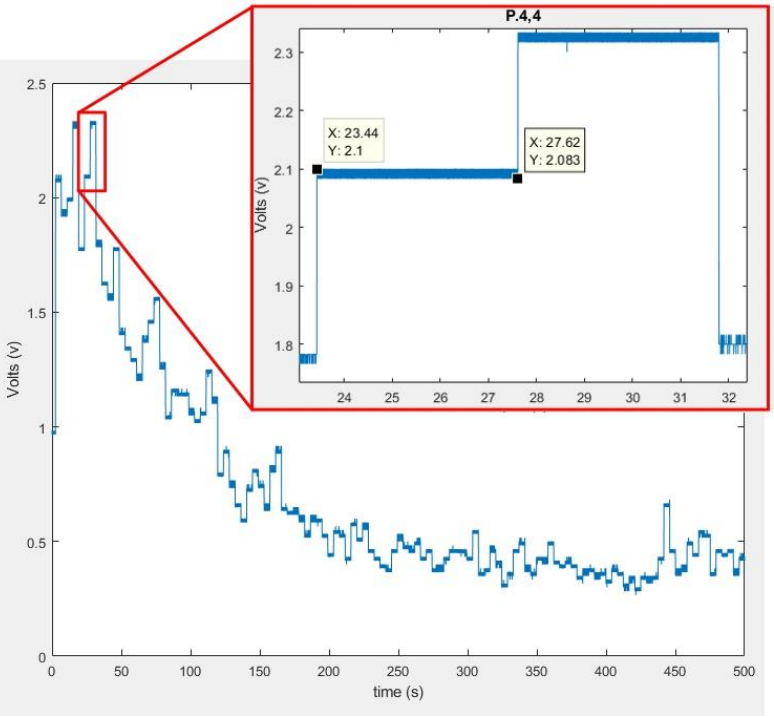


Fig.4.1: Tiempo requerido para 250 muestras.

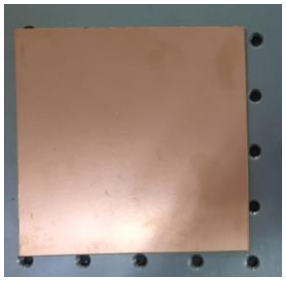
Sin embargo, en la Fig.4.1. podemos observar que la muestra tiene una duración de 4,18 segundos, es decir, cada 4,18 segundos tendremos un nuevo valor de la muestra. La forma teórica para determinar esta duración es mediante la ecuación (3):

$$T_m = \frac{N+1}{fps}\ (segundos) \qquad (3)$$

Donde T_m es el tiempo que dura la muestra en segundos, N es la cantidad de muestras secuenciales utilizadas para la GD y fps son los *"frames per second"*. Si nos fijamos en el numerador el valor N está siendo sumado en una unidad, esto se debe a que el FPGA necesita un *"frame"* para guardar el valor GD y enviarlo hacia el osciloscopio (Esto lo podemos ver en el capítulo 3, cuando se habla de la estructura y diseño implementado en el FPGA).

4.1. Estudio gravimétrico para una muestra de pintura.

Para iniciar con las mediciones es necesario ubicar la placa metálica sobre la balanza con el objetivo de establecer el peso de la placa metálica como valor inicial cero. La balanza utilizada tiene una precisión de 0,005 gramos.



(a) Placa metálica

(b) Balanza con placa metálica

Fig.4.1.1: Etapa inicial del experimento.

Una vez que la balanza ya está ajustada se deposita la muestra de pintura sobre la placa metálica y se anota su peso inicial a la vez que se inicia la grabación por medio de la cámara CCD. Se realiza un monitoreo del peso apuntando los momentos en que existe una variación. La muestra se deja secar durante 500 segundos.

Tabla 2: Variaciones del peso para la muestra.

Tiempo (Segundos)	Peso (Gramos)
0	0,415
25	0,410
65	0,405
100	0,400
129	0,395
170	0,390
211	0,385
249	0,380
294	0,375
338	0,370
386	0,365
437	0,360
490	0,355
546	0,350

La muestra utilizada tiene un peso inicial de 0,415 gramos que disminuye un 15,66% en un periodo de 546 segundos. Se detectaron 14 instantes donde la muestra tuvo una variación de peso.

4.2. Diferencia generalizada (DG) para píxeles centrales de una muestra.

La región estudiada corresponde a un cuadrado en el centro de la muestra de 4x4 píxeles con un total de 16 píxeles. La denominación del píxel estudiado será como los de una matriz siendo el primer píxel el de la esquina superior izquierda (P.1,1) y el ultimo píxel el de la esquina inferior derecha (P.4,4), guiarse con la Fig.4.1.1.

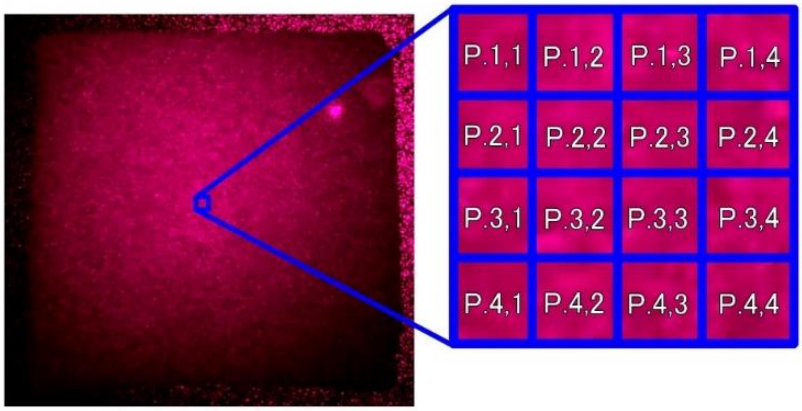


Fig.4.2.1: Muestra para un cuadrado de 4x4 píxeles.

El perfil de movilidad microscópica para la pintura en el píxel “P.1,1” lo podemos observar en la Fig.4.2.2. obtenido mediante MATLAB (el código MATLAB se encuentra en el Anexo A).

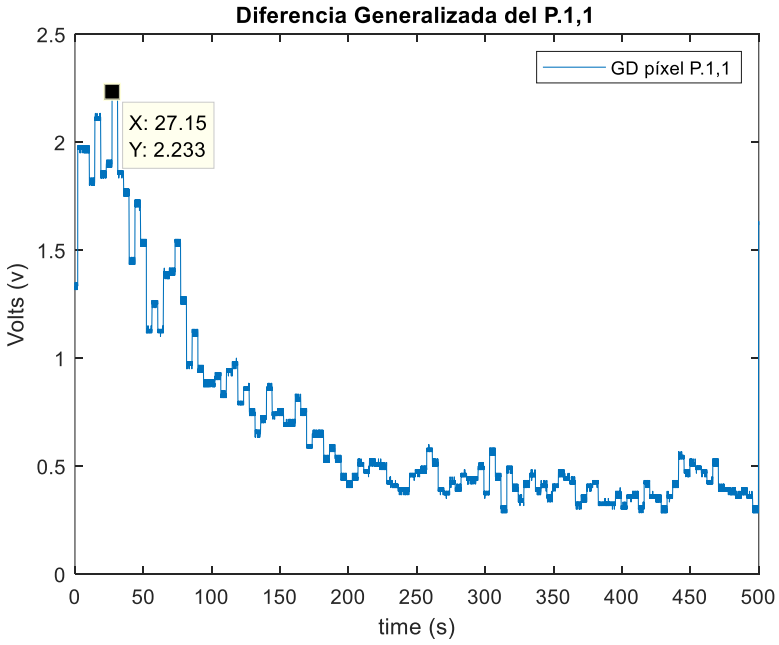


Fig.4.2.2: Diferencia generalizada (GD) del P.1,1.

Se puede observar un movimiento congruente con respecto a la actividad microscópica en el tiempo donde la mayor actividad microscópica se concentra en los primeros segundos y a medida que la pintura se seca va disminuye, siendo la actividad máxima registrada a los 27 segundos.

Para comparar esta curva con el estudio gravimétrico es necesario ajustarla, de modo que cada vez que se detecte una variación en el peso se calcule el promedio de la diferencia generalizada hasta ese punto (el código en MATLAB se encuentra en el Anexo B). Por ejemplo, si observamos la Tabla 2 la primera variación de peso ocurre a los 25 segundos, por lo tanto, calcularemos el promedio de la diferencia generalizada para los datos que se encuentren entre 0 y 25 segundos (como los datos son hasta 500 segundos no se considera la variación de peso en el segundo 546) obteniendo el gráfico de la Fig.4.2.3.

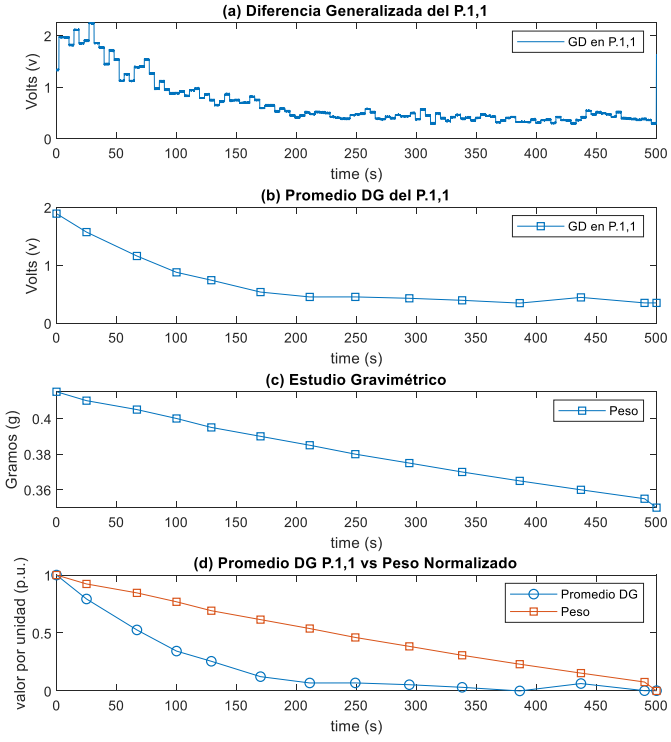


Fig.4.2.3: Promedio GD del P.1,1 vs Estudio Gravimétrico.

La curva (b) de la Fig.4.2.3 corresponde al promedio de la diferencia generalizada del P.1,1 para cada segundo donde existió una variación de peso, (c) corresponde al estudio gravimétrico y (d) es la comparación normalizada de ambas curvas utilizando la ecuación (4).

$$N_i = \frac{(X_i - x_{min})}{X_{max} - X_{min}} \qquad (4)$$

Donde X_i es el valor de la variable original que se desea normalizar, X_{min} es el valor mínimo y X_{max} es el valor máximo en un rango i de datos, N_i es el valor normalizado de X_i. La comparación normalizada nos permite observar que a medida que disminuye el peso también disminuye la actividad. Sin embargo, el peso tiene un comportamiento lineal con pendiente negativa mientras que el promedio GD disminuye de forma exponencial.

Mediante MATLAB obtenemos la tabla 3 que corresponde al promedio GD y el peso de la muestra. Además, estos valores fueron llevados a Excel donde se realizó un gráfico para determinar la correlación entre el peso y el promedio GD.

Tabla 3: Peso vs Promedio GD del P.1,1.

Peso (gramos)	Promedio GD (volts)
0,415	1,890
0,41	1,572
0,405	1,159
0,4	0,877
0,395	0,741
0,39	0,537
0,385	0,452
0,38	0,453
0,375	0,429
0,37	0,394
0,365	0,346
0,36	0,445
0,355	0,350

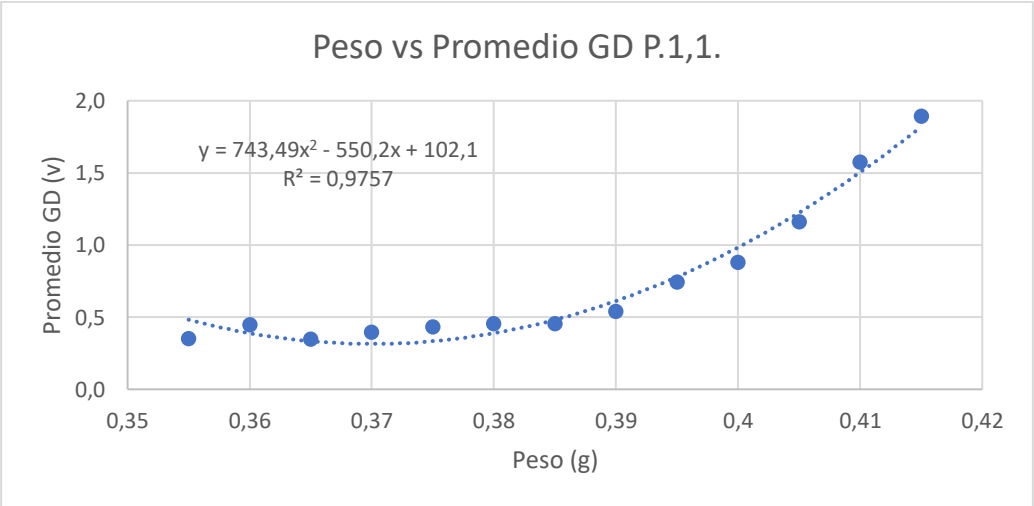


Fig.4.2.4: Correlación del promedio GD del P.1,1 y el peso.

Respecto a la Fig.4.2.4 se observa que la relación entre el promedio de actividad microscópica del P.1,1 y el peso no es lineal, sino más bien se ajusta a un modelo de regresión cuadrática presentando una correlación de los datos del 97,57% ($R^2 = 0{,}9757$). Se obtiene además un Error Cuadrático Medio (ECM) de 0,0057 que al ser tan bajo significa que la predicción del modelo cuadrático se ajusta muy bien a los valores observados. Por otro lado, el Error Porcentual Absoluto Medio (MAPE) es del 12%, lo que significa que la regresión cuadrática presenta un error promedio del 12% entre los datos reales y los predichos.

A continuación, se realiza una tabla con todos los valores de R^2 para una aproximación cuadrática entre el promedio de actividad de la muestra para cada píxel y el peso. Las tablas "Peso vs Promedio GD" y las figuras de correlación del Peso y el Promedio GD para cada píxel se encuentran en el Anexo C.

Tabla 4: R^2 para la aproximación cuadrática de cada píxel.

Pixel (x,y)	R^2
P.1,1	0,9757
P.1,2	0,9789
P.1,3	0,9832
P.1,4	0,9799
P.2,1	0,9787
P.2,2	0,9787
P.2,3	0,9811
P.2,4	0,9797
P.3,1	0,9806
P.3,2	0,9781
P.3,3	0,9834
P.3,4	0,9861
P.4,1	0,9790
P.4,2	0,9801
P.4,3	0,9846
P.4,4	0,9868

Todos los pixeles ubicados en el cuadrado estudiado tuvieron una alta correlación para una aproximación cuadrática entre el promedio GD y el peso, siendo el píxel P.4,4 el que obtuvo la mayor correlación con un 98,68% ($R^2 = 0{,}9868$), un EMC igual a 0,0033 y un MAPE del 9,42% (en el Anexo D se puede observar una tabla para el EMC y el MAPE de cada píxel).

Bajo los parámetros establecidos con anterioridad, la luz generada por la fibra óptica presenta un haz gaussiano como el de la Fig.4.2.5.

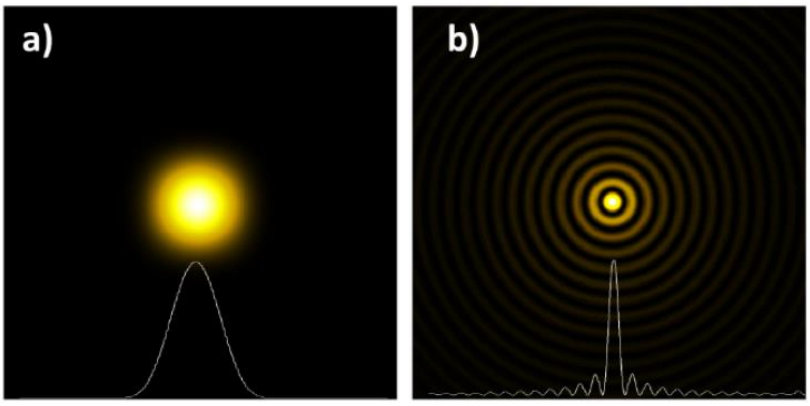


Fig.4.2.5: comparación (a) Haz gaussiano y (b) Haz Bessel [44].

Los resultados indican que a medida que el píxel se acerca al centro de la muestra nos acercamos al centro del haz gaussiano ya que se van presentando mejores valores de R^2. Podemos observar en la Fig.4.2.6 una representación que nos muestra como aumenta el R^2 a medida que el píxel se aproxima al centro.

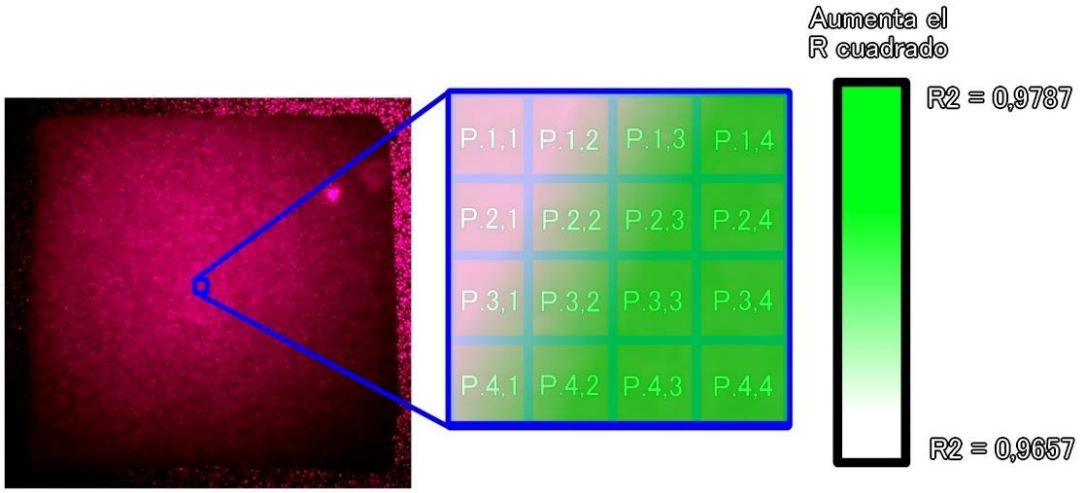


Fig.4.2.6: Relación de R^2 y proximidad del centro.

Capítulo 5. Conclusiones

La implementación de lógica programable en el FPGA Nexys video A-7 que permite la conexión de una cámara CCD y un monitor por medio de protocolo HDMI es factible gracias a las funciones específicas de los FPGA, entre las que se encuentran los “ISERDESE2” y “OSERDESE2” para la deserialización y serialización. El “MMCME2” para el diseño de distintos relojes que sincronizan distintas etapas. Los “IBUFDS” y “OBUFDS” para convertir señales diferenciales en señales de un solo extremo o convertirlas de señales de un solo extremo a señales diferenciales, entre otros.

Para un píxel que cuenta con 256 tonalidades se llegó a la conclusión de que su valor equivalente en Volts al medir la intensidad del píxel en un osciloscopio era aproximadamente de 3,3 volts, este valor corresponde al máximo y es un umbral que no se debe superar para el estudio de actividad microscópica. También se evaluó el balance de blancos (AWB) de la cámara analizando los valores que provocaban una saturación en el píxel central (zona con mayor intensidad lumínica), donde se llegó a la conclusión de que con un AWB Rojo = 105 el píxel recién dejaba de estar saturado, sin embargo, trabajar con valores cercanos a la saturación puede significar perdida de información por lo que se decidió utilizar un AWB Rojo = 50 que corresponde al 50% de la curva.

Aplicar la diferencia generalizada para el procesamiento de imagen de Speckle en tiempo real si es posible. Se logró aplicar el algoritmo GD mediante un FPGA que seguía el ritmo de la tasa de refrescos propia de la cámara, esto significa que la FPGA puede aplicar este algoritmo a una velocidad de 0,016 segundos permitiendo visualizar el comportamiento de la actividad de la muestra en el osciloscopio al mismo tiempo que se está realizando el experimento.

Se logró determinar que utilizando una aproximación cuadrática que caracteriza el promedio de la diferencia generalizada con el peso de la muestra durante 500 segundos se obtienen valores coherentes con respecto a la teoría del secado de pintura, llegando a valores de determinación del 98,68% ($R^2 = 0{,}9868$) en el mejor de los casos y un MAPE del 9,42%.

Finalmente, se concluye que el sistema montado con cámara CCD captura con éxito los patrones de Speckle de una muestra, además, estos patrones pueden ser analizados en tiempo real mediante un FPGA aplicando un algoritmo de diferencia generalizada (GD).

5.1 Trabajo Futuro.

- Implementación de varias Block RAM en el diseño, con la finalidad de poder guardar la información de varios pixeles de una imagen en distintos instantes, para así aplicar un algoritmo de análisis de actividad microscópica en tiempo real en una región más grande. Actualmente el análisis realizado implicada el estudio píxel por píxel.
- Modificación del diseño mostrado en el monitor de modo que se pueda visualizar un mapa de movilidad generado a partir de algoritmos aplicados en la muestra.
- Comparación de algoritmos para el procesamiento de patrones de Speckle Dinámico implementados en FPGA para el análisis en tiempo real.
- Comparación entre cámara con sensor CMOS y CCD en un sistema de análisis de patrón de Speckle dinámico implementado en FPGA. En el proyecto solo se utilizó la cámara con sensor CCD gracias a la conexión por protocolo HDMI, sin embargo, los sensores CMOS y CCD tienen características que los diferencian y los hacen mejores o peores para trabajar en ciertas condiciones.

Bibliografía

[1] C. Meza Sparza, «Coloquio de Ingeniería se basó en «Speckle Láser Dinámico": técnica óptica para la biología y la industria,» UCSC, 06 10 2021. [En línea]. Available: https://www.ucsc.cl/noticias/coloquio-de-ingenieria-se-baso-en-speckle-laser-dinamico-tecnica-optica-para-la-biologia-y-la-industria/#:~:text=El%20speckle%20din%C3%A1mico%20es%20t%C3%ADpico,distinci%C3%B3n%20de%20espec%C3%ADmenes%20vegetales%2C%20etc.. [Último acceso: 19 05 2023].

[2] E. E. Ramírez Miquet, L. Martí López y O. R. Contreras Alarcón, «Dos configuraciones diferentes para la descripción temporal de actividad de Escherichia coli mediante speckle dinámico,» *Revista mexicana de física*, vol. 57, nº 5, pp. 441-445, 2011.

[3] F. A. Curotto Molina, «Design, implementation and characterization of a radio frecuency interference digital adaptive filter using a field-programmable gate array,» Universidad de Chile, Santiago, 2019.

[4] Digilent, «Digilent,» 1 09 2020. [En línea]. Available: https://digilent.com/reference/_media/reference/programmable-logic/nexys-video/nexys-video_rm.pdf. [Último acceso: 09 05 2023].

[5] H. Sendra, S. Murialdo y L. Passoni, «Dynamic laser speckle to detect motile bacterial response of Pseudomonas aeruginosa,» *Journal of Physics: Conference Series*, vol. 90, nº 1, p. 012064, 2007.

[6] I. M. Inciarte Vergara, L. Martí López, E. E. Ramírez Miquet, E. K. Hernández, Á. Viloria y F. Otárola Luna, «Procesado de patrones de speckle mediante dos métodos para medir la desecación de semillas de uchuva (Physalis peruviana L.),» *Interciencia*, vol. 37, nº 9, pp. 644-650, 2012.

[7] J. Cariñe, R. Guzmán y F. Torres-Ruiz, «Algorithm for dynamic Speckle pattern processing,» *Optics and Lasers in Engineering*, vol. 82, pp. 56-61, 2016.

[8] D. A. Nusdel Neira, «Desarrollo de sistema óptimo portable para análisis de movimiento microscópico,» Concepción, 2023.

[9] J. I. Amalvy, C. A. Lasquibar, R. Arizaga, H. Rabal y M. Trivi, «Application of dynamic speckle interferometry to the drying of coatings,» *Progress in organic coatings*, vol. 42, nº 1, pp. 89-99, 2001.

[10] D. Sierra-Sosa, M. Tebaldi, E. Grumel, H. Rabal y A. Elmaghraby, «Localized analysis of paint-coat drying using dynamic speckle interferometry,» *Optics and Lasers in Engineering*, vol. 106, pp. 61-67, 2018.

[11] M. Hultman, «Real-time multi-exposure laser speckle contrast imaging of skin microcirculatory perfusion,» Linköping, 2021.

[12] E. Todorovich, M. Vázquez, E. Cozzolino, F. Ferrara, G. J. A. Bioul, A. Dai Para y L. I. Passoni, «Speckle signal processing through FPGA,» de *II Congreso de Microelectrónica Aplicada (μEA 2011)(La Plata, 7 al 9 de septiembre de 2011)*, La Plata, 2011.

[13] S. J. Flores Asenjo, «Universitat Politècnica de València,» 26 01 2021. [En línea]. Available: http://hdl.handle.net/10251/159893. [Último acceso: 12 10 2023].

[14] Hamsternz, «github,» 04 08 2015. [En línea]. Available: https://github.com/hamsternz/Artix-7-HDMI-processing. [Último acceso: 10 05 2023].

[15] A. Domínguez Torres, «Procesamiento digital de imágenes,» *Perfiles Educativos,* nº 72, 1996.

[16] R. C. Gonzalez y R. E. Woods, «Digital Image Processing (3rd Edition),» de *Color Image Processing*, Prentice-Hall, 2007, pp. 424-428.

[17] V. R. Bustamante, «AUTHOREA,» 11 07 2022. [En línea]. Available: https://www.authorea.com/users/494129/articles/576299-sensores-cmos-y-ccd. [Último acceso: 16 10 2023].

[18] K. L. Hazelwood, S. G. Olenych, J. D. Griffin, C. S. Murphy, J. A. Cathcart y M. W. Davidson, «Hamamatsu,» [En línea]. Available: https://hamamatsu.magnet.fsu.edu/articles/microscopyimaging.html. [Último acceso: 29 04 2023].

[19] R. S. Castro Rios, «ESTUDIO E IMPLEMENTACIÓN DE UN CONVERTIDOR ANALÓGICO DIGITAL Y DIGITAL ANALÓGICO SIGMA DELTA,» Barcelona, 2015.

[20] R. Teja, «ElectronicsHub,» 10 04 2021. [En línea]. Available: https://www.electronicshub.org/sample-and-hold-circuit/. [Último acceso: 15 10 2023].

[21] A. M. Tripathi, «HIGH DEFINITION MULTIMEDIA INTERFACE (HDMI),» Kochi, 2010.

[22] Svanews, «Svanews,» 07 11 2016. [En línea]. Available: https://svanews.com/2016/11/07/types-of-computer-ports-and-their-functions/. [Último acceso: 15 10 2023].

[23] A. Bhattacharya, «Semtech,» 08 10 2020. [En línea]. Available: https://blog.semtech.com/esd-protection-for-hdmi-2.0. [Último acceso: 08 10 2023].

[24] hdmi-navi.com, «HDMI-NAVI,» 03 03 2017. [En línea]. Available: http://www.hdmi-navi.com/tmds/. [Último acceso: 07 05 2023].

[25] S. Rao, K. C. Padhy y K. Kumar, «VHDL Implementation of TMDS encoder for the transmission of video signals in serial communication,» *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET),* vol. 4, nº 4, 04 2015.

[26] G. C. González Morales, «Diseño de módulo Serializador de un sistema SerDes para protocolo de comunicación PCI Express,» Guadalajara, 2015.

[27] A. Athavale y C. Christensen, High-Speed Serial I/O Made Simple, A Designers' Guide, with FPGA Applications, 1.0 ed., Xilinx, 2005, pp. 26-28.

[28] T. R. Kuphaldt, Lessons In Electric Circuits, Volume IV – Digital, 2007.

[29] B. J. L. González Reaño y B. N. E. Yalta Soplin, «PROCESAMIENTO DIGITAL DE SEÑALES IMPLEMENTANDO HPRC,» Trujillo, 2009.

[30] J. A. Castaño, V. Moreno, A. Cabrera, A. Gutiérrez y V. Martínez, «Red de Procesadores Evolutivos para solucionar el Problema de los Tres Colores. Implementación en Hardware,» *Revista Cubana de Ciencias Informáticas,* vol. 9, nº 4, pp. 155-170, 15 10 2015.

[31] U. Farooq, Z. Marrakchi y H. Mehrez, Tree Based Heterogeneous FPGA Architectures, Application Specific Exploration and Optimization, Springer, 2012.

[32] C. Sisterna, «Field programmable gate arrays (FPGAS),» *Electrónica Digital II, Facultad de Ingeniería, Universidad Nacional de San Juan (Argentina).*

[33] paguayo, «FPGA (Field Programmable Gate Array),» 18 06 2019. [En línea]. Available: https://cursos.mcielectronics.cl/2019/06/18/fpga-field-programmable-gate-array/. [Último acceso: 19 10 2023].

[34] B. Díaz Fernández, «Lenguajes de descripción hardware para la síntesis de circuitos: VHDL y Verilog : analogías y diferencias : aplicación a un caso práctico,» Madrid, 2017.

[35] A. García Durán, «Diseño e implementación de un analizador multicanal para espectometría nuclear con Zynq utilizando vivado,» Universidad Autónoma de Zacatecas, Zacatecas, 2017.

[36] W. Green, «Video Timings: VGA, SVGA, 720p, 1080p,» Project F, 26 06 2020. [En línea]. Available: https://projectf.io/posts/video-timings-vga-720p-1080p/. [Último acceso: 16 10 2023].

[37] Florentw, «Serie de video para principiantes 16: Comprensión de la sincronización de video con la IP de VTC,» AMD, 12 12 2021. [En línea]. Available: https://support.xilinx.com/s/article/899769?language=ja. [Último acceso: 16 10 2023].

[38] J. A. Arias Cruz, «Análisis multivariante de imágenes de speckle para visualización de vasos sanguíneos,» 2020.

[39] W. Moebs, S. J. Ling y J. Sanny, Física universitaria volumen 3, vol. 3, Houston, Texas: OpenStax, 2021.

[40] D. Khodadad, «Multiplexed Digital Holography incorporating Speckle Correlation,» Luleå, 2016.

[41] Digilent, «PmodDA2 Reference Manual,» 24 05 2016. [En línea]. Available: https://digilent.com/reference/_media/reference/pmod/pmodda2/pmodda2_rm.pdf. [Último acceso: 19 10 2023].

[42] FotographiArte, «APERTURA DE DIAFRAGMA,» 30 01 2020. [En línea]. Available: https://fotographiarte.home.blog/2020/01/30/apertura-de-diafragma/. [Último acceso: 19 10 2023].

[43] R. Arizaga, M. Trivi y H. Rabal, «Speckle time evolution characterization by the co-occurrence matrix analysis,» *Optics & Laser Technology*, vol. 31, nº 2, pp. 163-169, 1999.

[44] N. Herrera, «TODoS @ CICESE,» 30 06 2022. [En línea]. Available: https://todos.cicese.mx/sitio/noticia.php?n=1778. [Último acceso: 20 10 2023].

[45] J. L. González Reaño y N. E. Yalta Soplin, «PROCESAMIENTO DIGITAL DE SEÑALES IMPLEMENTANDO HPRC,» Trujillo, 2009.

Anexo A.

Código en MATLAB para la diferencia generalizada de cada píxel, se carga un total de 16 píxeles.

```
clear
clc
px11= csvread('C4signal00000.csv',6);
px21= csvread('C3signal00000.csv',6);
px12= csvread('C4signal00001.csv',6);
px22= csvread('C3signal00001.csv',6);
px13= csvread('C4signal00002.csv',6);
px23= csvread('C3signal00002.csv',6);
px14= csvread('C4signal00003.csv',6);
px24= csvread('C3signal00003.csv',6);
px31= csvread('C4signal00004.csv',6);
px41= csvread('C3signal00004.csv',6);
px32= csvread('C4signal00005.csv',6);
px42= csvread('C3signal00005.csv',6);
px33= csvread('C4signal00006.csv',6);
px43= csvread('C3signal00006.csv',6);
px34= csvread('C4signal00007.csv',6);
px44= csvread('C3signal00007.csv',6);

tpx11 = px11(:, 1); % Primera columna
tpx13 = px13(:, 1); % Primera columna
tpx14 = px14(:, 1); % Primera columna
tpx23 = px13(:, 1); % Primera columna
tpx24 = px14(:, 1); % Primera columna

dpx11 = px11(:, 2); % Segunda columna
dpx21 = px21(:, 2); % Segunda columna
dpx12 = px12(:, 2); % Segunda columna
dpx22 = px22(:, 2); % Segunda columna
dpx13 = px13(:, 2); % Segunda columna
dpx23 = px23(:, 2); % Segunda columna
dpx14 = px14(:, 2); % Segunda columna
dpx24 = px24(:, 2); % Segunda columna
dpx31 = px31(:, 2); % Segunda columna
dpx41 = px41(:, 2); % Segunda columna
dpx32 = px32(:, 2); % Segunda columna
dpx42 = px42(:, 2); % Segunda columna
dpx33 = px33(:, 2); % Segunda columna
dpx43 = px43(:, 2); % Segunda columna
dpx34 = px34(:, 2); % Segunda columna
dpx44 = px44(:, 2); % Segunda columna

a1Long=size(px11, 1);
a13Long=size(px13, 1);
a14Long=size(px14, 1);
a23Long=size(px23, 1);
a24Long=size(px24, 1);

px11_v = reshape(dpx11', 1, a1Long);
px21_v = reshape(dpx21', 1, a1Long);
px12_v = reshape(dpx12', 1, a1Long);
```

```
px22_v = reshape(dpx22', 1, a1Long);
px13_v = reshape(dpx13', 1, a13Long);
px23_v = reshape(dpx23', 1, a23Long);
px14_v = reshape(dpx14', 1, a14Long);
px24_v = reshape(dpx24', 1, a24Long);
px31_v = reshape(dpx31', 1, a1Long);
px41_v = reshape(dpx41', 1, a1Long);
px32_v = reshape(dpx32', 1, a1Long);
px42_v = reshape(dpx42', 1, a1Long);
px33_v = reshape(dpx33', 1, a1Long);
px43_v = reshape(dpx43', 1, a1Long);
px34_v = reshape(dpx34', 1, a1Long);
px44_v = reshape(dpx44', 1, a1Long);

tiempo=linspace(0,500,length(px11));
t13=linspace(0,500,length(px13));
t14=linspace(0,500,length(px14));
t23=linspace(0,500,length(px23));
t24=linspace(0,500,length(px24));

i=1:a1Long;

Tv=[0 25 67 100 129 170 211 249 294 338 386 437 490 546];
Gv=[0.415 0.41 0.405 0.4 0.395 0.39 0.385 0.38 0.375 0.37 0.365 0.36
0.355 0.35];

%Tv=[0 34 67 111 145 188 246 298 352 410 516 594 685 782 902 1083];
%Gv=[0.445 0.44 0.435 0.43 0.425 0.42 0.415 0.41 0.405 0.4 0.395 0.390
0.385 0.38 0.375 0.37];

figure
subplot(4,4,1)
plot(tiempo,px11_v)
title('P.1,1')
xlabel('time (s)')
ylabel('Volts (v)')
xlim([0 500])

subplot(4,4,2)
plot(tiempo,px12_v)
title('P.1,2')
xlabel('time (s)')
ylabel('Volts (v)')
xlim([0 500])

subplot(4,4,3)
plot(t13,px13_v)
title('P.1,3')
xlabel('time (s)')
ylabel('Volts (v)')
xlim([0 500])

subplot(4,4,4)
plot(t14,px14_v)
title('P.1,4')
xlabel('time (s)')
```

```
ylabel('Volts (v)')
xlim([0 500])

subplot(4,4,5)
plot(tiempo,px21_v)
title('P.2,1')
xlabel('time (s)')
ylabel('Volts (v)')
xlim([0 500])

subplot(4,4,6)
plot(tiempo,px22_v)
title('P.2,2')
xlabel('time (s)')
ylabel('Volts (v)')
xlim([0 500])

subplot(4,4,7)
plot(t23,px23_v)
title('P.2,3')
xlabel('time (s)')
ylabel('Volts (v)')
xlim([0 500])

subplot(4,4,8)
plot(t24,px24_v)
title('P.2,4')
xlabel('time (s)')
ylabel('Volts (v)')
xlim([0 500])

subplot(4,4,9)
plot(tiempo,px31_v)
title('P.3,1')
xlabel('time (s)')
ylabel('Volts (v)')
xlim([0 500])

subplot(4,4,10)
plot(tiempo,px32_v)
title('P.3,2')
xlabel('time (s)')
ylabel('Volts (v)')
xlim([0 500])

subplot(4,4,11)
plot(tiempo,px33_v)
title('P.3,3')
xlabel('time (s)')
ylabel('Volts (v)')
xlim([0 500])

subplot(4,4,12)
plot(tiempo,px34_v)
title('P.3,4')
xlabel('time (s)')
```

```
ylabel('Volts (v)')
xlim([0 500])

subplot(4,4,13)
plot(tiempo,px41_v)
title('P.4,1')
xlabel('time (s)')
ylabel('Volts (v)')
xlim([0 500])

subplot(4,4,14)
plot(tiempo,px42_v)
title('P.4,2')
xlabel('time (s)')
ylabel('Volts (v)')
xlim([0 500])

subplot(4,4,15)
plot(tiempo,px43_v)
title('P.4,3')
xlabel('time (s)')
ylabel('Volts (v)')
xlim([0 500])

subplot(4,4,16)
plot(tiempo,px44_v)
title('P.4,4')
xlabel('time (s)')
ylabel('Volts (v)')
xlim([0 500])
```

Anexo B.

Código en MATLAB para el promedio de la diferencia generalizada, estudio gravimétrico y normalización.

```
clear
clc
load("p11.mat")

lmax=length(Gv);
x0=1;
for i=2:lmax
    index=find( tiempo<Tv(i));
    x1=index(length(index));
    sumando=a1datavector(x0:x1);
    GS(i-1)=mean(sumando);

    tv(i-1)=tiempo(x0);
    x0=x1;
end
GS(i)=GS(i-1);
tv(i)=tiempo(x1);
Tv(14)=500;
GS_estandar=(GS-min(GS))/(max(GS)-min(GS));
Gv_estandar=(Gv-min(Gv))/(max(Gv)-min(Gv));

figure
subplot(4,1,1)
plot(tiempo,a1datavector)
legend('GD en P.1,1')
title('(a) Diferencia Generalizada del P.1,1')
xlabel('time (s)')
ylabel('Volts (v)')

subplot(4,1,2)
plot(tv,GS,'-square')
legend('GD en P.1,1')
title('(b) Promedio DG del P.1,1')
xlabel('time (s)')
ylabel('Volts (v)')

subplot(4,1,3)
plot(Tv, Gv,'-square')
legend('Peso')
title('(c) Estudio Gravimétrico')
xlabel('time (s)')
ylabel('Gramos (g)')
xlim([0 500])

subplot(4,1,4)
plot(tv,GS_estandar,'-o')
title('(d) Promedio DG P.1,1 vs Peso Normalizado')
hold on
plot(Tv, Gv_estandar,'-square')
xlim([0 500])
```

```
hold off
xlabel('time (s)')
ylabel('valor por unidad (p.u.)')
legend('Promedio DG','Peso')

dataF=[Gv',GS']
```

Anexo C.

Tabla 5: Peso vs Promedio GD del P.1,2.

Peso (gramos)	Promedio GD (volts)
0,415	1,893
0,41	1,581
0,405	1,142
0,4	0,898
0,395	0,756
0,39	0,533
0,385	0,445
0,38	0,455
0,375	0,415
0,37	0,382
0,365	0,350
0,36	0,439
0,355	0,370

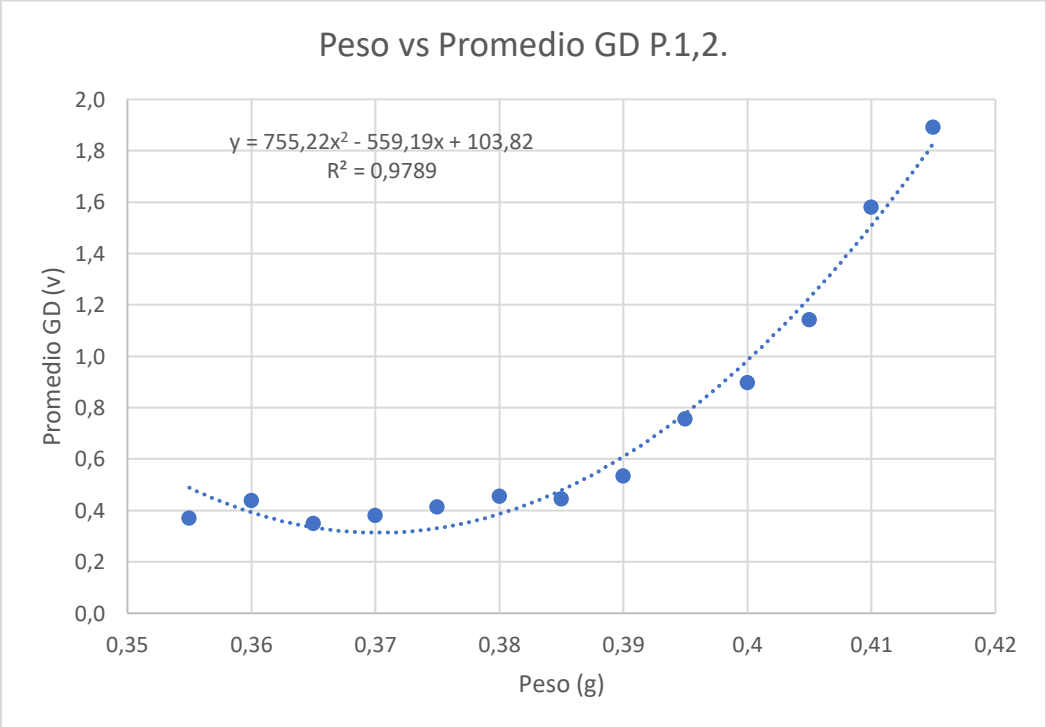


Fig.C1: Peso vs Promedio GD P.1,2.

Tabla 6: Peso vs Promedio GD del P.1,3.

Peso (gramos)	Promedio GD (volts)
0,415	1,859
0,41	1,596
0,405	1,178
0,4	0,944
0,395	0,790
0,39	0,533
0,385	0,459
0,38	0,471
0,375	0,421
0,37	0,380
0,365	0,354
0,36	0,459
0,355	0,390

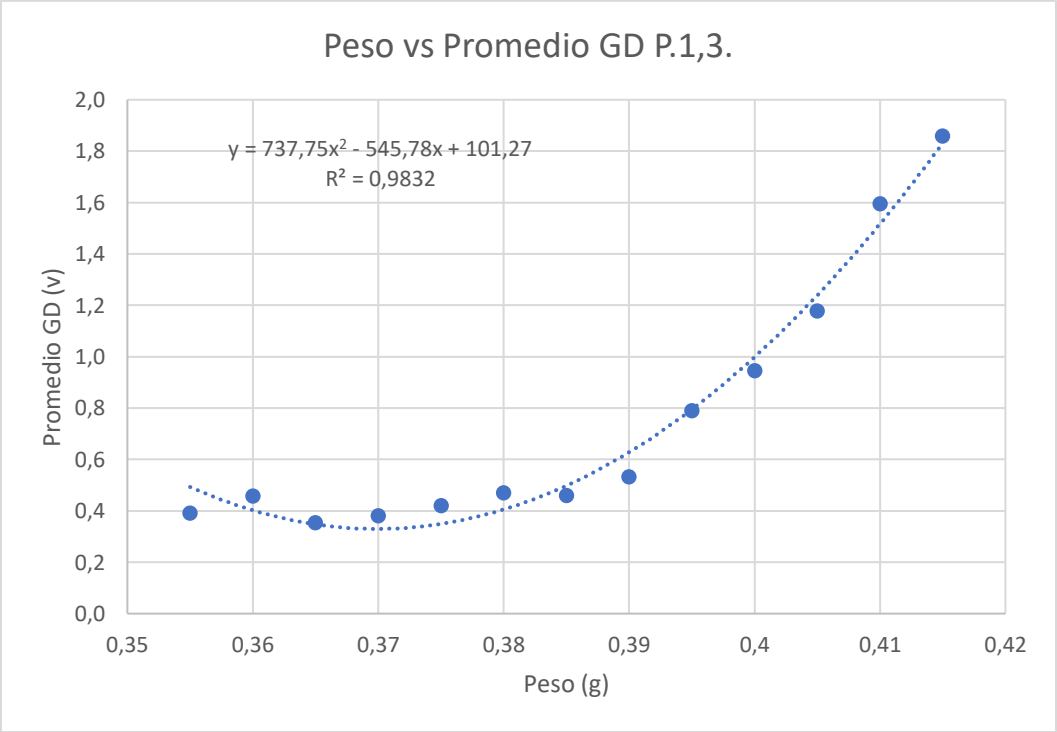


Fig.C2: Peso vs Promedio GD P.1,3.

Tabla 7: Peso vs Promedio GD del P.1,4.

Peso (gramos)	Promedio GD (volts)
0,415	1,832
0,41	1,575
0,405	1,144
0,4	0,935
0,395	0,784
0,39	0,522
0,385	0,459
0,38	0,456
0,375	0,431
0,37	0,390
0,365	0,351
0,36	0,447
0,355	0,365

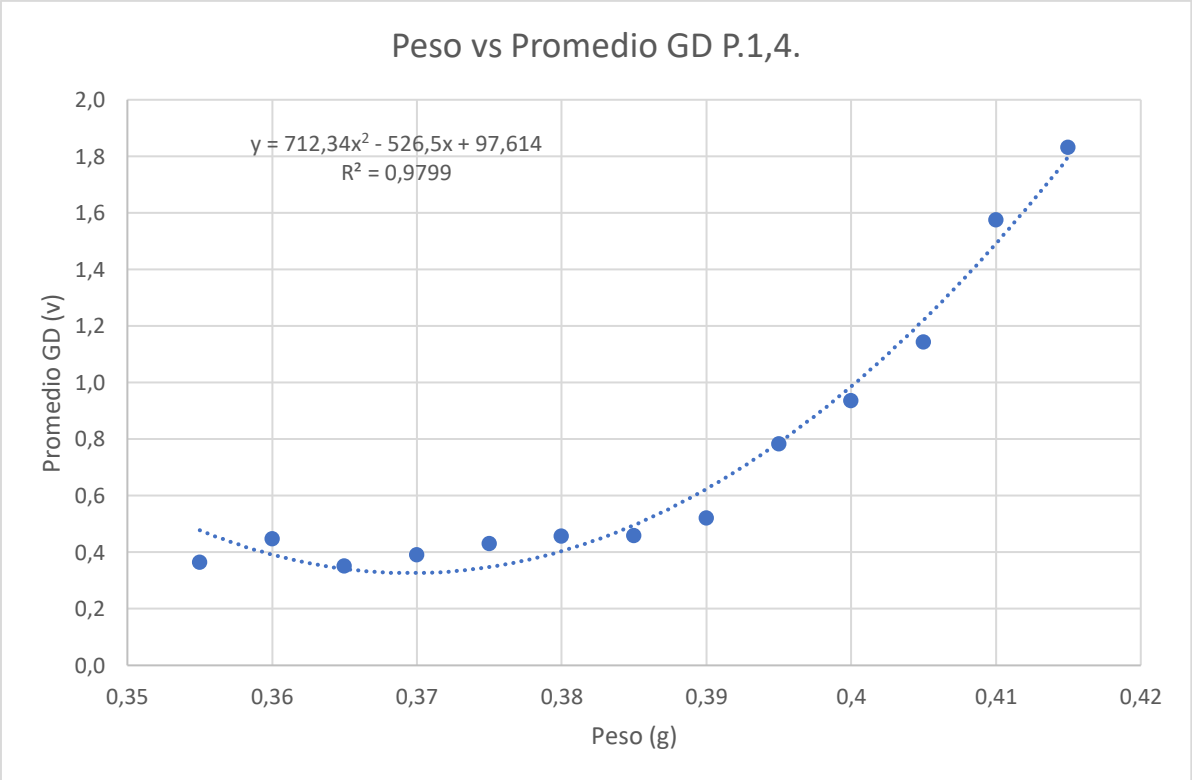


Fig.C3: Peso vs Promedio GD P.1,4.

Tabla 8: Peso vs Promedio GD del P.2,1.

Peso (gramos)	Promedio GD (volts)
0,415	1,878
0,41	1,549
0,405	1,199
0,4	0,904
0,395	0,716
0,39	0,546
0,385	0,460
0,38	0,466
0,375	0,428
0,37	0,394
0,365	0,335
0,36	0,438
0,355	0,341

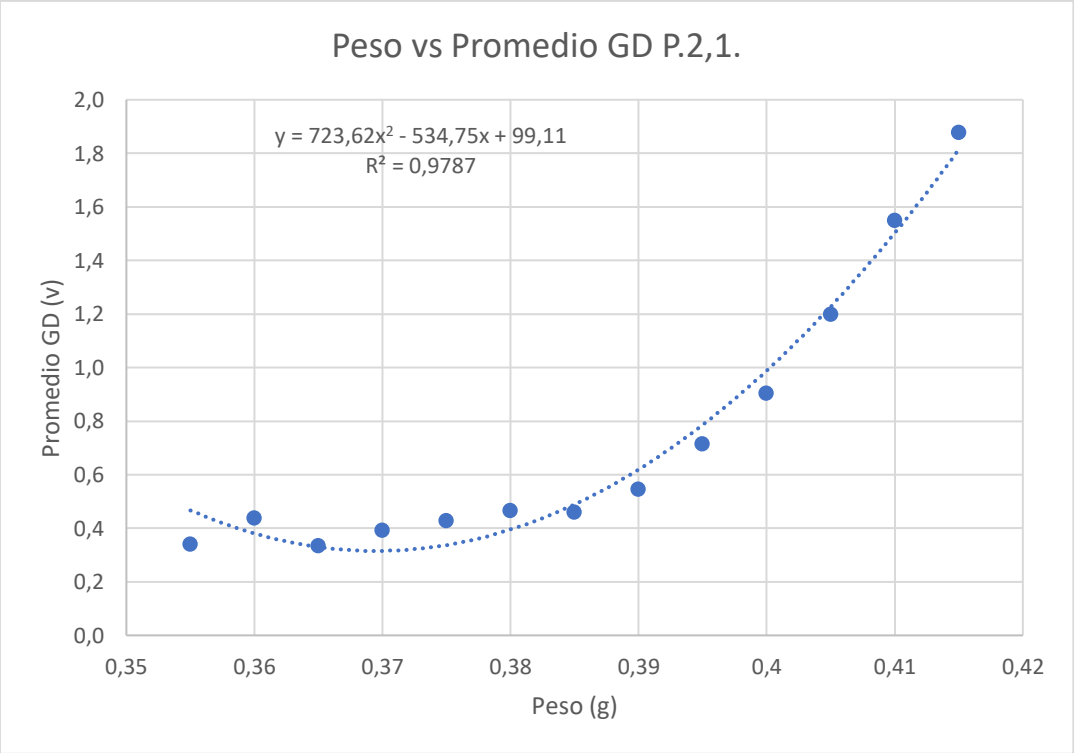


Fig.C4: Peso vs Promedio GD P.2,1.

Tabla 9: Peso vs Promedio GD del P.2,2.

Peso (gramos)	Promedio GD (volts)

0,415	1,908
0,41	1,586
0,405	1,200
0,4	0,904
0,395	0,706
0,39	0,534
0,385	0,452
0,38	0,457
0,375	0,416
0,37	0,374
0,365	0,336
0,36	0,443
0,355	0,349

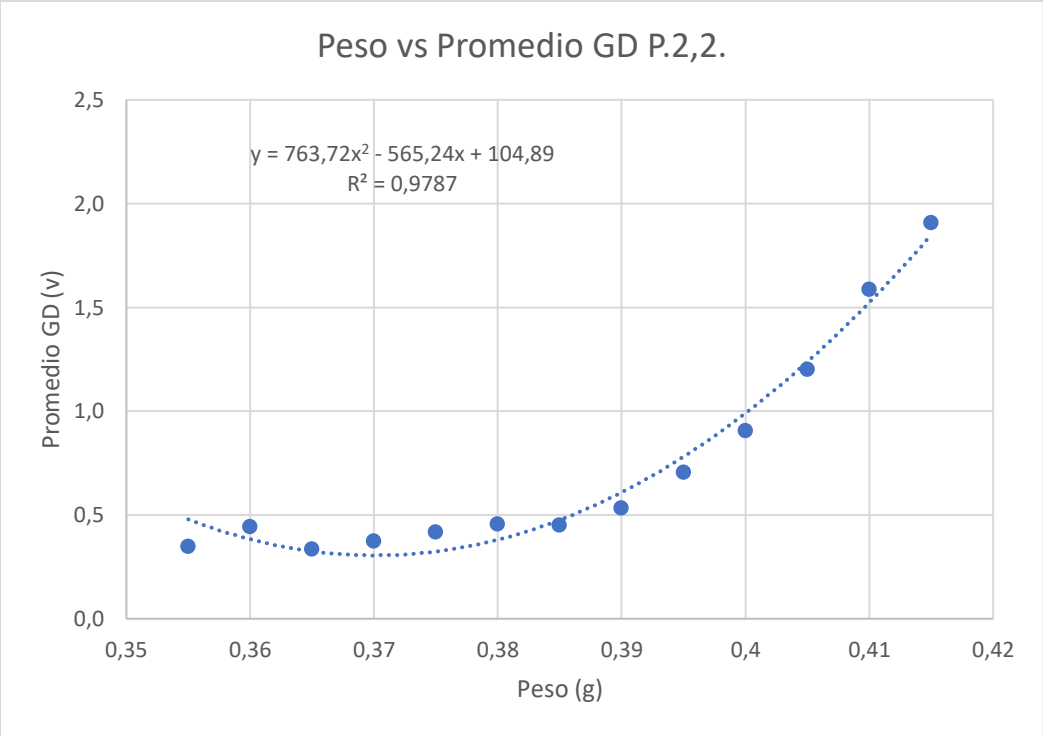


Fig.C5: Peso vs Promedio GD P.2,2.

Tabla 10: Peso vs Promedio GD del P.2,3.

Peso (gramos)	Promedio GD (volts)
0,415	1,870
0,41	1,635

0,405	1,222
0,4	0,946
0,395	0,733
0,39	0,532
0,385	0,457
0,38	0,456
0,375	0,413
0,37	0,385
0,365	0,333
0,36	0,464
0,355	0,372

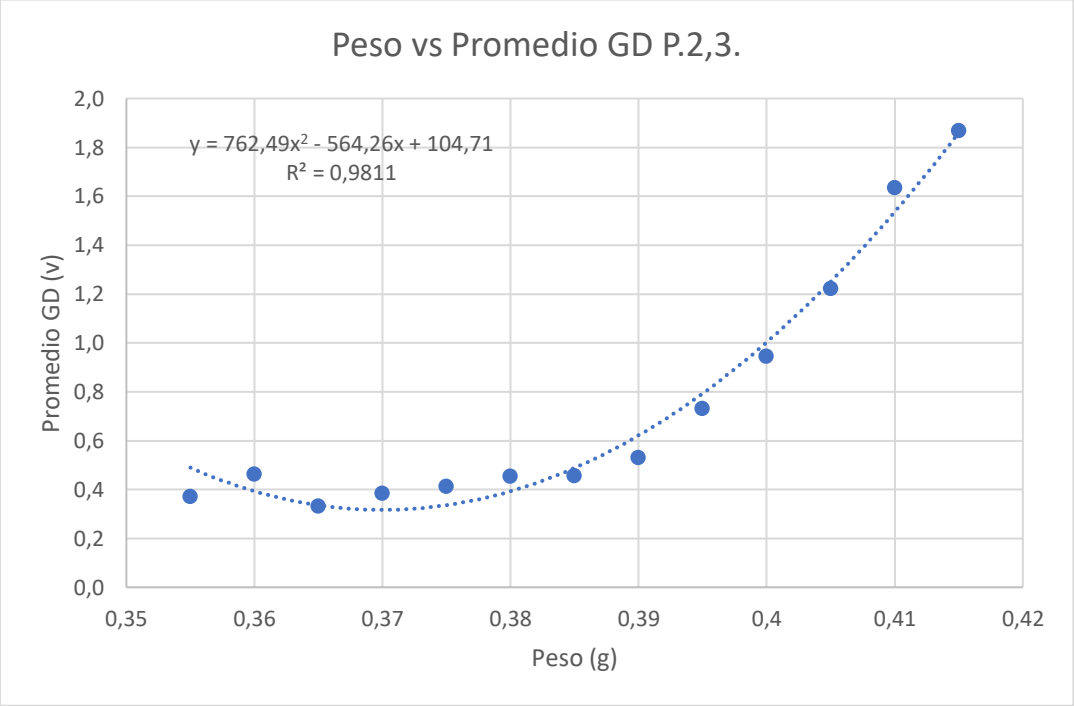


Fig.C6: Peso vs Promedio GD P.2,3.

Tabla 11: Peso vs Promedio GD del P.2,4.

Peso (gramos)	Promedio GD (volts)
0,415	1,801
0,41	1,623
0,405	1,195

0,4	0,935
0,395	0,710
0,39	0,537
0,385	0,455
0,38	0,432
0,375	0,419
0,37	0,386
0,365	0,328
0,36	0,449
0,355	0,363

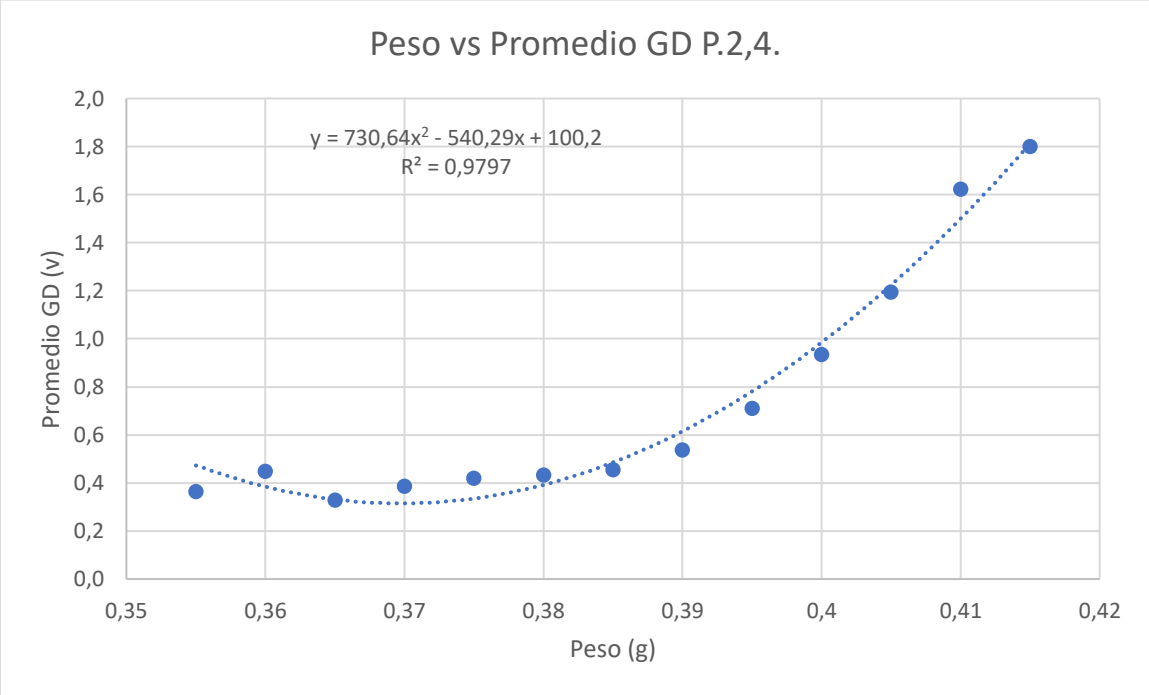


Fig.C7: Peso vs Promedio GD P.2,4.

Tabla 12: Peso vs Promedio GD del P.3,1.

Peso (gramos)	Promedio GD (volts)
0,415	1,859
0,41	1,575
0,405	1,267
0,4	0,974
0,395	0,705

0,39	0,539
0,385	0,486
0,38	0,480
0,375	0,429
0,37	0,408
0,365	0,340
0,36	0,449
0,355	0,357

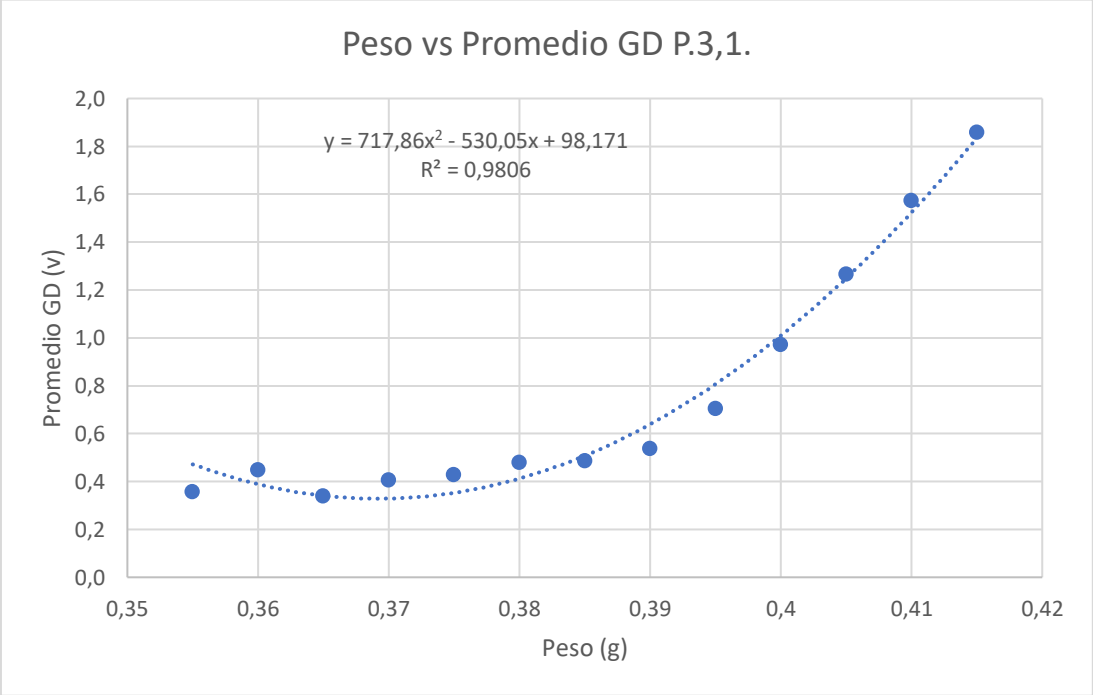


Fig.C8: Peso vs Promedio GD P.3,1.

Tabla 13: Peso vs Promedio GD del P.3,2.

Peso (gramos)	Promedio GD (volts)
0,415	1,880
0,41	1,628
0,405	1,247
0,4	0,978
0,395	0,694
0,39	0,539
0,385	0,463

0,38	0,480
0,375	0,425
0,37	0,404
0,365	0,335
0,36	0,452
0,355	0,358

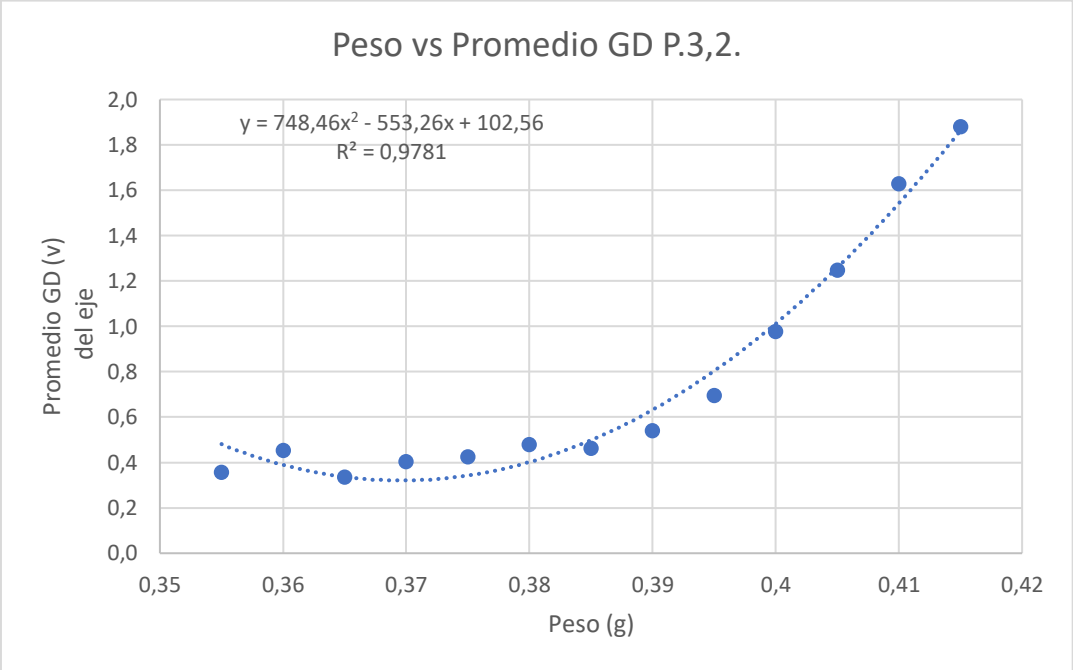


Fig.C9: Peso vs Promedio GD P.3,2.

Tabla 14: Peso vs Promedio GD del P.3,3.

Peso (gramos)	Promedio GD (volts)
0,415	1,862
0,41	1,665
0,405	1,241
0,4	0,990
0,395	0,707
0,39	0,558
0,385	0,475
0,38	0,456

0,375	0,415
0,37	0,410
0,365	0,340
0,36	0,461
0,355	0,429

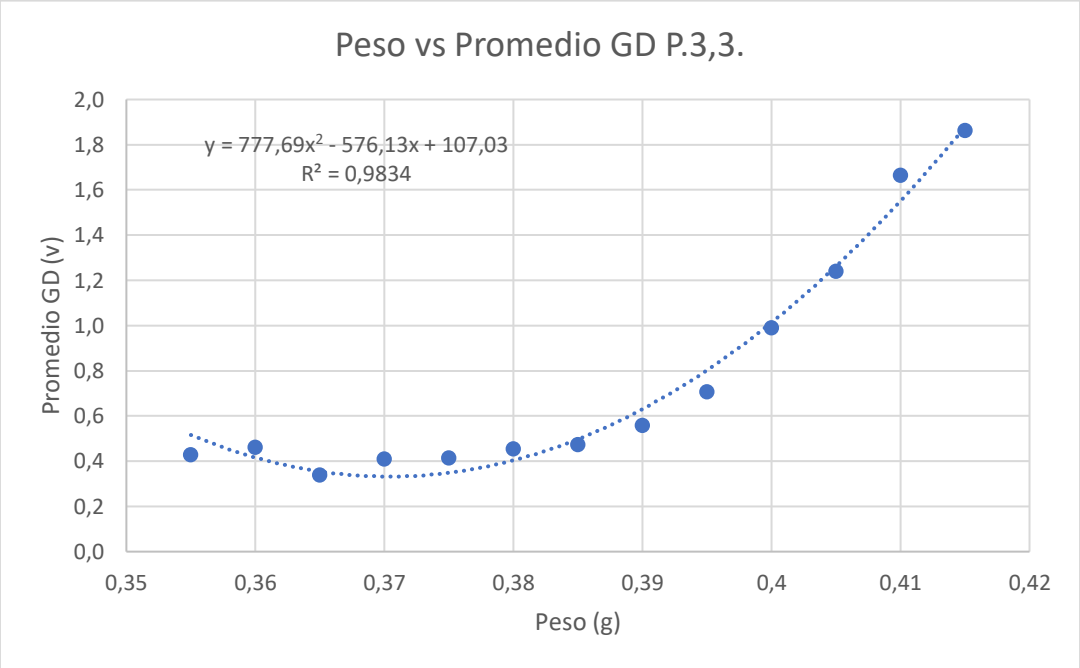


Fig.C10: Peso vs Promedio GD P.3,3.

Tabla 15: Peso vs Promedio GD del P.3,4.

Peso (gramos)	Promedio GD (volts)
0,415	1,865
0,41	1,620
0,405	1,240
0,4	0,975
0,395	0,706
0,39	0,556
0,385	0,464
0,38	0,434
0,375	0,408

0,37	0,409
0,365	0,342
0,36	0,456
0,355	0,419

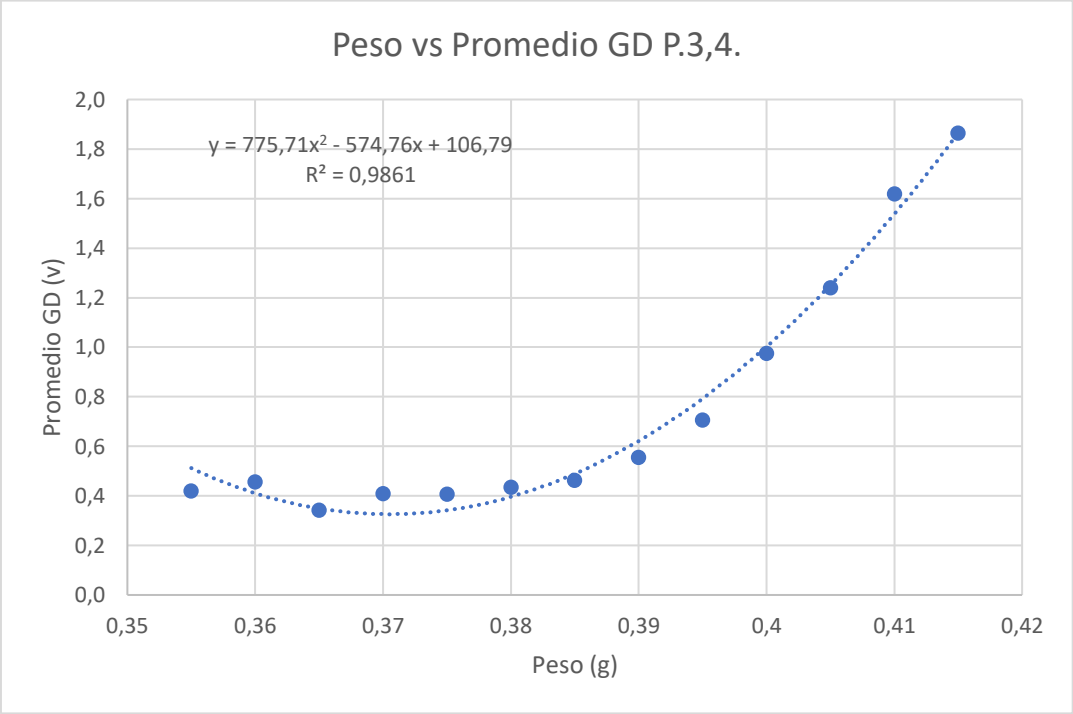


Fig.C11: Peso vs Promedio GD P.3,4.

Tabla 16: Peso vs Promedio GD del P.4,1.

Peso (gramos)	Promedio GD (volts)
0,415	1,900
0,41	1,585
0,405	1,255
0,4	0,988
0,395	0,704
0,39	0,512
0,385	0,470
0,38	0,470
0,375	0,447
0,37	0,402

0,365	0,331
0,36	0,433
0,355	0,362

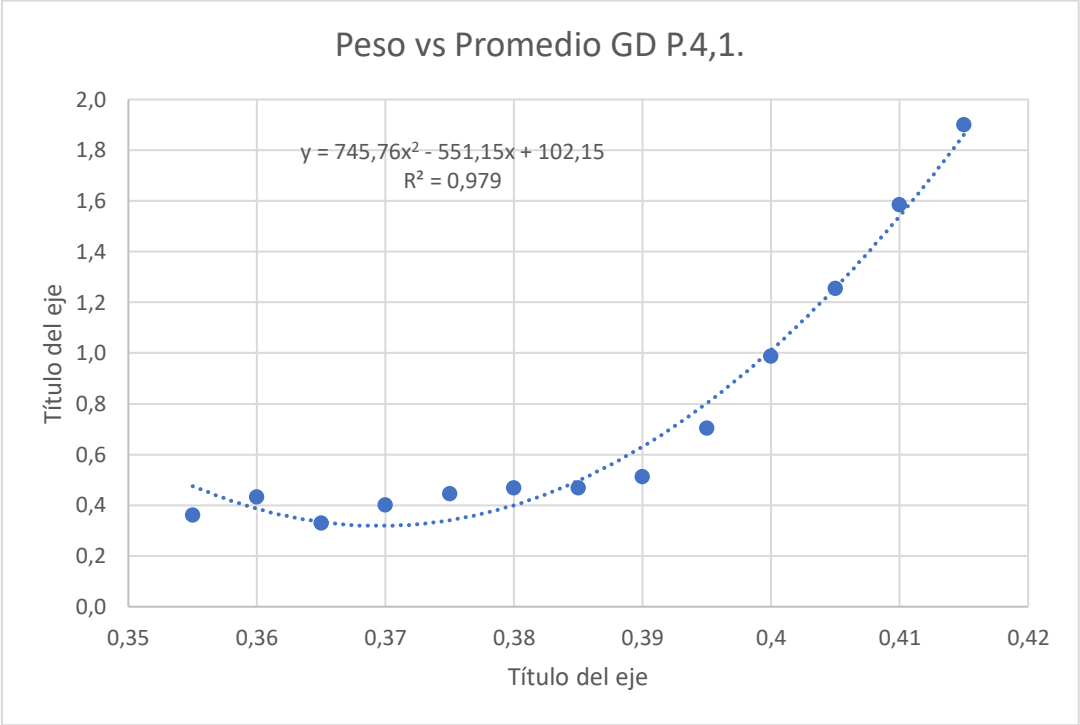


Fig.C12: Peso vs Promedio GD P.4,1.

Tabla 17: Peso vs Promedio GD del P.4,2.

Peso (gramos)	Promedio GD (volts)
0,415	1,952
0,41	1,616
0,405	1,249
0,4	0,997
0,395	0,694
0,39	0,523
0,385	0,459
0,38	0,477
0,375	0,435
0,37	0,394
0,365	0,327

0,36	0,436
0,355	0,387

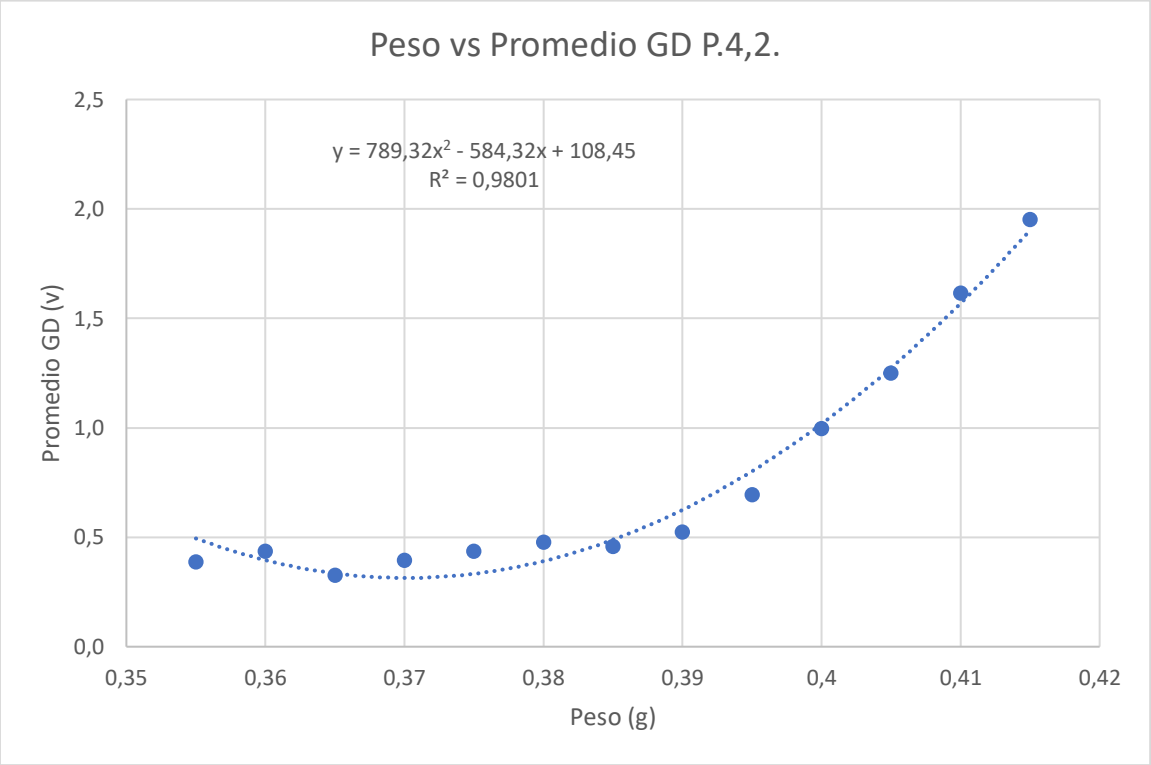


Fig.C13: Peso vs Promedio GD P.4,2.

Tabla 18: Peso vs Promedio GD del P.4,3.

Peso (gramos)	Promedio GD (volts)
0,415	1,933
0,41	1,657
0,405	1,259
0,4	1,027
0,395	0,715
0,39	0,557
0,385	0,473
0,38	0,472
0,375	0,433
0,37	0,408
0,365	0,334
0,36	0,453

0,355	0,430

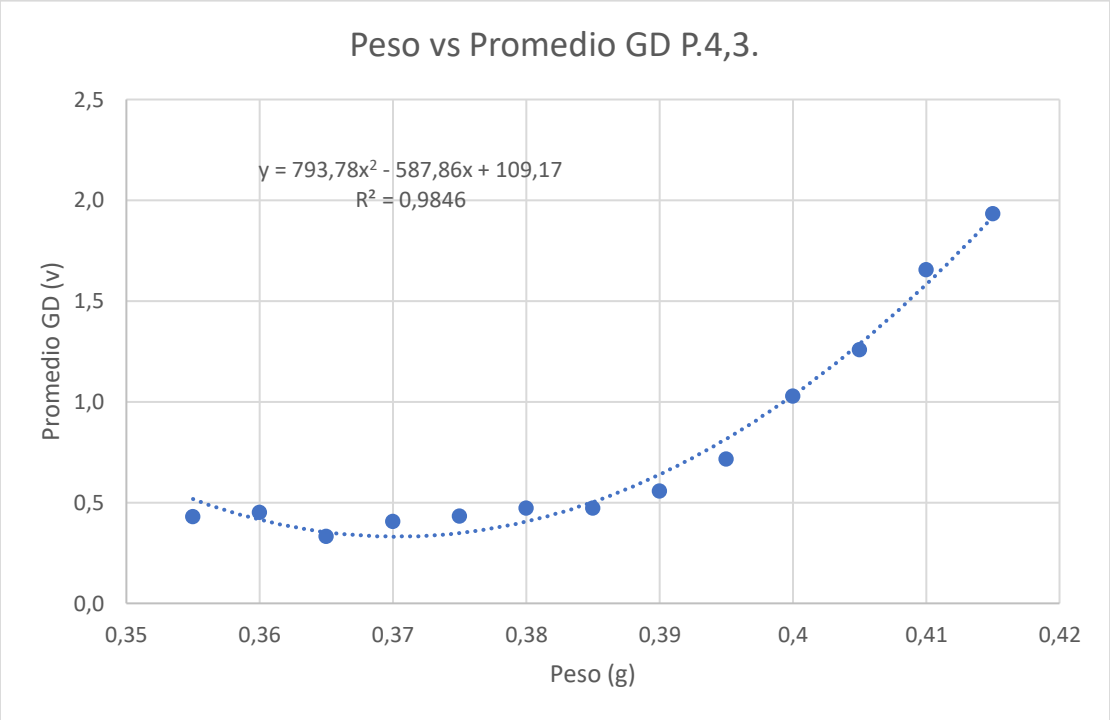


Fig.C14: Peso vs Promedio GD P.4,3.

Tabla 19: Peso vs Promedio GD del P.4,4.

Peso (gramos)	Promedio GD (volts)
0,415	1,921
0,41	1,616
0,405	1,255
0,4	1,016
0,395	0,727
0,39	0,563
0,385	0,467
0,38	0,449
0,375	0,417
0,37	0,406
0,365	0,342
0,36	0,463
0,355	0,408

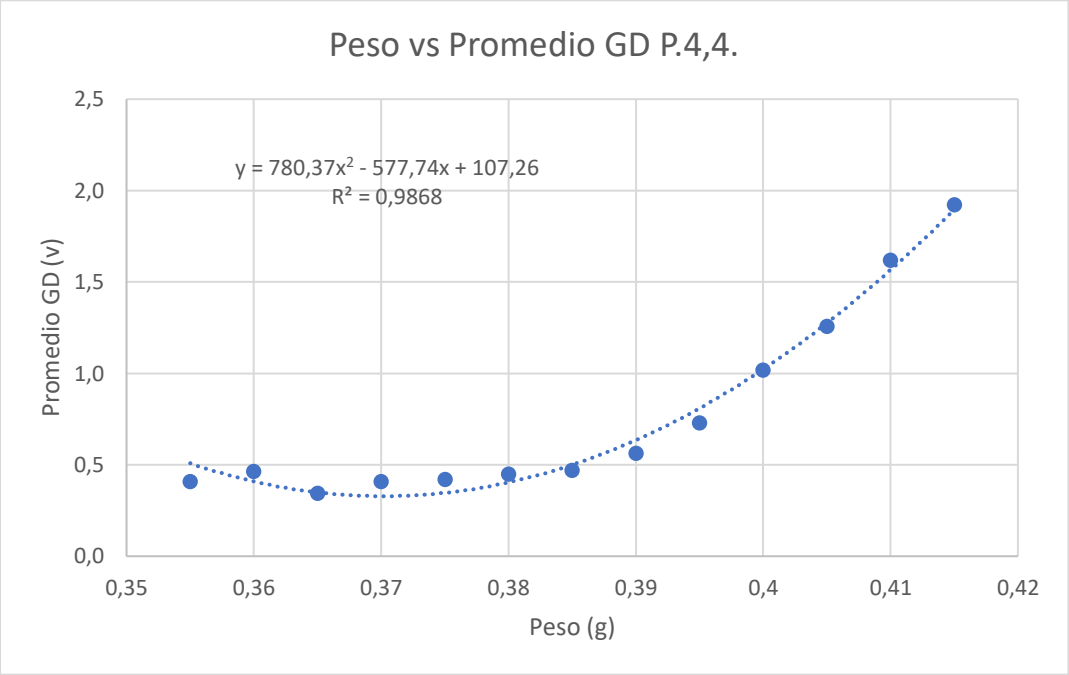


Fig.C15: Peso vs Promedio GD P.4,4.

Anexo D.

Tabla 20: EMC y MAPE de cada Píxel.

Pixel (x,y)	ECM	MAPE
P.1,1	0,0057	12,59%
P.1,2	0,0050	11,65%
P.1,3	0,0039	9,95%
P.1,4	0,0045	10,78%
P.2,1	0,0049	11,97%
P.2,2	0,0052	12,39%
P.2,3	0,0046	11,17%
P.2,4	0,0047	10,87%
P.3,1	0,0045	11,10%
P.3,2	0,0054	12,01%
P.3,3	0,0040	9,43%
P.3,4	0,0033	9,00%
P.4,1	0,0052	11,79%
P.4,2	0,0052	11,66%
P.4,3	0,0039	9,98%
P.4,4	0,0033	9,42%