UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

INSTITUTO DE INVESTIGACIÓN EN CIENCIAS BÁSICAS Y APLICADAS

CENTRO DE INVESTIGACIÓN EN INGENIERÍA Y CIENCIAS APLICADAS

TÍTULO DE LA TESIS

IMPLEMENTACIÓN Y EVALUACIÓN DE ALGORITMOS PARA FILTRADO Y DESENVOLVIMIENTO DE FASE EN MICROSCOPÍA HOLOGRÁFICA DIGITAL, MEDIANTE DESPLAZAMIENTO DE FASE UTILIZANDO ARQUITECTURAS DE HARDWARE FPGA (DE1-SOC)

TESIS PARA OBTENER EL GRADO DE:

DOCTORADO EN INGENIERÍA Y CIENCIAS APLICADAS

CON OPCIÓN TERMINAL EN TECNOLOGÍA ELÉCTRICA

PRESENTA:

M.I.C.A VICTOR MANUEL JUAREZ NUÑEZ

DIRECTORES: **DR. DARWIN MAYORGA CRUZ.**

DR. ÁLVARO ZAMUDIO LARA.

CUERNAVACA, MORELOS **FEBRERO 2019**

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

Instituto de Investigación en Ciencias Básicas y Aplicadas

INSTITUTO DE INVESTIGACION EN CIENCIAS BASICAS Y APLICADAS
CENTRO DE INVESTIGACION EN INGENIERIA Y CIENCIAS APLICADAS
Jefatura de Posgrado en Ingeniería y Ciencias Aplicadas

"2019, a 100 años del asesinato del General Emiliano Zapata Salazar"

Cuernavaca, Morelos, a 25 de enero de 2019.

CIICAp

DR. ROSENBERG JAVIER ROMERO DOMÍNGUEZ
COORDINADOR DE POSGRADO
EN INGENIERÍA Y CIENCIAS APLICADAS
P R E S E N T E

Atendiendo a la solicitud para emitir DICTAMEN sobre la revisión de la TESIS "IMPLEMENTACIÓN Y EVALUACIÓN DE ALGORITMOS PARA FILTRADO Y DESENVOLVIMIENTO DE FASE EN MICROSCOPÍA HOLOGRÁFICA DIGITAL, MEDIANTE DESPLAZAMIENTO DE FASE UTILIZANDO ARQUITECTURAS DE HARDWARE FPGA (DE1-SOC)" que presenta el alumno **VÍCTOR MANUEL JUÁREZ NÚÑEZ,** para obtener el título de **DOCTORADO EN INGENIERÍA Y CIENCIAS APLICADAS** con opción terminal en **TECNOLOGÍA ELÉCTRICA.**

Nos permitimos informarle que nuestro voto es:

NOMBRE	DICTAMEN	FIRMA
DR. PEDRO ANTONIO MÁRQUEZ AGUILAR	Aprobado	
DR. JOSÉ ANTONIO MARBÁN SALGADO	Aprobado	Jose Antonio MS
DR. J JESÚS ESCOBEDO ALATORRE	Aprobado	
DR. JOSÉ GUADALUPE VELÁSQUEZ AGUILAR	Aprobado	
DR. OSCAR SARMIENTO MARTÍNEZ	APROBADO	
DR. ÁLVARO ZAMUDIO LARA	Aprobado	
DR. DARWIN MAYORGA CRUZ	Aprobado	

PLAZO PARA LA REVISIÓN 20 DÍAS HÁBILES (A PARTIR DE LA FECHA DE RECEPCIÓN DEL DOCUMENTO)

NOTA. POR CUESTION DE REGLAMENTACIÓN LE SOLICITAMOS NO EXCEDER EL PLAZO SEÑALADO, DE LO CONTRARIO LE AGRADECEMOS SU ATENCIÓN Y NUESTRA INVITACIÓN SERÁ CANCELADA.

Av. Universidad 1001 Col. Chamilpa, Cuernavaca Morelos, México, 62209, Edificio 48.
Tel. (777) 329 70, 00, Ext. 6208 /raquel.sotelo@uaem.mx

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

Instituto de Investigación en Ciencias Básicas y Aplicadas

INSTITUTO DE INVESTIGACION EN CIENCIAS BASICAS Y APLICADAS
CENTRO DE INVESTIGACION EN INGENIERIA Y CIENCIAS APLICADAS

Jefatura de Posgrado en Ingeniería y Ciencias Aplicadas

"2019, a 100 años del asesinato del General Emiliano Zapata Salazar"

ASUNTO: APROBACIÓN DE TESIS
Cuernavaca, Morelos, 25 de enero de 2019.

CIICAp

C. VÍCTOR MANUEL JUÁREZ NÚÑEZ
PRESENTE

Por este conducto le notifico que su tesis de Doctorado "IMPLEMENTACIÓN Y EVALUACIÓN DE ALGORITMOS PARA FILTRADO Y DESENVOLVIMIENTO DE FASE EN MICROSCOPÍA HOLOGRÁFICA DIGITAL, MEDIANTE DESPLAZAMIENTO DE FASE UTILIZANDO ARQUITECTURAS DE HARDWARE FPGA (DE1-SOC)"

Fue aprobada en su totalidad por el jurado revisor y examinador integrado por los ciudadanos

NOMBRE	FIRMA
DR. PEDRO ANTONIO MÁRQUEZ AGUILAR	
DR. JOSÉ ANTONIO MARBÁN SALGADO	Jose Antonio M.S.
DR. J JESÚS ESCOBEDO ALATORRE	
DR. JOSÉ GUADALUPE VELÁSQUEZ AGUILAR	
DR. OSCAR SARMIENTO MARTÍNEZ	
DR. ÁLVARO ZAMUDIO LARA	
DR. DARWIN MAYORGA CRUZ	

Por consiguiente, se autoriza a editar la presentación definitiva de su trabajo de investigación para culminar en la defensa oral del mismo.

Sin otro particular aprovecho la ocasión para enviarle un cordial saludo.

Atentamente
Por una humanidad culta
Una universidad de excelencia

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS
JEFATURA DE POSGRADO
INGENIERÍA Y CIENCIAS APLIC

DR. ROSENBERG JAVIER ROMERO DOMÍNGUEZ
COORDINADOR DEL POSGRADO EN INGENIERÍA Y CIENCIAS APLICADAS

CIICA

RJRD*RSU/nmc

"CIICAp, XX Aniversario, 20 años de Investigación de Excelencia e Innovación"

Av. Universidad 1001 Col. Chamilpa, Cuernavaca Morelos, México, 62209, Edificio 48.
Tel. (777) 329 70, 00, Ext. 6208 /raquel.sotelo@uaem.mx

Agradecimientos

Agradezco sinceramente a los directores de mi tesis, al Dr. Darwin Mayorga Cruz, que fue desde un principio el impulsor de mi trabajo en esta carrera que mediante su conocimiento y consejo, además de su apoyo para la culminación de este tesis, la sencillez y su alto grado de humanidad que me mostro fue pieza importante, así mismo al Dr. Álvaro Zamudio Lara, que mediante sus recomendaciones, guía y la aportación de sus conocimientos trajeron a mi vida dentro de este centro un sentido de trabajo, honestidad y humildad. Ciertamente que sin su ayuda no hubiera sido posible este trabajo.

Mi agradecimiento al Dr. Jesús Escobedo Alatorre que gracias a su apoyo y conocimientos que me brindo a lo largo de este camino parte importante en el término de este trabajo, gracias sinceramente; del mismo modo a cada uno de los doctores que me compartieron su conocimiento dentro de este centro de investigación al Dr. Pedro A. Márquez Aguilar, Dr. Miguel A. Basurto Pensado, Dra. Margarita Tecpoyótl Torres, Dr. Volodymyr Grimalsky, Dr. Antonio Rodríguez Martínez, Dr. Gilberto Anzueto Sánchez, Dra. Viridiana H. León y al Dr. Mykola Kakazyey. Así mismo un agradecimiento al Dr. José Antonio Marban Salgado, por su amistad y consejos que siempre brinda a todos, gracias por todo.

La dedicación de este trabajo se la debo a toda mi familia, principalmente a mi esposa y a mis adorables y hermosas hijas, y sobre todo y ante todo a Dios.

Agradezco al Centro de Investigación en Ingeniería y Ciencias Aplicadas, así como al personal académico y administrativo por las facilidades otorgadas.

Agradecimientos al Consejo Nacional de Ciencia y Tecnología (CONACYT, México) el apoyo económico brindado durante el desarrollo de este trabajo de Doctorado.

Índice general

Índice general i
Índice de figuras iv
Índice de tablas viii
Resumen x
Abstract xiii
Motivación xv
Objetivo xvi
Justificación xvii
CAPÍTULO 1 1
Introducción y antecedentes 1
1.1 Los FPGA. 1
CAPÍTULO 2 8
Marco teórico 8
2.1 Introducción 8
2.2 Holografía 9
2.2.1 Holografía convencional (analógica) 9
2.2.2 Holografía digital 11
2.2.3 Referencias de implementación 12
2.2.4 Grabación y reconstrucción de hologramas digitales 14
2.2.5 Holografía digital mediante desplazamiento de fase 17

2.3 Microscopía holográfica digital 19
2.3.1 Modificación de fase en DHM 20
2.4 Filtros digitales. 21
2.4.1 Filtros espaciales 22
2.4.2 Filtro Bilateral 24
2.4.3 Transformada de Fourier 25
2.5 FPGA 27
2.5.1 Dispositivos FPGA 28
2.5.2 Dispositivos basados en SRAM 30
2.5.3 Arquitecturas Basadas en LUT 31
2.5.4 CLB y LAB 32
2.5.5 RAM incorporadas 32
2.5.6 Multiplicadores integrados, sumadores, MAC, etc. 33
2.5.7 Núcleos de microprocesador duro 34
2.5.8 Núcleos de microprocesador blando 35
2.5.9 Árboles de reloj y Administradores de reloj 36
2.5.10 E/S de propósito general 37
2.5.11 Estándares configurables de E/S 37
2.5.12 SoC FPGAs 38
2.6 OPENCL 40
2.6.1 Referencias de implementación. 40
2.6.1 Fundamentos conceptuales de OpenCL 42
2.6.3 Modelo de plataforma 42
2.6.4 Modelo de ejecución 43

2.6.5 Cómo se ejecuta un Kernel en un dispositivo OpenCL 44
2.6.6 Contexto (context) 46
2.6.7 Cola de Comandos - (Command-Queues) 47
2.6.8 Modelo de memoria 47
2.6.9 Modelos de programación 49
2.6.11 Modelo de Programación de Tarea en Paralelo 50
CAPÍTULO 3 52
Desarrollo experimental 52
3.1 Elementos ópticos, conceptos básicos 52
3.2 Elementos electrónicos 55
3.3 Plataforma de desarrollo: DE1-SoC Board 57
3.4 Arreglo experimental para PSDHM 58
3.5 Desarrollo metodológico 60
3.6 Obtención de las imágenes 61
3.7 Evaluación del algoritmo de PS (Phase Shift) en Matlab® 62
3.8 Configuración e instalación Software 66
CAPÍTULO 4 68
Resultados 68
4.1 Pruebas de calibración del arreglo para PSDHM. 68
4.2 Pruebas del algoritmo PS en el FPGA 69
4.3 Pruebas del FPGA con algoritmos escritos en C y OpenCL del Filtro Bilateral y la Transformada Rápida de Fourier. 71
4.4 Ejecución del demo sobre DE1_SoC 73
4.5 Implementación en el FPGA con algoritmos en OpenCL de PS, FB y la FFT como filtros para la eliminación de ruido en las imágenes de HD. 79

4.6 Resultados con el FPGA y los algoritmos PS, FB y FFT escritos en C y OpenCL, tiempos de procesamiento. 86

Conclusiones y Trabajo Futuro 95

Bibiliografía 97

ANEXO 1. Óptica Física. 108

ANEXO 2. 124

ANEXO 3. GPU, FPGA y OPENCL. 152

Índice de figuras

Figura.2.1. Primer holograma de D. Gabor [17]……..……..…..10

Figura.2.2. Primeros hologramas de un objeto tridimensional por Leith y Upatnieks [18]……..……..…….10

Figura.2.3. Imagen computada digitalmente del holograma detectado electrónicamente, por J. Goodman [23]…….……..….11

Figura.2.4. Imagen de contraste de fase obtenida con un objeto de fase pura, por C. Depeursinge et al [27]…….……..……12

Figura.2.5. a) Registro, b) Reconstrucción con la onda de referencia E_R, c) Reconstrucción con la onda conjugada de referencia E_R^* …….……15

Figura.2.6. Sistema de coordenadas para la reconstrucción de hologramas numéricos ..…….15

Figura.2.7. Reconstrucción de la imagen virtual……..……...16

Figura.2.8. Holografía digital de desplazamiento de fase con interferómetro de Michelson. Expansor de haz BX, divisor de haz BS, espejo de referencia PZT piezoeléctrico……..……..…….…. 18

Figura.2.9. Cambio de fase del microscopio holográfico digital……..……21

Figura.2.10 Proceso de filtrado…….……..……..…..22

Figura.2.11 Un vecindario 3 x 3 alrededor de un punto (x, y) en una imagen…..23

Figura.2.12 Operación mariposa……..……..……..…..26

Figura.2.13. Filtro de transformada de Fourier [53]……..……27

Figura.2.14. Brecha entre PLDs y ASICs……..……..……....28

Figura.2.15. Los elementos clave que forman un simple bloque lógico programable ..…….…….....29

Figura.2.16. Vista de arriba hacia abajo de la arquitectura de FPGA ……..... 30

Figura.2.17. Función requerida y tabla de verdad asociada……...31

Figura.2.18. Un CLB que contiene cuatro slice (el número de slice depende de la familia FPGA)……..……..……..……....32

Figura.2.19. Ilustración un chip con columnas de bloques RAM incrustados…..33

Figura.2.20. Un chip con columnas de multiplicadores incrustados y bloques RAM …….……..34

Figura.2.21. a) Un chip con núcleo incrustado fuera de la estructura principal, b) Uno o múltiples núcleos……..…..…….……35

Figura.2.22. a) Un árbol de reloj simple, b) Un administrador de reloj genera relojes hijos…….……..……..……..……..……..36

Figura.2.23. Un chip que muestra bancos de E/S de propósito general…..……38

Figura.2.24. Arquitectura FPGA Altera [62]……..……..….39

Figura.2.25. El modelo de plataforma OpenCL con un host y uno o más dispositivos OpenCL. …….……..…..43

Figura.2.26. Un ejemplo de cómo los identificadores globales (global IDs), ID locales (local IDs) e índices de grupos de trabajo están relacionados para un NDRange bidimensional…….45

Figura.2.27. Modelo de memoria en OpenCL y cómo las diferentes regiones de memoria interactúan con el modelo de plataforma……..…..……49

Figura.3.1. Láser Helio-Neón 633nm [69]……..……..……...….52

Figura.3.2. Objetivo de microscopio[70]……..…….……..……53

Figura.3.3. Divisor de haz 400-700nm, 50/50[71]…….……...54

Figura 3.4. Espejo [71]……..……...…….54

Figura.3.5. Lente biconvexa [71] ……..……..…….…..55

Figura.3.6. Cámara CCD [72]……...…….……56

Figura.3.7. Diagrama de bloques del DE1_SoC……..……57

Figura.3.8. Arreglo experimental para el método de PSDHM……..……..…..….60

Figura.3.9. Hologramas obtenidos con desplazamiento de fase cada π/2 ……..61

Figura.3.10. Imágenes de prueba utilizadas de 256x256 pixeles……..……..63

Figura.3.11. a) Fase Envuelta completa, b) Perfil de la imagen y c) Localización de saltos ..……...63

Figura 3.12. Algoritmo de desenvolvimiento implementado en Matlab®…..…..64

Figura.3.13. Imagen del algoritmo PS sobre las filas de la fase envuelta, sin discontinuidades en 2π para las filas……..……..……..…….64

Figura.3.14. Fase Desenvuelta…….……..…..…….65

Figura 3.15 Algoritmo función Mesh ()……...…..65

Figura.3.16. Comparación de los datos e imágenes de fase, después de aplicar la función Unwrap ()..…..……..……..….66

Figura 3.17. M. Visual Studio 2010……..…….……67

Figura.4.1. Fase desenvuelta del polímero sin compensación, a) Fase del objeto y b)Fase de referencia……..……..…….68

Figura.4.2. Fase desenvuelta muestra de Aluminio con $NaCl_2$, a)Fase sin compensación, b)Fase con compensación…….…..…..……69

Figura.4.3. Imágenes de prueba PS……..……..…..70

Figura.4.4. Resultados evaluación del algoritmo PS……..……70

Figura.4.5. Resultado de algoritmo PS en MVS……..……..…….……70

Figura.4.6. Arquitectura Altera SoC OpenCL [77]…….……..……73

Figura.4.7. Prueba Hello_World……..……..……74

Figura.4.8. Interfaz de edición para Visual Studio 2010…….……...75

Figura.4.9. Filtro bilateral. a) Imagen Lena sin filtro, b) Lena con Filtro……76

Figura.4.10. Filtro FFT, a) Pasa-Bajo y b) Pasa-Alto…….……76

Figura.4.11. Reporte de utilización de los Recursos Lógicos Programables del DE1_SoC del FB……..……..……77

Figura.4.12. Reporte de utilización de los Recursos Lógicos Programables del DE1_SoC para la FFT…….……..……..……..……..78

Figura.4.13. Secuencia de filtrado…….…..……82

Figura.4.14. Archivos *aocx generados……..……..….84

Figura.4.15. Tiempo resultante para implementar el FB en el FPGA……..…...85

Figura.4.16. Tiempo en la implementacion de la FFT en el FPGA…..……..…85

Figura 4.17 Reporte RLP del algoritmo PS……..….86

Figura.4.18. Conjuntos de imágenes desplazadas π / 2, 24 bits RGB obtenidas por PSDHM; Los grupos 1 y 3 corresponden a la película de polímero, el grupo 2 son interferogramas de referencia y el grupo 4 corresponde a microesferas...87

Figura 4.19. Gráficas 2D y de superficie 3D en Matlab® representando las distribuciones de fase desenvueltas. Película de polímero……..……..90

Figura 4.20. Gráficas 2D y de superficie 3D en Matlab® representando las distribuciones de fase desenvueltas para el proceso 4. Esferas microcristalinas.....91

Figura 4.21. Gráficas 2D y de superficie 3D en Matlab® representando las distribuciones de fase desenvueltas para la película de polímero. Procesos 5: (a) 1 iteración, (b) 3 iteraciones. Proceso 6: (c) 1 iteración…..…….94

Índice de tablas

Tabla 1. Pruebas de imágenes con desplazamiento de fase. ……70

Tabla 2. Implementación en FPGA. ……..……..……...……71

Tabla 3. Requerimientos. Requerimientos de la tarjeta……..……..……..…72

Tabla 4. Resultados Filtros……..…...……...……79

Tabla 5. Resultados primeras pruebas en el DE1_SoC. ……..……79

Tabla 6.Grupo de imágenes de Prueba. ……..……..……..……80

Tabla 7. Resultados en la implementación del Filtro Bilateral en C y OpenCL para MVS10 en PC……..……..……81

Tabla 8. Implementación de la FFT en MVS210. ……..……82

Tabla 9. Filtro Fourier en MVS2010 en PC…..……..……..……83

Tabla 10. Archivos *.aocx, del Filtro bilateral y Transformada Rápida de Fourier ……..……..…….……...83

Tabla 11. Tiempos de desenvolvimiento de fase obtenidos para los proceso 1 y representación de la fase desenvuelta……..……..……88

Tabla 12. Tiempos de desenvolvimiento de fase obtenidos en los proceso 2 y representación de la fase desenvuelta……..……...…88

Tabla 13. Tiempos de desenvolvimiento de fase obtenidos en los proceso 3…89

Tabla 14. Tiempo de desenvolvimiento de fase, proceso 4……..……..…91

Tabla 15. Tiempos de desenvolvimiento de fase obtenidos en los procesos 5 y 6, como se describe en el texto…….94

Resumen

La Microscopía Holográfica Digital (DHM) es utilizada con frecuencia para el registro de la amplitud y la fase por ejemplo de imágenes con profundidad de foco de especímenes biológicos, permitiendo obtener información cualitativa y cuantitativa de ellos, por lo que es un método muy adecuado para la observación de procesos biológicos dinámicos además de que compite con otros métodos tales como la Microscopía de Contraste de Fase (PCM), la Tomografía Óptica Coherente (OCT), o la Microscopía de Fuerza Atómica (AFM), entre otros. La caracterización y análisis de las células de sangre humana permite realizar diagnósticos clínicos más certeros para la prevención de enfermedades infecciosas, así como el estudio sobre los cambios de los glóbulos rojos humanos [1], que entre otras características proporcionan su dimensión y estados de deformación; asimismo, los métodos de DHM se han venido utilizando también como alternativas para la visualización, monitoreo y registro de procesos en tiempo real en varias áreas de la ingeniería tales como la topografía tridimensional (3D) de superficies, aplicaciones de resolución temporal, MEMS, metrología, inspección de procesos industriales, micro-óptica o seguimiento de partículas 3D [2], entre otras.

Un problema frecuente en DHM es la optimización de las operaciones computacionales de los algoritmos de filtrado de imágenes y desenvolvimiento de la fase óptica de los hologramas digitales registrados durante un proceso. En ocasiones, las características de los procesadores de los equipos de cómputo disponibles comercialmente limitan notoriamente el desempeño de los algoritmos para realizar sus tareas con una mayor eficiencia, ya sea por su tamaño o su consumo de energía; por estas razones en los últimos años se han venido incorporando diversos sistemas hardware para un mejor aprovechamiento de los mismos.

Por lo que las matrices de puertas programables por campo (FPGA) y los sistemas en chip (SoC), han sido aplicados durante más de 20 años en diferentes áreas tales como cómputo de alto desempeño, comunicaciones, procesamiento de señales de imágenes y video, minería de datos, automatización, visión por computadora, industria aeroespacial, defensa, medicina, bioinformática, etc. [3]. Estos dispositivos tienen ventajas en comparación con otros dispositivos de hardware como un bajo consumo de energía o un bajo costo con respecto a un microprocesador o un GPU, además de que estos pueden incorporar procesadores digitales de señales (DSP) o RAM embebida entre otras herramientas; también pueden ser programados con lenguajes de bajo nivel como VHDL o alto nivel como el lenguaje de computo abierto (OpenCL) que entre otros es una (API) que proporciona procesamiento en paralelo de tareas y datos, es compatible con una amplia gama de niveles de paralelismo y dispositivos de hardware, y es capaz de aumentar la eficiencia de sistemas homogéneos o heterogéneos de uno o varios dispositivos que utilizan CPU, GPU o FPGA, por lo que los FPGA presentan una seria opción para el procesamiento en DHM.

El trabajo desarrollado en esta tesis se basa en la implementación de algoritmos de filtrado y desenvolvimiento de fase óptica de imágenes holográficas digitales en un dispositivo SoC FPGA, así como la implementación de un sistema experimental de microscopía holográfica digital mediante desplazamiento de fase (PSDHM) para el análisis 3D de muestras metálicas y transparentes. Para la programación y pruebas iniciales de los algoritmos se utilizó la plataforma Matlab® y Visual Studio 2010 y posteriormente OpenCL. Primeramente se implementaron dos algoritmos simples: uno de desplazamiento de fase de cuatro pasos para obtener mapas de distribución de fase, y otro de desenvolvimiento de la fase (PS) para representar la topografía 3D de las superficies de las muestras; una vez evaluados, se implementaron en el FPGA. Las imágenes obtenidas presentaban discontinuidades asociadas con el ruido, para obtener una mejor reconstrucción, se implementaron dos algoritmos más de filtrado: el Filtro Bilateral (BF) y otro filtro por Transformada Rápida de Fourier (FFT).

Los algoritmos de desenvolvimiento, BF y FFT en código OpenCL, se implementaron en la tarjeta DE1_SoC de Altera, los resultado obtenidos se utilizan para realizar un análisis comparativo de los diferentes tiempos de procesamiento en el procesador ARM y el FPGA, finalmente se discuten las ventajas y desventajas de los datos obtenidos.

Abstract

Digital Holography Microscopy (DHM) is frequently used to record both amplitude and phase of depth-of-focus images of biological specimens which allows qualitative and quantitative information acquisition, making it a suitable method for observation of dynamic biological processes as it competes with similar tecniques such as Phase Contrast Microscopy (PCM), Optical Coherent Tomography (OCT) or Atomic Force Microscopy (AFM), among others. As an example, characterization and analysis of human blood cells allows a more accurate clinical diagnose for of infectious diseases prevention, as well as the study on the changes of human red blood cells[1], that among other characteristics also provide their dimension and deformation states. Additionally, DHM methods also have been used as alternatives for visualization, monitoring and recording of real-time processes in several areas of engineering such as surface three-dimensional (3D) topography, time-resolved applications, MEMS, metrology, industrial inspection, micro-optics or 3D particle tracking[2], among others.

A frequent problem in DHM is the optimization of computational operations involved in algorithms for image filtering and optical phase unwrapping of digital holograms recorded during a particular process. Usually, commercial computing processors characteristics limit significantly the algorithms to perform their tasks with higher efficiency; for these reasons, various hardware systems have been recently incorporating in computing systems to improve them. Field-programmable gate arrays (FPGA) and systems in chip (SoC) have been applied during more than 20 years in different areas such as high performance computing, video and image processing, communications, digital signal processing, data mining systems, automotive, computer vision, aerospace and defense, medical, bioinformatics, etc.[3]. These devices have advantages compared to other hardware, such as low power consumption and low cost with respect to a microprocessor or a GPU; in addition, they allow the incorporation of digital signal processors (DSP) or embedded RAM among other tools.

They can also be programmed with high-level languages such as OpenCL. This is an application programming interface (API) that provides parallel processing of tasks and data, supports a wide range of levels of parallelism and hardware devices, and is capable to increase the efficiency of homogeneous or heterogeneous systems of one or more devices that use CPU, GPU or FPGA, so FPGA present a factible option for processing in DHM.

The work developed in this thesis is based on implementation of algorithms for filtering and optical phase unwrapping of digital holographic images in a SoC FPGA device, as well as the implementation of an experimental system of phase-shifting digital holographic microscopy (PSDHM) for 3D analysis of metallic samples surfaces. For initial programming and testing of algorithms, Matlab® platform was used; later, Visual Studio 2010 (for C programming) and finally OpenCL were also used. Two simple algorithms were firstly implemented: one with a four-step phase shifting method for obtaining of holograms and phase distribution maps, and another for phase unwrapping (PS) to perform a 3D topography of sample surfaces; once were tested, algorithms were implemented in the FPGA circuit. In order to filter the image noise observed in the holograms and their corresponding 3D mesh plots obtained during phase unwrapping (i.e. topographic discontinuities) and to improved their reconstructions, two more algorithms were implemented for filtering: the Bilateral filter and another for a Fast Fourier Transform (FFT) filter, both in the spatial and frequency domains. The obtained images presented discontinuities associated with noise, to obtain a better reconstruction, two more filtering algorithms were implemented: the Bilateral Filter (BF) and another filter by Fast Fourier Transform (FFT). The development algorithms, BF and FFT in OpenCL code, were implemented in the Altera card DE1_SoC, the results obtained are used to perform a comparative analysis of the different processing times in the ARM processor and the FPGA, finally the advantages are discussed and disadvantages of the data obtained.

Motivación

Los FPGA son dispositivos que actualmente se utilizan para un amplio rango de aplicaciones, realización de tareas específicas debido a sus notables características y ventajas, en aplicaciones como comunicaciones, procesamiento de imágenes y video, redes neuronales y procesamiento digital de señales, imágenes [4,5], entre otros, haciendo de estos dispositivos una buena opción para la obtención y procesamiento de información visual 3D, obtenidas por ejemplo por medio de métodos ópticos como los interferométricos. La DHM permite obtener imágenes que contienen información tanto de la amplitud como de la fase de un objeto de interés con profundidad de foco, lo que permite obtener información acerca de sus propiedades morfológicas y ópticas tales como el tamaño o distribución de su índice de refracción, entre otras, por lo que resulta de gran interés mostrar el potencial de los FPGA en el desarrollo de algoritmos para la aceleración de proceso de desenvolvimiento de fase y la implementación de algoritmos de filtrado aplicados en Microscopía Holográfica Digital mediante Desplazamiento de Fase (PSDHM).

Objetivo

Se propone el desarrollo e implementación de algoritmos para el desenvolvimiento de fase óptica, Filtro Bilateral y de Transformada de Fourier para el procesamiento y mejora de información de imágenes holográficas obtenidas por métodos de PSDHM en arquitecturas de hardware FPGA sobre una tarjeta Altera DE1_SoC, así como investigar métodos para la aceleración de los tiempos de obtención de datos mediante programación en OpenCL, y demostrar el potencial de la utilización de estos dispositivos como una alternativa más eficiente y económica.

Objetivos específicos:

- Desarrollo e implementación de algoritmos para la adquisición y procesamiento de imágenes mediante arquitecturas de hardware FPGA en PSDHM.
- Investigar los métodos para la aceleración de procesos mediante OpenCL.
- Demostrar el potencial de la utilización de estos dispositivos como una alternativa eficiente, económica y de bajo consumo de energía.

Justificación

Los FPGA se están convirtiendo en parte de la mayoría de los diseños integrados actuales. Su naturaleza reconfigurable es beneficiosa para acortar sus tiempos de salida al mercado. Existe también una tendencia de que los FPGA reemplacen a los DSP y ASIC por una variedad de razones; los FPGA están aumentando la disponibilidad de los procesadores integrados y los recursos DSP dedicados, que los aparta para ser utilizados solamente en la lógica de pegamento (glue logic) y como componente lógico principal de un sistema integrado. Los chips FPGA son circuitos que se pueden programar dinámicamente para aplicaciones o funcionalidades. En la actualidad, son fabricados por varias compañías en muchas configuraciones. El mercado de estos dispositivos representa una distribución anual de millones de chips y más de 4 mil millones de dólares en ventas tan solo en 2016.

Los FPGA fueron capaces de superar a las GPU en diversos grados de velocidad y/o eficiencia energética. Algunas tareas fueron solo un 50% más rápidas, mientras que otras fueron 440% más rápidas y otras aún más, pero con un 130% de mejora en términos de rendimiento por vatio (el calor a menudo es un factor limitante, por lo que el rendimiento por vatio puede ser crítico) [6].

Actualmente los FPGA se han utilizado y previamente en interferometría para reconstrucción rápida de imágenes 3D de hologramas digitales, un ejemplo es la utilización de la transformada de Fresnel, así como entre otras aplicaciones en holografía [7]. De acuerdo a sus características particulares las cuales se comentan en esta tesis, y evaluando el desempeño de un FPGA de Altera DE1_SoC en tareas de desenvolvimiento de fase y filtrado, mostrando su utilidad como una herramienta eficiente mediante la implementación de algoritmos de alto nivel, para realizar PSDHM.

El avance en la tecnología está evolucionando continuamente, las herramientas de procesamiento no son una excepción en la que busca aumentar el rendimiento, reducir la potencia de consumo, así como reducir el costo de los dispositivos y sistemas, la combinación de un FPGA y CPU en un SoC configurable, indican las soluciones de innovación y con el uso de software que trabaje en conjunto para acelerar los procesos como los conjuntos de herramientas de FPGA modernos incluyen la compilación de síntesis de alto nivel de algoritmos en código C, Cuda® y OpenCL a lógica o a microprocesadores integrados , la tendencia hacia el rendimiento de los sistemas a partir del paralelismo es una realidad, como se puede observar.

La ley de Moore[8] es la norma que durante cinco décadas ha marcado el destino de los procesadores y chips que llevan los equipos tecnológicos, es decir que con el paso del tiempo tiende a multiplicar el rendimiento y dividir su coste según Simon Viñals director de Tecnología de Intel indica que para el 2020 se contaran con procesadores con tecnología de 7nm[9], el reto de Intel es crear capas tridimensionales de transistores interconectados entre sí, así mismo vemos que otra tecnología clave en el futuro podrían ser los FPGA por su reconfigurabilidad en diversos campos de la industria, inteligencia artificial y biología entre otros. Los FPGAs Intel eliminan los cuellos de botella y aceleran el mundo inteligente y conectado al ofrecer mayor flexibilidad, más inteligencia y mayor eficiencia. Como aceleradores de algoritmos multifunciones, FPGAs y SoC FPGAs proporcionan la combinación ideal de configuración y programación, de hardware y software. Esto permite que los diseñadores de sistemas creen mejores sistemas explorando cuales cargas de trabajo trabajan más eficientemente en los dominios de CPUs y FPGAs.

Desde la relativamente reciente adquisición Intel de Altera por parte del gigante de chips Intel [10], hasta avances menos comentados en el frente de programación (progreso de OpenCL, avances en hardware y software de competidores en FPGA como Intel/Altera, Xilinx) y, por supuesto, competencia consistente para el mercado de aceleración de cómputo de las GPU, que dominan el mercado de coprocesadores.

Por ahora, se puede vislumbrar una de las muchas maneras en que los FPGA podrían encajar en el ecosistema de hiperescala (junto con otras perspectivas de hardware futuro) con un anuncio de que Intel trabajará en futuros diseños con un FPGA integrado y Chip Xeon [11]. Un chip Broadwell de 15 núcleos con los FPGA Altera Arria 10 GX podría ser la próxima generación de un FPGA y CPU.

CAPÍTULO 1

Introducción y antecedentes

1.1 Los FPGA.

Los avances de la tecnología digital durante las últimas décadas han sido a pasos realmente grandes, mencionando algunos ejemplos, los automóviles han pasado con los años de tener muy pocos controles electrónicos a ser vehículos controlados en su mayor parte por tecnología digital, algunos de ellos equipados con sistemas de navegación, comunicación e información inalámbrica; el audio digital nos llevó a utilizar discos compactos y hasta reproductores de mp3, el video digital al DVD, Blu-Ray, adelantos tanto en cámaras de video y fotográficas digitales para entornos profesional y del hogar, teléfonos celulares, así como la revolución de la televisión digital que provee una mayor definición de la imagen y flexibilidad en programación.

Como sabemos existen muchos campos en que la tecnología digital ha incursionado desde la salud, los servicios, recreación, tecnologías de la información y las comunicaciones con el gran crecimiento del internet y las redes sociales entre otros, las herramientas de software para desarrollar sistemas complejos, dispositivos de comunicaciones o las estaciones de juego interactivo, no son más que algunos ejemplos de los cambios de nuestro estilo de vida haciéndolo cada vez más confortable.

La mayoría de estos sistemas tienen algo en común: su tamaño, con dimensiones tan pequeñas que es impresionante pensar que sean igual o más potentes que los sistemas mucho más grandes que existieron hace algunos años.

Estos avances son posibles gracias al desarrollo de la microelectrónica, la cual ha permitido la miniaturización de los componentes para obtener así mayores beneficios de los chips (circuitos integrados) y para ampliar las posibilidades de aplicación.

Primero se desarrollaron los circuitos de baja escala de integración (SSI o Small Scale Integration), después los de mediana escala de integración (MSI o Médium Scale Integration) y posteriormente los de muy alta escala de integración (VLSI o Very Large Scale Integration) hasta llegar a los circuitos integrados de propósito específico (ASIC).

En el mundo de la informática y la electrónica, se acostumbra a dos formas diferentes de realizar cálculos: hardware y software. El hardware de computadora, como los circuitos integrados de aplicación específica (ASIC), proporciona una alta optimización de recursos para realizar rápidamente tareas críticas, pero está configurado permanentemente a una sola aplicación a través de un esfuerzo de diseño y fabricación multimillonario.

Los entornos informáticos de hoy son cada vez más multifacéticos, explotando capacidades de una gama de microprocesadores multi-core, unidades de procesamiento central (CPU), procesadores de señal digital, hardware reconfigurable (FPGA) y unidades de procesamiento gráfico (GPU), presentado con tanta heterogeneidad, el proceso de desarrollo software eficiente para una gama tan amplia de arquitecturas plantea una serie de desafíos a la comunidad de programación [12].

Los sistemas informáticos heterogéneos también agregan riqueza al permitir que el programador seleccione la mejor arquitectura para ejecutar la tarea o elegir la tarea correcta para hacer un uso óptimo de una arquitectura determinada. Recientemente, ha habido un aumento en la comunidad de diseño de computadoras experimentando con la construcción de sistemas heterogéneos. Estamos viendo nuevos sistemas en el mercado que combinan diferentes clases de arquitecturas. Lo que ha retrasado esta progresión ha sido la falta de un entorno de programación estandarizado que pueda gestionar el conjunto diverso de recursos en un marco común.La necesidad de una informática heterogénea está conduciendo a nuevos lenguajes de programación para explotar el nuevo hardware.

Un ejemplo es OpenCL, que ha sido desarrollado específicamente para aliviar la carga de programación al escribir aplicaciones para sistemas heterogéneos.

OpenCL también aborda la tendencia actual de aumentar el número de núcleos en una arquitectura determinada, admite la ejecución en CPU, GPU y FPGA, así también en procesadores digitales de señales (DSP) y unidades de procesamiento acelerado (APU) heterogéneas. Las arquitecturas ya admitidas cubren una amplia gama de enfoques para extraer el paralelismo y la eficiencia de los sistemas de memoria y las secuencias de instrucciones.

La interfaz estándar de OpenCL le permiten al programador "unir" sin problemas una aplicación dentro de la cual la ejecución puede ocurrir en un amplio conjunto de dispositivos heterogéneos de uno o varios fabricantes. Las aplicaciones OpenCL pueden dirigirse a varios dispositivos de hardware a la vez, y estos dispositivos no tienen que tener la misma arquitectura o incluso el mismo proveedor. Por lo tanto tenemos que un OpenCL estándar puede definir un conjunto de tipos hardware, de datos y de estructuras de datos. Aunque se tienen otros entornos OpenCL para Java y Python, el estándar solo requiere que OpenCL proporcione bibliotecas en C y C ++. Podemos ver que en algunos trabajos en OpenCL se ha implementado para evaluar hardware con eficiencia a partir de un FPGA [13] y para comparar los resultados que en el caso de un CPU y GPU en la implementación de algoritmos de filtrado [14].

Por otro lado el software de computadora proporciona la flexibilidad para cambiar aplicaciones y realizar una gran cantidad de tareas diferentes. Los FPGA son dispositivos verdaderamente revolucionarios que combinan los beneficios del hardware y el software. Implementan circuitos al igual que el hardware, proporcionando enormes beneficios de potencia, área y rendimiento sobre software, pero puede reprogramarse de manera económica y sencilla para implementar un amplio rango de tareas, por ejemplo aplicaciones en comunicaciones como transmisión óptica, comunicación inalámbrica, aceleradores de servidores, exploración de petróleo y gas, biociencia, comunicaciones seguras, aviónica y sistemas de guía, radares de próxima generación entre otras [15].

Al igual que el hardware de la computadora, los FPGA implementan cálculos espacialmente, computando simultáneamente millones de operaciones en recursos distribuidos a través de un chip de silicio. Estos sistemas pueden ser cientos de veces más rápidos que los diseños basados en microprocesadores. A diferencia de los ASIC, los cálculos están programados en el chip, no fijados permanentemente por el proceso de fabricación. Esto significa que un sistema basado en estas puertas lógicas programables puede ser programado (configurado) y reprogramado muchas veces.

Estos dispositivos brindan casi todos los beneficios de la flexibilidad y el desarrollo de modelos software, y casi todos los beneficios de la eficiencia del hardware. En comparación con un microprocesador, son generalmente varios órdenes de magnitud más rápidos y eficientes en energía, pero la creación de programas eficientes para ellos son más complejos. Normalmente son útiles solo para las operaciones que procesan grandes flujos de datos, como procesamiento de señales, redes y similares; en comparación un diseño ASIC puede tomar meses o años para desarrollar y tener una aplicación a un alto costo, un diseño de FPGA solo podría tomar días para crear y costar de decenas a cientos de dólares. Para sistemas que no requieren el más alto rendimiento y la mayor eficiencia energética, la simplicidad de su desarrollo y la capacidad de corregir errores fácilmente y actualizar la funcionalidad los convierte en una alternativa de diseño convincente.

El FPGA Intel® SoC fue diseñado para dar solución a una amplia gama de aplicaciones. El SoC FPGA combina un procesador, periféricos y FPGA en un único dispositivo personalizable por el usuario, incluye versiones SoC como el popular Cyclone® V de 28 nm y Familias Arria® FPGA. Basado en la tecnología de proceso de 20 nm, por un lado Arria 10 SoC ofrece una actualización de rendimiento para usuarios Arria V SoC y agrega características de seguridad mejoradas. El Stratix® 10 SoC ofrece a los usuarios lo último en rendimiento con un Procesador ARM Cortex ® - A53 de cuatro núcleos basado en el liderazgo de Intel, con tecnología de proceso de silicio Tri-Gate (FinFET) de borde 14 nm.

Independientemente de cuál se utilice, los FPGA SoC cumplen con los requisitos del mercado cambiantes y los estándares de interfaz, mayor interconexión de ancho de banda con una interfaz de ancho de banda pico de más de 125 Gbps (en el Arria V SoC) (HPS) a FPGA y un alto ancho de banda de la interfaz FPGA a SDRAM. El SoC cuenta con líneas de productos de gama alta, gama media y bajo costo, basados en tecnologías de proceso de vanguardia de TSMC (28 nm y 20 nm) e Intel (Tri-Gate de 14 nm) con una vida útil de cada dispositivo superior a los 20 años [16].

1.2 Holografía digital.

Dennis Gabor inventó la holografía óptica en 1948 como un método para registrar y reconstruir la amplitud y la fase de un campo de ondas, buscando solucionar la aberración esférica presente en las lentes magnéticas del microscopio electrónico [17]. Una imagen holográfica u holograma es un patrón de interferencia registrado ya sea fotográficamente, en algún medio de fase o digitalmente como resultado de la superposición entre un campo de onda dispersado desde el objeto y un fondo coherente llamado onda de referencia. Generalmente se registra en una superficie plana, pero contiene la información de todo el campo de una onda tridimensional.

En holografía convencional, la onda objeto suele reconstruirse al iluminar nuevamente el holograma con la onda de referencia. Los avances en la informática permitieron transferir el proceso de reconstrucción a una operación computacional. Un gran paso fue el desarrollo de la grabación directa de hologramas de Fresnel con dispositivos de carga acoplada (CCD) por Ulf Schnars y Werner Jüptner [18], introduciendo así la Holografía Digital (DH); Schnars y Jüptner aplicaron la DH a la interferometría y demostraron que la reconstrucción del holograma digital ofrece muchas más posibilidades que el procesamiento óptico convencional.

Desde mediados de los años noventa del siglo pasado, la DH se ha ampliado, mejorado y aplicado a varias tareas de medición, algunas de las más importantes son mejoras de las técnicas experimentales y del algoritmo de reconstrucción, aplicaciones en análisis de deformación y medición de forma, aplicaciones en imágenes, seguimiento de partículas y microscopía entre otras.

Los hologramas pueden almacenar no solo imágenes tridimensionales de objetos, sino también cualquier tipo de datos codificados correctamente. El acceso directo a la fase y a los perfiles de amplitud en la microscopía hace que la DH mediante mediante microscopía cuantitativa de fase (DH-QPM) sea especialmente potente y versátil.

Por otra parte en DH aplicando desplazamiento de fase también se ha aplicado a la microscopía, utilizando dos tipos principales de interferómetros para realizar estos procedimientos que aún continúan siendo utilizados, como el Michelson y Mach-Zehnder.

El interferómetro de Michelson es apropiado para objetos reflectantes, aunque también es posible organizar la transmisión de doble trayectoria colocando una muestra transparente en un espejo. El interferómetro Mach-Zehnder es más adecuado para objetos transparentes. La microscopía holográfica digital (DHM) es una forma especial de microscopía utilizada ampliamente en campo de la biología para observar células y tejidos vivos de forma no intrusiva en su entorno. Algunos trabajos previos reportan algoritmos implementados en sistemas PC convencionales sobre reconstrucción en DHM implementados en GPU para mejorar el rendimiento en la conexión con distintos sistemas de hardware, referencias [19, 20].

En este trabajo de tesis, se presenta la implementación de un algoritmo sencillo de desenvolvimiento de fase programado en OpenCL para la reconstrucción de imágenes usando un circuito FPGA, y aplicado para microscopía holográfica digital mediante desplazamiento de fase (PSDHM). Los procedimientos de desenvolvimiento de fase implementados en algoritmos OpenCL se realizaron en imágenes holográficas de 1024x1024, 512x512 y 256x256 pixeles obtenidas por PSDHM. Además, los procesos OpenCL para filtrado por Transformada de Fourier Rápida (FFT) y filtrado Bilateral (BF) también se implementaron en el FPGA para disminuir el ruido y mejorar la calidad de las imágenes.

Se llevó a cabo una evaluación de estos algoritmos mediante una comparación de sus tiempos de procesamiento, en filtrado, realizados independientemente en un procesador ARM y el circuito de FPGA, al igual que sus correspondientes tiempos de desenvolvimiento de fase, asociadas a las superficies de algunas muestras metálicas y transparentes analizadas.

CAPÍTULO 2

Marco teórico
2.1 Introducción

La holografía digital es una tecnología emergente dentro del nuevo paradigma general de aplicaciones en reconstrucción de imágenes. Al reemplazar los procedimientos fotoquímicos convencionales con imágenes electrónicas, se abrió la puerta a una amplia gama de nuevas capacidades. En la holografía digital, el patrón de interferencia es generado ópticamente por la superposición de los haces objeto y de referencia, luego grabado digitalmente en una CCD, muestreado y transferido a una computadora como una matriz de números. La DH ofrece una serie de ventajas significativas, como la capacidad de adquirir hologramas rápidamente, disponibilidad de información completa de amplitud y fase del campo óptico, y la versatilidad del procesamiento interferométrico y de imágenes técnicas. La combinación de la microscopía convencional con holografía digital ofrece la posibilidad de lograr altas resoluciones espaciales y de medir la forma y/o deformaciones de los objetos en diferentes planos focales.

Como sabemos, con el continuo avance de la tecnología informática en el proceso de grabación y reconstrucción de hologramas utilizando PC, se ha avanzado tanto que ahora se investigan sistemas que usan más de un tipo de procesador, para aprovechar las capacidades de procesamiento especializado de la informática heterogénea. Antes de 2010, la computación de propósito general en unidades de procesamiento gráfico (GPGPU) se consideraba una novedad en el mundo de la computación de alto rendimiento y no merecía atención seria sobre todo porque se utilizaba para realizar rutinas no-gráficas.

Hoy en día hay principal atención a los sistemas que integran CPU-FPGA, y en esta tendencia de la industria se están impulsando a los llamados SoC FPGA a jugar un papel importante en el paradigma de la computación heterogénea, también sumando en conjunto con lenguajes de alto nivel como OpenCL también utilizado para programar FPGA en entornos heterogéneos.

2.2 Holografía

2.2.1 Holografía convencional (analógica)

La holografía de Gabor [17], en un esfuerzo por mejorar la resolución del microscopio electrónico, donde la corrección de las aberraciones de las lentes de electrones planteaba dificultades técnicas, en lugar de tratar de perfeccionar las lentes, Gabor la dispensó por completo, intuyendo que el patrón de difracción del haz de electrones contenía información completa sobre la amplitud y fase de la onda de electrones. El registro de la difracción de ondas de electrones lo usó para sintetizar ópticamente el campo de onda objeto, lo que permitió el uso de óptica de luz visible para la formación de las imágenes, que es una tarea mucho más fácil y desarrollada en comparación con la óptica electrónica. Nombró al nuevo principio de imagenología como “holografía” (del griego *holos*-todo, *graphos*-grabado), por su capacidad para registrar toda la óptica de campo.

Aunque Gabor realizó los experimentos para registrar y reconstruir ópticamente las imágenes, la falta de una fuente de iluminación suficientemente coherente impidió un mayor progreso de la técnica (Figura 2.1, Arriba: el holograma; abajo a la izquierda: el objeto; abajo a la derecha: la imagen reconstruida). Durante la década de 1950, el principio de holografía se aplicó principalmente a electrones y rayos X (microscopía de difracción), pero dos importantes invenciones desencadenaron un crecimiento verdaderamente explosivo de la holografía óptica, así como de sus técnicas y aplicaciones.

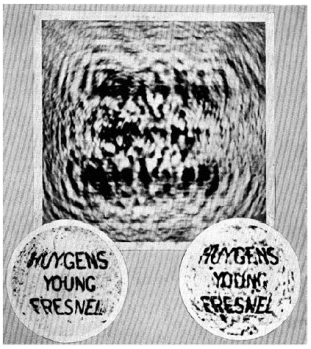


Figura 2.1 Primer holograma de D. Gabor
Referencia [17].

Una fue el láser, poderosa fuente coherente de luz, capaz de proporcionar un contraste de interferencia de alta calidad. La otra fue la iluminación fuera de eje, de Emmett Leith y Juris Upatnieks [21], donde se utilizó una onda de referencia separada, eliminando el problema del orden cero y las imágenes gemelas superpuestas de la configuración en eje de Gabor. Se había podido lograr entonces la reconstrucción de dos tipos de objetos que no son directamente posibles con el método de Gabor: objetos que no transmiten una onda de fondo fuerte (por ejemplo, letras transparentes sobre un fondo oscuro) y objetos de tono continuo. En 1964, Leith y Upatnieks demostraron la reconstrucción holográfica de objetos sólidos tridimensionales (figura 2.2), que se parecen en gran medida a los objetos originales, ya que exhiben la propiedad de paralaje entre objetos cercanos y más distantes [22].

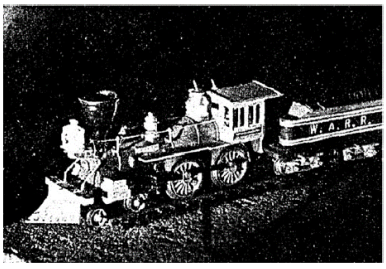


Figura 2.2 Primeros hologramas tridimensionales de Leith y Upatnieks
Referencia [18].

2.2.2 Holografía digital

La propagación del campo óptico se describe de forma completa y precisa mediante la teoría de difracción y, por lo tanto, es susceptible de computación numérica. La demostración de la viabilidad de la reconstrucción numérica de un holograma fue realizada por Joseph Goodman *et al* en 1967 [23].

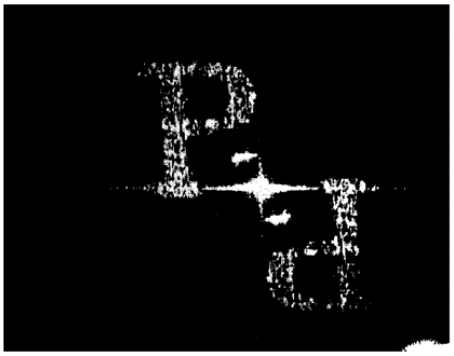


Figura 2.3 Imagen computada digitalmente del holograma detectado electrónicamente, por J. Goodman
Referencia [23].

Otro precursor de la holografía digital fue Haddad *et al*, con un microscopio holográfico [24], utilizaron una pequeña gota de glicerol como lente para crear la iluminación de referencia esférica divergente necesaria para holografía con la transformada rápida de Fourier; también incorporaron una CCD con un área sensible de 2048x2048 píxeles y tamaño de píxel de 9µm. Los cálculos de la transformada de Fourier en una PC produjeron imágenes micro-holográficas de una sección de un parásito *ascaris*. El procesamiento de una lente numérica logró enfoques numéricos a diferentes distancias focales.

En la actualidad, las cámaras CCD y las tecnologías informáticas se han desarrollado a un nivel suficiente para la implementación práctica de la DH, también existen otros métodos de obtención previos de aplicación de algoritmos para obtener hologramas generados por computadora [25,26].

En 1999, el grupo de Christian Depeursinge (E'cola Polytechnique Fe'de'rale de Lausanne, Suiza) introdujo el método de microscopía de contraste de fase cuantitativa (QPCM) mediante DH, que produce directamente un perfil de superficie con menos de unos pocos nanómetros de ruido efectivo (figura 2.4) [27].

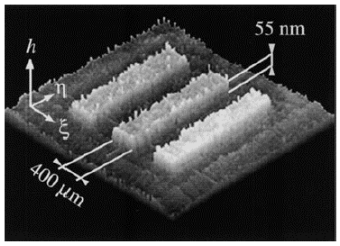


Figura 2.4 Imagen de contraste de fase obtenida con un objeto de fase pura, por C. Depeursinge et al.
Referencia [27].

2.2.3 Referencias de implementación

En DHM un solo holograma se utiliza para centrarse numéricamente en la imagen holográfica a cualquier distancia [28]. El acceso directo a la información de fase conduce a la QPCM con sensibilidad nanométrica de objetos de fase transparentes o reflectantes [29], y a la aplicación de la DHM en transmisión para la caracterización del espesor de recubrimientos delgados [30].

Desarrollos recientes en imágenes ópticas computacionales 3D tienen el comienzo de una nueva era para la investigación biológica [31], además que estas técnicas en 3D, la microscopía holográfica integrada con procesamiento numérico están permitiendo a los investigadores obtener información valiosa y cuantitativa sobre la estructura de las células y microorganismos con métodos no invasivos y en tiempo real.

Los resultados de las superficies caracterizadas y las células analizadas demuestran las aplicaciones de la DHM para la inspección técnica y obtención de imágenes de células vivas [32, 33].

Actualmente, el control de calidad de los procesos de producción para elementos semiconductores electrónicos es bastante costoso y lleva mucho tiempo, cuando se trata de investigar mediciones precisas para inspeccionar la producción correcta de las diferentes dimensiones y capas del componente eléctrico, las técnicas de medición comunes son la microscopía de fuerza atómica (AFM) y la microscopía electrónica de barrido (SEM).

El uso de técnicas de imágenes holográficas digitales puede ser una alternativa, para dar solución a estos dos tipos de medición con AFM y SEM. Con DH se puede obtener una imagen de un área completa con una sola toma y no se requiere exploración al combinarla con microscopía óptica [34]. Al combinar la holografía con microscopía digital en formato de video, se obtiene microscopía video-holográfica en línea, capaz de registrar imágenes holográficas en 3D de especímenes biológicos, que preserva la dimensión temporal [35].

El desenvolvimiento de fases ópticas de longitud de onda múltiple es un método rápido y robusto para eliminar discontinuidades en periodos cada 2π, en comparación con los métodos basados en algoritmos de software [36], con la supresión del orden cero y las imágenes gemelas mediante la holografía digital de desplazamiento de fase permite un uso eficiente de la matriz de pixeles [37]. Una de las principales áreas de aplicación de la DH en la metrología de deformaciones y vibraciones [38, 39]. En relación a aplicaciones de procesamiento óptico, como el reconocimiento de patrones y cifrado, la DH también ofrece nuevas capacidades y ventajas [40]. También, normalmente en la creación de Hologramas Digitales se requiere de un sistema interferómetro coherente sin embargo en [41] se utiliza un sistema de microscopia basado en Holografía de Fresnel de correlación incoherente y su uso en el registro de imágenes fluorescentes tridimensionales, también en [42], se demuestra que utilizando iluminación con fuentes parcialmente coherentes se mejora la calidad de imágenes holográficas.

2.2.4 Grabación y reconstrucción de hologramas digitales

El concepto de grabación de hologramas digitales se ilustra en la figura 2.5a. Una onda de referencia plana y la onda reflejada desde el objeto están interfiriendo en la superficie de un dispositivo acoplado cargado (CCD). El holograma resultante es electrónicamente grabado y almacenado. El objeto es en general un cuerpo tridimensional con superficie de reflexión difusa, ubicada a una distancia d del CCD.

En la reconstrucción óptica, la imagen virtual aparece en la posición del original objeto y la imagen real se forma a una distancia d también, pero en la dirección opuesta del CCD, ver figura 2.5b. La difracción de una onda de luz en una apertura (en este caso un holograma) que es montado en perpendicular al rayo entrante es descrita por la integral Fresnel- Kirchhoff [Anexo 1]

$$\Gamma(\xi', \eta') = \frac{i}{\lambda}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(x,y)E_R(x,y)\frac{exp\left(-i\frac{2\pi}{\lambda}\rho_\prime\right)}{\rho'}\,dxdy \qquad \text{Ec. 2.1,}$$

Con

$$\rho' = \sqrt{(x-\xi')^2 + (y,\eta')^2 + d^2} \qquad \text{E. 2.2;}$$

$h(x,y)$ es la función del holograma y ρ' es la distancia entre un punto en el plano del holograma y un punto en el plano de reconstrucción. Las cantidades geométricas son explicadas en la figura 2.6. El factor de inclinación es considerado como 1, por que los ángulos θ y θ' son aproximadamente iguales a 0. Esto es válido para todos los algoritmos de reconstrucción. Para el plano de la onda de referencia $E_R(x,y)$ es simplemente dado por la amplitud real.

$$E_R = a_R + i0 = a_R \qquad \text{Ec. 2.3}$$

El patrón de difracción se calcula a una distancia d detrás del plano CCD, lo que significa que reconstruye la amplitud compleja en el plano de la imagen real. Eq. (2.1) es la base para la reconstrucción numérica del holograma. Porque el campo de onda reconstruido $\Gamma(\xi', \eta')$ es una función compleja, tanto la intensidad como la fase se puede calcular [43].

Esto está en contraste con el caso del holograma óptico reconstrucción, en la que solo la intensidad se hace visible. Esta interesante propiedad de la Holografía Digital se utiliza en la Interferometría Holográfica Digital.

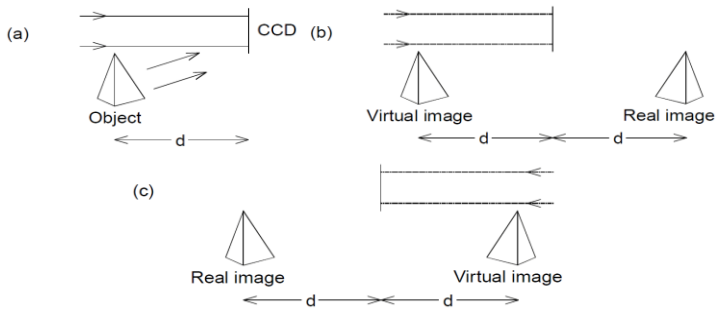


Figura 2.5 a) Registro, b) Reconstrucción con la onda de referencia E_R, c) Reconstrucción con la onda conjugada de referencia E_R^*

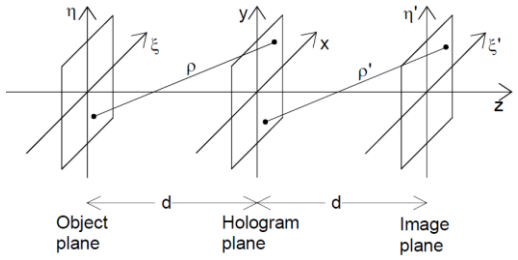


Figura 2.6 Sistema de coordenadas para la reconstrucción de hologramas numéricos.

La imagen real podría distorsionarse, se puede producir una imagen real no distorsionada utilizando el haz de referencia conjugado para la reconstrucción. Para reconstruir una imagen real sin distorsiones en Holografía digital por lo tanto, es necesario insertar E_R^* en vez de E_R en Eq. (2.1):

$$\Gamma(\xi,\eta) = \frac{i}{\lambda}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(x,y)E_R^*(x,y)\frac{exp\left(-i\frac{2\pi}{\lambda}\rho\right)}{\rho}\,dxdy \qquad \text{Ec. 2.4}$$

$$\rho = \sqrt{(x-\xi)^2 + (y,\eta)^2 + d^2} \qquad \text{Ec. 2.5}$$

Este esquema de reconstrucción se muestra en la figura 2.5c. La imagen real emerge en esa posición, donde se ubicó el objeto durante la grabación.

Debería mencionarse eso para la onda de referencia plana definida en Eq. (2.3) ambas fórmulas reconstrucción, Eq. (2.1) y (2.4), son equivalentes porque $E_R = E_R^* \equiv a_R$. La configuración de la figura 2.5 con una onda de referencia plana que incide perpendicularmente en el CCD se usa a menudo en Holografía digital.

La reconstrucción de la imagen virtual también es posible mediante la introducción de propiedades de la imagen de una lente en el proceso de reconstrucción numérica. Esta lente corresponde a la lente del ojo de un observador que mira a través de un holograma ópticamente reconstruido. En el caso más simple, este objetivo se encuentra directamente detrás del holograma, figura 2.7. Las propiedades de imagen de una lente con distancia focal f son consideradas por un factor complejo:

$$L(x,y) = exp\left[i\,\frac{\pi}{\lambda f}\,(x^2+y^2\right] \qquad \text{Ec. 2.6}$$

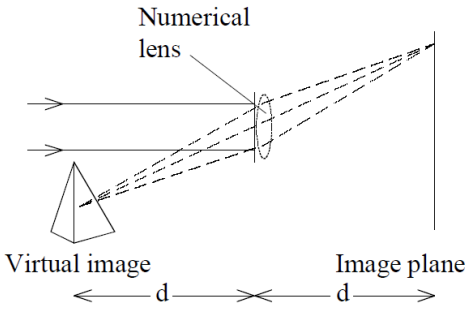


Figura 2.7 Reconstrucción de la imagen virtual.

Para una magnificación de factor 1, una distancia focal de $f = \frac{d}{2}$ tiene que ser considerada.

La lente descrita por Ec. (2.6) causa aberraciones de fase, que pueden corregirse multiplicando el campo de onda reconstruido por un factor

$$P(\xi',\eta') = \exp\left[i\frac{\pi}{\lambda f}\,(\xi'^2+\eta'^2)\right] \qquad \text{Ec. 2.7}$$

La fórmula completa para la reconstrucción a través de una lente virtual con $f = \frac{d}{2}$ es por lo tanto:

$$\Gamma(\xi', \eta') = \frac{i}{\lambda} P(\xi', \eta') \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(x, y) E_R(x, y) L(x, y) \frac{exp\left(-i\frac{2\pi}{\lambda}\rho_I\right)}{\rho_I} \, dxdy \qquad \text{Ec. 2.8}$$

2.2.5 Holografía digital mediante desplazamiento de fase

La amplitud y fase de una onda de luz se puede reconstruir a partir de un único holograma por métodos numéricos como Reconstrucción por Convolución, Fourier o Aproximación de Fresnel [Anexo 3]. La configuración en línea de DH hace uso del conteo de píxeles completo para formar la imagen holográfica, pero los términos de orden cero y de imagen gemela se superponen en la imagen. La holografía digital de desplazamiento de fase (PSDH) es un método muy efectivo para eliminar estos términos, introducido por I. Yamaguchi, donde el campo complejo en el holograma se obtiene por interferometría de desplazamiento de fase [44]. Desde el campo complejo en el plano de holograma, que incluye la información de amplitud y fase, el campo óptico en cualquier otro plano se puede obtener mediante difracción numérica.

La onda de referencia se guía a través de un espejo montado en un piezoeléctrico transductor (PZT). Con este PZT, la fase de la onda de referencia se puede desplazar paso a paso. Se registran varios (al menos tres) interferogramas con cambios de fase mutuos. Por simplicidad, supongamos que la referencia es una onda plana que normalmente incide en el plano de holograma $E_R = \mathcal{E}_R \exp(i\alpha)$, donde α es la fase global. La onda del objeto tiene la amplitud $E_O(x, y)$ y la distribucion de la fase $\varphi(x, y)$, de modo que:

$$E_o = \mathcal{E}_o(x, y) \exp[i\varphi(x, y)] \qquad \text{Ec.2.9}$$

Por lo tanto la intensidad de interferencia es:

$$I_\alpha = |E_O + E_R|^2 = \mathcal{E}_R^2 + \mathcal{E}_O^2(x,y) + \mathcal{E}_R\mathcal{E}_O(x,y)e^{i(\varphi-\alpha)} + \mathcal{E}_R\mathcal{E}_O(x,y)e^{-i(\varphi-\alpha)}$$

$$I_\alpha = \mathcal{E}_R^2 + \mathcal{E}_O^2(x,y) + 2\mathcal{E}_R\mathcal{E}_O(x,y)\cos[\varphi(x,y) - \alpha] \qquad \text{Ec.2.10}$$

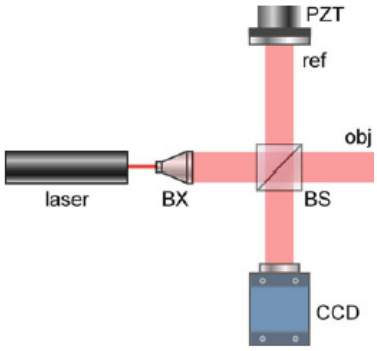


Figura 2.8 Holografía digital de desplazamiento de fase con interferómetro de Michelson. Expansor de haz BX, divisor de haz BS, espejo de referencia PZT piezoeléctrico.

En el PSDH original de cuatro pasos [44], cuatro hologramas con cambios de fase $\alpha = 0, \frac{\pi}{2}, \pi, 3\pi/2$ se adquieren, por ejemplo, utilizando un espejo de referencia montado en un piezoeléctrico. (Fig. 2.8):

$$I_0 = \mathcal{E}_R^2 + \mathcal{E}_O^2 + 2\mathcal{E}_R\mathcal{E}_O cos\varphi \ , \qquad I_{\pi/2} = \mathcal{E}_R^2 + \mathcal{E}_O^2 - 2\mathcal{E}_R\mathcal{E}_O sen\varphi$$

$$I_\pi = \mathcal{E}_R^2 + \mathcal{E}_O^2 - 2\mathcal{E}_R\mathcal{E}_O cos\varphi \ , \qquad I_{3\pi/2} = \mathcal{E}_R^2 + \mathcal{E}_O^2 + 2\mathcal{E}_R\mathcal{E}_O sen\varphi \qquad \text{Ec. 2.11}$$

que luego se combinan numéricamente para extraer el perfil de fase

$$\varphi(x,y) = tan^{-1}\left[\frac{I_{\pi/2} - I_{3\pi/2}}{I_0 - I_\pi}\right] \qquad \text{Ec. 2.12}$$

Y la amplitud del campo objeto $\sqrt{E_O^2(x,y)}$ se puede obtener mediante una exposición separada del objeto sin el haz de referencia, lo que requiere un total de cinco exposiciones.

La imagen holográfica es obtenida entonces de $E_O(x,y) = \sqrt{E_O^2}\exp[i\varphi]$. Alternativamente el campo complejo puede ser obtenido:

$$E_O(x,y) = 1/4\mathcal{E}_R\left[(I_O - I_\pi) + i(I_{\frac{3\pi}{2}} - I_{\frac{\pi}{2}})\right] \qquad \text{Ec. 2.13}$$

Esto define completamente el campo óptico complejo $E_O(x,y;0)$ del objeto en el plano de holograma, y la teoría de difracción se puede utilizar para calcular el campo óptico $E_O(x,y;z)$ a cualquier distancia z del holograma.

Estos procedimientos eliminan las contribuciones de los términos de orden cero y la imagen gemela. La interferometría de desplazamiento de fase (PSI), es decir, sin la difracción numérica, se ha utilizado ampliamente en metrología de superficie y otras aplicaciones [45].

El principio de desplazamiento de fase se aplica igualmente a las técnicas de proyección de franjas no interferométricas para el perfil de superficie 3D. Se han desarrollado muchas técnicas para PSI y PSDH y algunas de ellas se describen en [46]. La idea de combinar dos hologramas con una diferencia de fase en cuadratura ya había sido concebida en la década de 1950 por Gabor y Goss, pero la complejidad del sistema optomecánico era sustancial, lo que dificultaba su implementación práctica. Por otro lado, con la implementación digital, muchas de las manipulaciones ópticas se reemplazan con operaciones numéricas de una manera altamente eficiente y versátil, produciendo aplicaciones potentes en muchas áreas diferentes.

2.3 Microscopía holográfica digital

La microscopía es una de las principales áreas de investigación y aplicación de la holografía digital. El acceso directo a la fase y a los perfiles de amplitud hace que la microscopía de fase cuantitativa mediante holografía digital (DH-QPM) sea especialmente potente y versátil. Se desarrollan varias técnicas de DH especialmente para la obtención de imágenes por microscopía, que son posibles gracias a las características de imagen particulares de DH.

Los principios digitales holográficos e interferométricos son la base de muchas otras técnicas de QPM con nuevas capacidades.

Al igual que en la fotografía clásica, la profundidad de campo está determinada por la distancia desde el plano del objeto más cercano en foco a la del plano más lejano que también se enfoca simultáneamente. En microscopía, la profundidad de campo es muy corta y generalmente se mide en unidades de micras, por lo tanto, la profundidad de campo está muy limitada debido a un alto aumento de un objetivo.

La investigación de un objeto tridimensional con resolución microscópica requiere, por lo tanto, ciertos pasos de reenfoque. La holografía digital ofrece la posibilidad de enfocarse en diferentes capas de objetos por métodos numéricos.

Además, las imágenes están libres de aberraciones debido a las imperfecciones de las lentes ópticas. Uno de los trabajos fundamentales en el campo de la Microscopía Holográfica Digital (DHM) ha sido realizado por, Kebbel, Hartmann y Jüptner [47].

2.3.1 Modificación de fase en DHM

La Holografía digital con desplazamiento de fase también se aplicó a la microscopía [48]. El principio de este método se muestra en la configuración de la figura 2.9. Un haz de luz se acopla a un interferómetro Mach-Zehnder. La muestra a investigar (objeto) está montada en un brazo del interferómetro. Se ve una imagen en el objetivo CCD por un objetivo de microscopio (MO). Un segundo objetivo está montado en el brazo de referencia para formar un frente de onda de referencia con la misma curvatura. Ambas ondas parciales interfieren en el objetivo CCD. Una imagen de la muestra superpuesta por un fondo coherente (onda de referencia) se forma en el objetivo CCD.

Se graba un conjunto de imágenes con desplazamiento de fase. El desplazamiento de fase se realiza mediante un transductor piezoeléctrico (PZT) en el brazo de referencia del interferómetro.

A partir de estas imágenes desplazadas en fase, la amplitud compleja del frente de onda del objeto en el plano de la imagen puede calcularse como se describe en el [anexo 1]. El enfoque numérico en cualquier otro plano de objetos ahora es posible con la integral de Fresnel-Kirchhoff.

La calidad de las imágenes grabadas con luz coherente es en general de menor calidad que las grabadas con luz incoherente debido al ruido coherente. Dubois, Joannes y Legros desarrollaron, por lo tanto, un microscopio holográfico digital de desplazamiento de fase con un LED ordinario como fuente de luz [Dubois F 1999]. La calidad de la imagen mejora (menos ruido de speckle) debido a la reducida coherencia espacial de esa fuente de luz en comparación con las imágenes generadas por un láser.

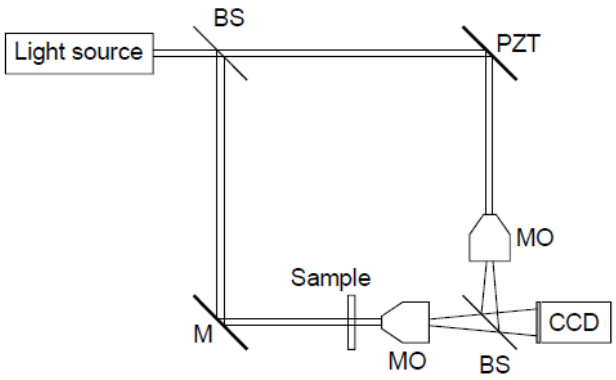


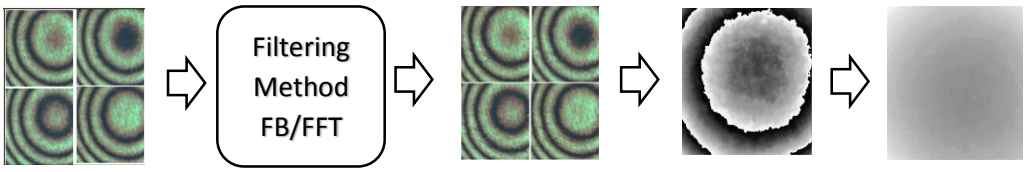
Figura 2.9 Cambio de fase del microscopio holográfico digital.

2.4 Filtros digitales.

Durante un proceso de adquisición de imágenes, la calidad de las imágenes almacenadas debe mejorar debido a factores inherentes al proceso, como errores en los sensores, imperfecciones en los lentes ópticos o mala focalización, entre otros. Por eso es conveniente resaltar parte de la información gráfica y, por lo tanto, se hace necesario un preprocesamiento, que consiste en una optimización de la imagen antes de que tenga lugar el procesamiento definitivo.

Las formas de mejora de imagen se pueden dividir en dos grandes categorías: métodos de dominio espacial y métodos de dominio de frecuencia. El término "dominio espacial" se refiere al mismo plano de imagen y se basa en la manipulación directa de píxeles en una imagen; el filtrado espacial se utiliza para suprimir el ruido o suavizar las imágenes. Las técnicas de dominio de frecuencia se basan en la modificación de transformada de Fourier de una imagen; El filtro bilateral es un filtro de suavizado que conserva los bordes sobre la imagen: el valor de un pixel se calcula en base a una media ponderada de los pixeles vecinos y con valores similares. En las regiones más uniformes, los pixeles de un vecindario son similares entre si y el filtro actúa eliminando las pequeñas diferencias atribuibles al ruido.

Tal procedimiento permite mantener los bordes; una nuestra secuencia de filtrado aplicada en la imágenes obtenidas en este trabajo se ilustra en la Fig. 2.10.



Imágenes sin filtro Imágenes filtradas Fase envuelta Fase Desenvuelta

Figura 2.10 Proceso de filtrado.

2.4.1 Filtros espaciales

Las operaciones de vecindad funcionan con los valores de los píxeles de la imagen en la localidad y los valores correspondientes de una subimagen que tiene las mismas dimensiones que el vecindario. La subimagen es llamada filtro, máscara, kernel, plantilla o ventana, siendo los primeros tres términos la terminología más prevalente. Los valores en una subimagen de filtro se refieren como coeficientes, en lugar de píxeles.

El proceso del filtrado espacial consiste en simplemente mover la máscara de filtro de un punto a otro en una imagen. En cada punto (x, y), la respuesta del filtro en ese punto se calcula utilizando un predefinida relación. Entre los filtros más usados están el Filtro Gaussiano, Laplace, y Mediana entre otros [49, 50, 51, 52]. Para una máscara de 3x3 figura 2.11, el resultado (o respuesta), R, del filtrado lineal con la máscara de filtro en un punto (x, y) en el la imagen es: $R = w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + \cdots + w(0,0)f(x,y) + \cdots + w(1,1)f(x+1,y+1)$

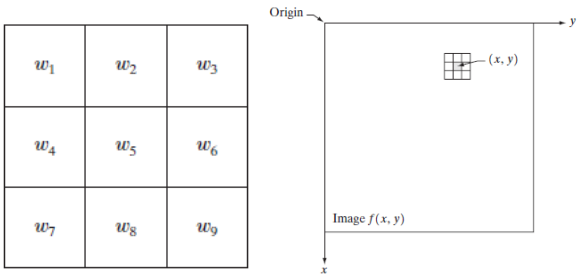


Figura 2.11 Un vecindario 3 x 3 alrededor de un punto (x, y) en una imagen.

Existe, sin embargo, una relación entre los filtros espaciales y el filtrado en el dominio de frecuencias, por eso toman su nombre de ahí:

Filtros Pasa-Baja. Se utilizan en la reducción del ruido o en la difusión de bordes (ambos asociados con frecuencias altas). Suavizan las imágenes produciendo una cierta pérdida de nitidez (smoothing).

Filtros Pasa-Alta. Con el efecto contrario (sharpening). Se utilizan para resaltar los detalles finos o para recuperar detalles perdidos en una mala adquisición de la imagen.

Filtros Pasa-Banda. Son una combinación de ambos, ya que resaltan un determinado tipo de detalles (banda de frecuencias).

2.4.2 Filtro Bilateral

El filtro bilateral es un filtro de suavizado que conserva los bordes: el valor de un pixel se calcula en base a una media ponderada de los pixeles vecinos y con valores similares. En las regiones más uniformes, los pixeles de un vecindario son similares entre si y el filtro actúa eliminando las pequeñas diferencias atribuibles al ruido. Cuando el pixel central se encuentra en un borde entre zonas oscuras y claras el filtro reemplaza su valor por la media de los pixeles brillantes, ignorando los oscuros (cuando se centra en uno oscuro se promedian los oscuros e ignoran los brillantes). Este comportamiento permite mantener los bordes.

Una de las formas más elementales de eliminar el ruido es mediante una convolución gaussiana, que es el concepto fundamental para un Filtro Bilateral; una imagen filtrada por convolución gaussiana está dada por:

$$GC[I]_p = \sum_{q \in S} G_\sigma(\|p - q\|)\, I_q \qquad \text{Ec. 2.14}$$

donde $G_\sigma(\|p - q\|)$ denota el kernel Gaussiano:

$$G_\sigma(\|p - q\|) = \frac{1}{(2\pi\sigma^2)} exp^{\left(\frac{\|p-q\|^2}{2\sigma^2}\right)} \qquad \text{Ec. 2.15}$$

El filtrado gaussiano es un promedio ponderado de la intensidad de las posiciones adyacentes con un peso que disminuye con la distancia espacial q a la posición central p. El peso para el píxel q está definido por gaussiano $G_\sigma(\|p - q\|)$, donde σ es un parámetro que define el tamaño del vecindario o la extensión de la ventana. A medida que esta ventana se extiende sobre toda la imagen, el resultado es un efecto borroso en los bordes de la imagen. El Filtro Bilateral indicado por *BF* se define por:

$$BF[I]_p = \frac{1}{W_p} \sum_{q \epsilon S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}\left(\left|I_p - I_q\right|\right) I_q \qquad \text{Ec. 2.16}$$

Aquí el término W_p es un factor de normalización para asegurar que el promedio ponderado de píxeles sea igual a 1 dentro de la ventana. Se define como:

$$W_p = \sum_{q \epsilon S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(|I_p - I_q|) \quad \text{Ec. 2.17}$$

I_p es el valor de la imagen en la posición p, los parámetros σ_s y σ_r especifican la cantidad de filtrado para la imagen. La ecuación 2.16 representa un promedio ponderado normalizado, donde G_{σ_s} es una ponderación especial que disminuye la influencia de píxeles distantes, G_{σ_r} es un rango gaussiano que disminuye la influencia de los píxeles q cuando sus valores de intensidad difieren de I_p, el filtro bilateral está controlado por dos parámetros: σ_s y σ_r, en este trabajo de tesis utilizamos el valor de σ_s como "tamaño de ventana" y σ_r como de sigma valor de "sigma" en los resultados implementados, referencias [53, 54, 55, 56].

2.4.3 Transformada de Fourier

La "Transformada de Fourier" es un proceso que toma muestras de datos y genera su contenido de frecuencia. Su aplicación general se puede resumir de la siguiente manera.

• Si se toma una señal de audio, resultara su contenido en frecuencia.

• En una imagen de datos, resultara su contenido de frecuencia espacial.

La salida de la Transformada de Fourier contendrá toda la información de su entrada. Un proceso conocido como la Transformada de Fourier inversa se puede utilizar para recuperar la señal original, es un proceso común utilizado en varios campos y se utiliza ampliamente en muchos programas donde este procedimiento funciona como un ecualizador, un filtro, un compresor, etc. La transformada de Fourier $F(u)$, de una sola función continua variable $f(x)$, se define mediante la ecuación:

$$F(u) = \int_{-\infty}^{\infty} f(x)\, e^{-j2\pi ux} dx \quad \text{Ec. 2.18}$$

Donde $j = \sqrt{-1}$. Recíprocamente, dado $F(u)$, podemos obtener $f(x)$ por medio de la transformada inversa de Fourier:

$$f(x) = \int_{-\infty}^{\infty} F(u)\, e^{j2\pi ux} du \qquad \text{Ec. 2.19}$$

Aunque esta fórmula permite procesar datos digitales, con un número de muestra N finito, existe una desventaja en este método porque, a medida que aumenta el número de N^2 puntos, el tiempo de procesamiento también aumenta a una potencia de 2. Por esta razón, un Se desarrolló un proceso alternativo conocido como la transformada discreta de Fourier (DFT) para estimar la transformada de Fourier. El algoritmo Cooley-Tukey aprovecha la naturaleza cíclica de la transformada de Fourier y resuelve el problema con (N log N), dividiendo DFT en DFT más pequeños. Existe una implementación optimizada de DFT, llamada Fast Fourier Transform (transformada rápida de Fourier). El núcleo de esta transformada rápida de Fourier (FFT) consiste en la "operación mariposa", la operación se realiza en un par de muestras de datos cada vez, cuyo gráfico de flujo de señal. La operación tiene su nombre debido al hecho de que cada segmento de este gráfico de flujo se parece a una mariposa Figura. 2.12 Operación mariposa.

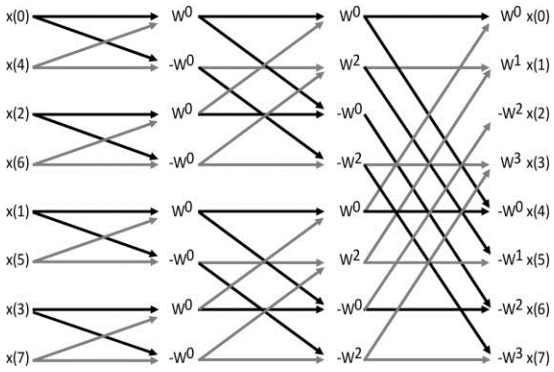


Figura 2.12 Operación mariposa.

El algoritmo básico de FFT se realiza en datos unidimensionales. Para obtener la FFT de una imagen, se toma una FFT para una fila y una columna. Luego, la FFT se calcula primero para cada fila, esta se transpone y una FFT se calcula nuevamente para dicho resultado.

Esto se hace para un acceso más rápido a la memoria, ya que los datos se almacenan en la fila principal. Si no se realiza la transposición, se produce un acceso alterno a la memoria que reduce considerablemente la tasa de rendimiento a medida que aumenta el tamaño de la imagen. Una vez que se obtiene la imagen de transformada de Fourier $F(x,y)$ de $f(x,y)$, se multiplicará por un filtro $H(u,v)$ (una máscara que se puede usar como filtro pasa alto, bajo o pasa banda); en este trabajo utilizamos diferentes tamaños para el radio de máscara, principalmente tomamos un filtro basa baja, por lo que se toma en cuenta el tamaño total de una imagen y el radio de filtro a utilizar; de esta forma obtendremos una función $g(x,y)$ (Fig. 2.13) referencias [57,58].

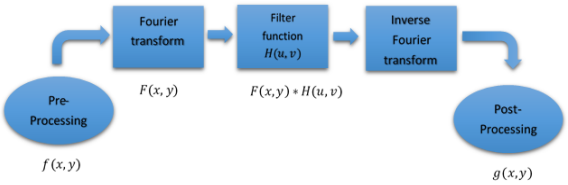


Figura 2.13 Filtro de transformada de Fourier
Referencia [53].

Como resultado obtenemos $g(x,y) = F(x,y) * H(u,v)$, la Imagen Filtrada se obtiene aplicando la transformada de Fourier inversa de $g(x,y)$.

$$Imagen\ Filtrada = F^{-1}[g(x,y)] \qquad \text{Ec. 2.20}$$

2.5 FPGA

Los arreglos de puertas programables en campo FPGA (Field-Programmable Gate Array) son circuitos integrados (CI) digitales que contienen bloques lógicos configurables (programables) junto con las interconexiones configurables entre estos bloques. Los ingenieros de diseño pueden configurar estos dispositivos para realizar una amplia variedad de tareas. Dependiendo de la forma en que se implementen, algunos FPGA solo pueden programarse una sola vez, mientras otros pueden reprogramarse una y otra vez.

La parte "programable de campo" del nombre de FPGA se refiere al hecho de que su programación tiene lugar "en el campo" (a diferencia de los dispositivos cuya funcionalidad interna está cableada por el fabricante). Esto puede significar que los FPGA son configurados en el laboratorio, o puede referirse a modificar el función de un dispositivo residente en un sistema electrónico que tiene ya se ha desplegado en el mundo exterior, a continuación se presenta un resumen de las características más importantes de estos dispositivos.

2.5.1 Dispositivos FPGA

Alrededor del comienzo de la década de 1980, se hizo evidente que había una brecha digital en CI. En un extremo, había dispositivos programables como SPLD y CPLD, que eran altamente configurables, tenían un diseño y modificación rápida, pero que no podría soportar funciones grandes o complejas. En el otro extremo del espectro estaban los ASIC. Estos podrían admitir funciones extremadamente grandes y complejas, pero su diseño era muy costoso y laborioso. Además, una vez que se había implementado un diseño como ASIC, era efectivamente congelado en silicio (Figura 2.14).

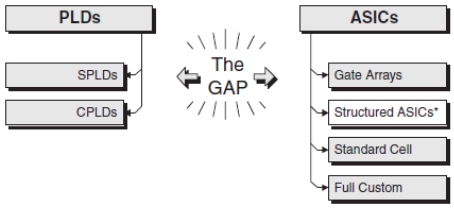


Figura 2.14 Brecha entre PLDs y ASICs

Con el fin de resolver este problema, Xilinx desarrolló una nueva clase de CI llamada matriz de compuertas programable en campo, o FPGA, que pusieron a disposición del mercado en 1984.

Para la arquitectura CPLD clásica está compuesta de bloques lógicos de tipo PAL/GAL o PLA, con interconexiones programables. Básicamente, una FPGA difiere en cuanto a la arquitectura, no utiliza matrices de tipo PAL/PLA y tiene unas densidades mucho mayores que los dispositivos CPLD (para más explicación de abreviaturas anexo 1). Estos primeros dispositivos se basaban en el concepto de un bloque lógico programable, que comprendía una tabla de búsqueda de 3 entradas (LUT), un registro que podía actuar como un flip-flop o un pestillo, y un multiplexor, junto con algunos otros elementos. La figura 2.15 muestra un bloque lógico programable muy simple (los bloques lógicos en los FPGA modernos pueden ser mucho más complejos).

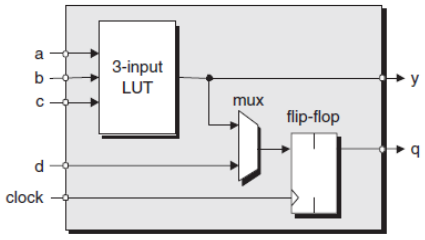


Figura 2.15 Los elementos clave que forman un simple bloque lógico programable.

Cada FPGA contenía una gran cantidad de estos bloques lógicos programables. Por medio de células de programación SRAM apropiadas, cada bloque lógico en el dispositivo podría configurarse para realizar una función diferente.

Cada registro podría configurarse para inicializarse que contenga un 0 lógico o un 1 lógico y para actuar como un flip-flop (como se muestra en la Figura 2.15) o un pestillo. Si se seleccionó la opción flip-flop, el registro podría configurarse para ser activado por un positivo o reloj negativo (la señal del reloj era común a todos los bloques lógicos).

El FPGA completo comprendía una gran cantidad de "islas" de bloques lógicos programables rodeados por un "mar" de interconexiones programables (Figura 2.16).

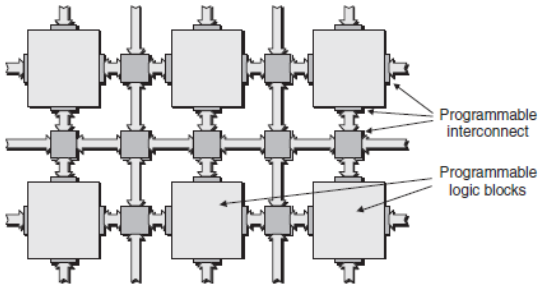


Figura 2.16 Vista de arriba hacia abajo de la arquitectura de FPGA.

Además de la interconexión local reflejada en la figura 2.16, también habría rutas de interconexión globales (de alta velocidad) que podrían transportar señales a través del chip sin tener que atravesar múltiples elementos de conmutación locales. El dispositivo también incluiría pines y pads de E/S primarios. Por medio de sus propias células SRAM, la interconexión podría programarse de tal forma que las entradas primarias al dispositivo estuvieran conectadas a las entradas de uno o más programables bloques lógicos, y las salidas de cualquier bloque lógico podrían usarse para conducir las entradas a cualquier otro bloque lógico, las salidas primarias del dispositivo o ambos.

2.5.2 Dispositivos basados en SRAM

La mayoría de los FPGA se basan en el uso de celdas de configuración de SRAM, lo que significa que se pueden configurar una y otra vez. Las principales ventajas de esta técnica es que las nuevas ideas de diseño se pueden implementar de forma rápida y probada, mientras que las normas y protocolos en evolución pueden acomodarse de forma relativamente sencilla. Cuando el sistema se activa por primera vez, el FPGA puede programarse inicialmente para funcionar una función como autocomprobación o prueba de placa/sistema, y luego puede reprogramarse para realizar su tarea principal.

Además, las células SRAM se crean usando exactamente las mismas tecnologías CMOS. Una desventaja de los dispositivos basados en SRAM es que tienen que reconfigurarse cada vez que se enciende el sistema.

A diferencia de los dispositivos basados en SRAM, que se programan mientras residen en el sistema, los dispositivos basados en antifuse se programan fuera de línea utilizando un programador de dispositivo especial. En primer lugar, estos dispositivos no son volátiles (sus datos de configuración permanecen cuando el sistema está apagado), lo que significa que están disponibles de inmediato tan pronto como se aplica energía al sistema.

2.5.3 Arquitecturas Basadas en LUT

Los LUTs son componentes de memoria SRAM donde se almacena una tabla de verdad que define una función booleana y cuyas entradas de dirección son las variables de entrada de dicha función. Las arquitecturas FPGA basadas en LUT a menudo se clasifican como de grano medio, lo que deja la apelación de grano grueso para aplicarse a estos nuevos dispositivos basados en nodos. El concepto subyacente detrás de una LUT es relativamente simple. Un grupo de señales de entrada se usa como un índice (puntero) en una tabla de búsqueda. El contenido de esta tabla está organizado de manera que la celda a la que apunta cada combinación de entrada contiene el valor deseado. Por ejemplo, supongamos que deseamos implementar la función:

$$y = (a \,\&\, b) \mid c$$

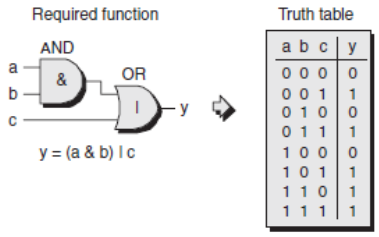


Figura 2.17 Función requerida y tabla de verdad asociada.

Esto se puede lograr cargando un LUT de 3 entradas con los valores apropiados. A los fines de los siguientes ejemplos, supondremos que la LUT está formada por células. Una técnica comúnmente utilizada es usar las entradas para seleccionar la celda SRAM deseada usando una cascada de compuertas de transmisión.

2.5.4 CLB y LAB

Xilinx llama un bloque lógico configurable (CLB) y a lo que Altera se refiere como un bloque de matriz lógica (LAB).Usando CLB como ejemplo, algunos FPGA Xilinx tienen dos slice en cada CLB, mientras que otros tienen cuatro. Para este ejemplo un CLB equivale a un solo bloque lógico en nuestra visualización original de "islas" de lógica programable en un "mar" de interconexión programable (Figura 2.18).

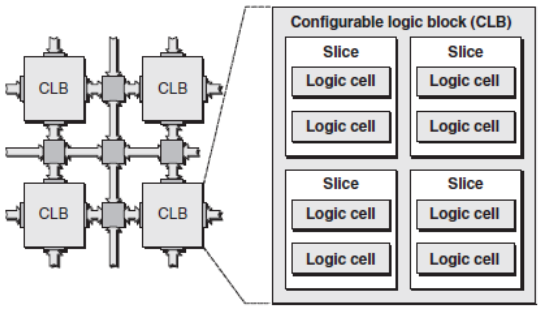


Figura 2.18 Un CLB que contiene cuatro slice (el número de slice depende de la familia FPGA).

2.5.5 RAM incorporadas

Muchas aplicaciones requieren el uso de memoria, por lo que los FPGA ahora incluyen trozos relativamente grandes de RAM incrustada llamada e-RAM o block RAM. Dependiendo de la arquitectura del componente, estos bloques pueden colocarse alrededor de la periferia del dispositivo, esparcidas por la cara del chip en relativo aislamiento u organizados en columnas, como se muestra en la Figura 2.19.

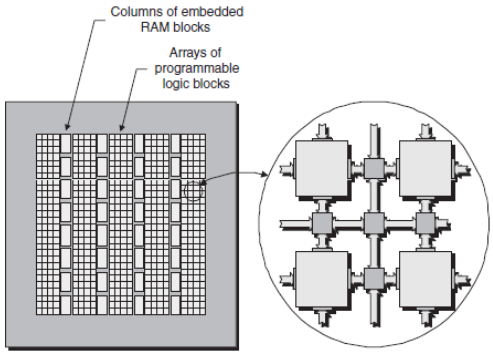


Figura 2.19 Ilustración un chip con columnas de bloques RAM incrustados.

Dependiendo del dispositivo, dicha RAM podría contener desde unos pocos miles hasta decenas de miles de bits. Además, un dispositivo puede contener entre decenas y cientos de estos bloques de RAM, proporcionando así una capacidad total de almacenamiento de unos pocos cientos de miles de bits hasta varios millones de bits. Cada bloque de RAM se puede usar de forma independiente, o se pueden combinar múltiples bloques para implementar bloques más grandes. Estos bloques se pueden usar para una variedad de propósitos, como la implementación de RAM estándar de puerto único o doble, funciones como el primero que entra es el primero en salir (FIFO), máquinas de estado, etc.

2.5.6 Multiplicadores integrados, sumadores, MAC, etc.

Algunas funciones, como los multiplicadores, son intrínsecamente lentas si se implementan conectando juntos una gran cantidad de bloques lógicos programables. Dado que muchas aplicaciones requieren estas funciones, muchos FPGA incorporan bloques multiplicadores especiales cableados. Estos se encuentran típicamente cerca de los bloques de RAM porque estas funciones a menudo se utilizan conjuntamente (Figura 2.20).

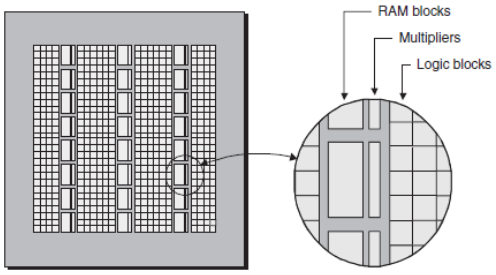


Figura 2.20 Un chip con columnas de multiplicadores incrustados y bloques RAM.

Del mismo modo, algunos FPGA ofrecen bloques de sumadores dedicados. Una operación que es muy común en las aplicaciones de tipo DSP se llama multiply-and-accumulate (MAC). Como su nombre sugeriría, esta función multiplica dos números y agrega el resultado a un total acumulado almacenado en un acumulador.

2.5.7 Núcleos de microprocesador duro

Un núcleo de microprocesador duro es uno que se implementa como un bloque dedicado, predefinido (cableado) (estos núcleos solo están disponibles en ciertas familias de dispositivos). Cada uno de los principales proveedores de FPGA ha optado por un tipo de procesador particular para implementar sus núcleos duros. Por ejemplo, Altera ofrece procesadores ARM incrustados como el SoC utilizado en esta tesis con el DE1_SoC, QuickLogic ha optado por soluciones basadas en MIPS y Xilinx presenta núcleos PowerPC. Existen dos enfoques principales para integrar dichos núcleos en el FPGA. El primero es ubicarlo en una tira al costado del tejido FPGA principal (Figura 2.21). En este escenario, todos los componentes se forman típicamente en el mismo chip de silicio, aunque también se podrían formar en dos chips y empaquetarse como un módulo multichip (MCM).

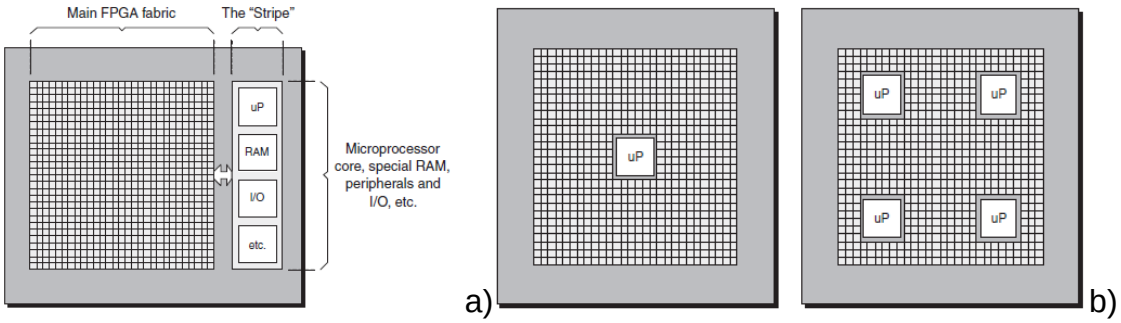


Figura 2.21 a) Un chip con núcleo incrustado fuera de la estructura principal, b) Uno o múltiples núcleos.

La estructura principal de FPGA también incluiría los bloques de RAM incorporados, los multiplicadores y otros introducidos anteriormente, pero se han omitido en esta ilustración para simplificar las cosas.

2.5.8 Núcleos de microprocesador blando

A diferencia de insertar físicamente un microprocesador en la estructura del chip, es posible configurar un grupo de bloques lógicos programables para que actúen como un microprocesador. Suelen denominarse núcleos blandos, pero pueden categorizarse con mayor precisión como "blandos" o "firmes", según la forma en que se asigna la funcionalidad del microprocesador a los bloques lógicos. Los núcleos blandos son más simples y más lentos que sus contrapartes de núcleo duro. Sin embargo, tienen la ventaja de que solo necesita implementar un núcleo si lo necesita y también que puede crear la instancia de todos los núcleos que necesite hasta quedarse sin recursos en forma de bloques lógicos programables. Por ejemplo, Altera ofrece el Nios, mientras que Xilinx tiene el Micro-Blaze. El Nios tiene variantes de arquitectura de 16 y 32 bits, que operan en fragmentos de datos de 16 bits o 32 bits, respectivamente (ambas variantes comparten el mismo conjunto de instrucciones de 16 bits de ancho).

2.5.9 Árboles de reloj y Administradores de reloj

Todos los elementos sincrónicos dentro de un FPGA, por ejemplo, los registros configurados para actuar como flip-flops dentro de los bloques lógicos programables necesitan ser accionados por una señal de reloj. Dicha señal de reloj típicamente se origina en el mundo exterior, entra al FPGA a través de un pin de entrada de reloj especial, y luego se en ruta a través del dispositivo y se conecta a los registros apropiados.

Considerando una representación simplificada que omita los bloques lógicos programables y muestre solo el árbol de reloj y los registros a los que está conectado (Figura 2.22a). En lugar de configurar un pin de reloj para conectarse directamente en un árbol de reloj interno, ese pin se puede usar para manejar una función especial de cableado (bloque) llamada un administrador de reloj que genera un número de relojes hija (Figura 2.22b).

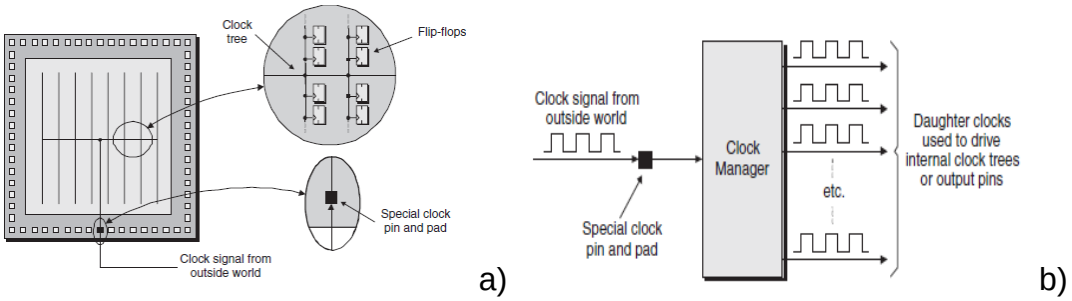


Figura 2.22 a) Un árbol de reloj simple, b) Un administrador de reloj genera relojes hijos.

Estos relojes hijos se pueden usar para controlar árboles de reloj internos o pines de salida externos que se pueden usar para proporcionar servicios de reloj a otros dispositivos en la placa de circuito del host. Cada familia de FPGA tiene su propio tipo de administrador de reloj, donde diferentes administradores de reloj pueden admitir solo un subconjunto de las siguientes características: Algunos administradores de reloj FPGA se basan en bucles de fase bloqueada (PLL), mientras que otros se basan en bucles de bloqueo de retraso digital (DLL).

Los PLL se han utilizado desde la década de 1940 en implementaciones analógicas, pero el énfasis reciente en los métodos digitales ha hecho que sea deseable hacer coincidir las fases de señal digitalmente.

2.5.10 E/S de propósito general

Los paquetes de FPGA de hoy en día pueden tener 1,000 o más pines (E/S, señales, entrada/salida), que están dispuestos como una matriz en la base del paquete. De manera similar, cuando se trata del chip de silicio dentro del paquete, las estrategias de empaquetado de chips permiten que la energía, la tierra, el reloj y los pines de E/S se presenten en la superficie del chip.

2.5.11 Estándares configurables de E/S

Considerando por un momento un producto electrónico desde la perspectiva de los arquitectos e ingenieros que diseñan la placa de circuito. Dependiendo de lo que se pretenda de hacer, los dispositivos que están usando, el entorno en el que funcionará la placa, y así sucesivamente, por lo tanto seleccionarán un estándar particular que se utilizará para transferir señales de datos. (En este contexto, "estándar" se refiere a aspectos eléctricos de las señales, como sus niveles de tensión lógica 0 y lógica 1).

El problema es que existe una gran variedad de estándares, y sería doloroso tener que crear FPGA especiales para acomodar cada variación. Por esta razón, las E/S de propósito general de un FPGA se pueden configurar para aceptar y generar señales que se ajusten a cualquier estándar requerido. Estas señales de E/S de propósito general se dividirán en varios bancos; supondremos que ocho de esos bancos están numerados del 0 al 7 (Figura 2.23).

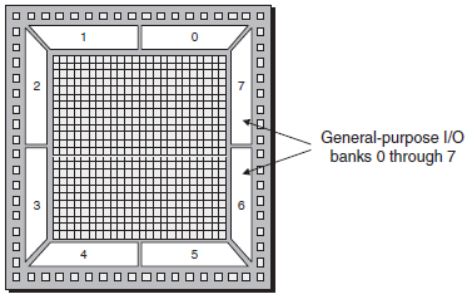


Figura 2.23 Un chip que muestra bancos de E/S de propósito general.

El punto interesante es que cada banco se puede configurar individualmente para admitir un estándar de E/S particular. Por lo tanto, además de permitir que el FPGA funcione con dispositivos que usan múltiples estándares de E/S, esto permite que el FPGA se use realmente para interconectar entre diferentes estándares de E/S (y también para traducir entre diferentes protocolos que pueden basarse en estándares eléctricos particulares).

2.5.12 SoC FPGAs

Al principio de su evolución, los FPGA fueron percibidos por los ingenieros de diseño como compuertas lógicas configurables que podían aplicarse en operaciones simples ya menudo repetitivas en sistemas de bajo volumen que no podían justificar el mayor gasto de un circuito integrado específico para aplicaciones (ASIC). Recientemente, la integración de núcleos ARM® Cortex®-A en FPGA y núcleos de cómputo intensivo podría llevar a creer que las rutas de los verdaderos SoC multinúcleo y los llamados SoC de FPGA, figura 2.24.

La fase evolutiva más reciente para los FPGA ha sido incluir núcleos de procesamiento disponibles en la fábrica, como uno o más núcleos ARM, así como núcleos únicos destinados a reproducir procesadores de señales digitales (DSP). El dispositivo resultante ha sido llamado FPGA SoC. Estos SoCs extendieron la flexibilidad de programación con la inclusión de DSP de coma flotante.

En un FPGA SoC, los núcleos de procesamiento, coprocesadores, aceleradores y otros motores de procesamiento en un SoC se pueden programar en un lenguaje de nivel relativamente alto como C ++ y los flujos de datos a nivel del sistema se pueden redefinir al mismo tiempo. De hecho, el software que se ejecuta en núcleos en un SoC tiene la capacidad de volver a asignar dinámicamente los canales de datos en respuesta a la carga de procesamiento, algo que un FPGA SoC no podía hacer.

La integración de estas tecnologías en la misma pieza de silicio elimina el costo de los paquetes de plástico y ahorra espacio en la placa. Si tanto la CPU como el FPGA utilizan memorias externas separadas, también puede ser posible consolidar ambas en un solo dispositivo de memoria, para mayores ahorros. Como las señales entre el procesador y el FPGA ahora residen en el mismo silicio, la comunicación entre los dos consume sustancialmente menos energía en comparación con el uso de chips separados. La integración de miles de conexiones internas entre el procesador y el FPGA conducen a un ancho de banda sustancialmente más alto y una latencia(retardo temporal) más baja en comparación con una solución de dos chips. Anteriormente, la falta de un procesador ARM había sido una barrera para el uso de la tecnología FPGA para una producción completa [59].

Figura 2.24 Arquitectura FPGA Altera
Referencia [62].

2.6 OPENCL

OpenCL es un estándar de la industria para la programación de computadoras compuesto por una combinación de CPU, GPU, FPGA y otros procesadores. Estos los llamados sistemas heterogéneos se han convertido en una clase importante de plataformas y OpenCL es el primer estándar de la industria que aborda directamente sus necesidades. Lanzado por primera vez en diciembre de 2008 con los primeros productos disponibles en el otoño de 2009, OpenCL es una tecnología relativamente nueva.

Con OpenCL, se puede escribir un único programa que se puede ejecutar en una amplia gama de sistemas, desde teléfonos celulares hasta portátiles y nodos en supercomputadores masivos. Ningún otro estándar de programación paralelo tiene un alcance tan amplio. Esta es una de las razones por las que OpenCL es tan importante y tiene el potencial de transformar la industria del software. También es la fuente de gran parte de las críticas lanzadas en OpenCL.

Las compañías de semiconductores continuarán comprimiendo más y más transistores en un único dado, pero estos proveedores competirán en eficiencia energética en lugar de rendimiento bruto.

2.6.1 Referencias de implementación.

La bioinformática es una de las tecnologías más poderosas en las ciencias de la vida hoy en día, y se está utilizando en la investigación de teorías de evolución y diseño de proteínas, entre otras aplicaciones, algoritmos, métodos y diferentes hallazgos utilizados en estos estudios ofrecen una gran cantidad de aplicaciones, tales como la clasificación funcional de proteínas, estructura secundaria predicción, enhebrado y modelado de homólogos distantes relacionados proteínas para representar su comportamiento a lo largo de una célula ciclo de vida, y alineamientos de secuencia y estructura.

Un ejemplo de implementación donde se analiza la aceleración en GPU y FPGA del algoritmo Smith-Waterman (SW) que es un método de alta sensibilidad para alineaciones locales, mostrando eficiencia en paralelismo de datos y potencia de consumo [60]. En algoritmos como el K-vecino más cercano (KNN) que se utiliza para la categorización de texto, análisis predictivo, datos de minería y reconocimiento de imágenes entre otros implementado con OpenCL [61], utilizando un FPGA como una plataforma beamforming para imágenes de ultrasonido con una eficiencia en consumo de potencia, con los resultados presentados dan respuesta a la creación de un procesador en tiempo real [62,63].

La reconstrucción 3D en tiempo real de una escena, con un sensor de profundidad de bajo costo puede mejorar el desarrollo de tecnologías en los dominios de realidad aumentada, robótica móvil, y más; sin embargo, las implementaciones actuales requieren una computadora con una poderosa GPU, que limita su potencial aplicaciones con requisitos de baja potencia, para implementar baja potencia en reconstrucción 3D, tenemos en [64] la incorporación de dos algoritmos prominentes de la reconstrucción en 3D con un FPGA Altera Stratix® V; una evaluación portabilidad a través de CPU, GPU, APU multinúcleo, donde el estudio experimental demuestra que OpenCL se puede usar de forma realista como una descripción de hardware lenguaje (HDL) mediante el empleo de herramientas como SOpenCL que traducir automáticamente el código OpenCL sin modificaciones a Verilog y eliminar la carga de la programación HDL[65].

Dentro de las implementaciones de la FFT por mencionar en un FPGA, una de ellas a partir de VHDL, que plantea una menor cantidad de recursos utilizados garantizado el rendimiento del dispositivo [66], también en [67] se implementa la descripción y síntesis sobre VHDL de la FFT 2D con representación binaria de punto fijo utilizando la herramienta de programación Simulink HDL Coder de Matlab®; mostrando una manera rápida y fácil de manejar desbordamiento y registros de creación, sumadores y multiplicadores de datos complejos en VHDL.

Con la plataforma System Generator de Xilinx [68], para el FPGA; la síntesis e implementación del diseño se realiza desde el ambiente ISE, La validación final se realiza mediante el modelado sobre la tarjeta de desarrollo ML507, basada en un FPGA de la familia Virtex-5 del mismo fabricante. Los resultados muestran la factibilidad de la implementación propuesta en el FPGA seleccionado.

2.6.1 Fundamentos conceptuales de OpenCL

Estos pasos se logran a través de una serie de API dentro de OpenCL más un entorno de programación para núcleos. OpenCL trabaja bajo los siguientes modelos:

•Modelo de plataforma: Una descripción de alto nivel del sistema heterogéneo.

•Modelo de ejecución: Una representación abstracta de cómo las secuencias de instrucciones se ejecutan en la plataforma heterogénea.

•Modelo de memoria: La colección de regiones de memoria dentro de OpenCL y cómo interactúan durante un cálculo OpenCL.

•Modelos de programación: Las abstracciones de alto nivel que usa un programador cuando diseña algoritmos para implementar una aplicación.

2.6.3 Modelo de plataforma

El modelo de plataforma OpenCL define una representación de alto nivel de cualquier plataforma heterogénea utilizada con OpenCL. Este modelo se muestra en la Figura 2.25, cada dispositivo OpenCL tiene una o más unidades de cálculo, cada una de las cuales tiene uno o más elementos de procesamiento. Una plataforma OpenCL siempre incluye un único host. El host interactúa con el entorno externo al programa OpenCL, incluidas las E/S o la interacción con el usuario de un programa.

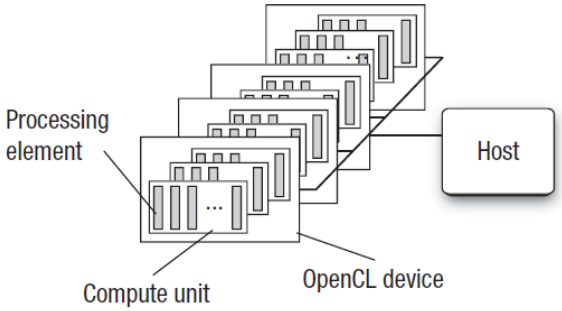


Figura 2.25 El modelo de plataforma OpenCL con un host y uno o más dispositivos OpenCL.

El host está conectado a uno o más dispositivos OpenCL. El dispositivo es donde se ejecutan las secuencias de instrucciones (o kernels); por lo tanto, a menudo se hace referencia a un dispositivo OpenCL como un dispositivo informático. Un dispositivo puede ser una CPU, una GPU, un DSP o cualquier otro procesador proporcionado por el hardware y admitido por el proveedor de OpenCL.

Los dispositivos OpenCL se dividen además en unidades de cálculo que se dividen en uno o más elementos de procesamiento (PE). Los cálculos en un dispositivo ocurren dentro de los PE.

2.6.4 Modelo de ejecución

El modelo de ejecución de OpenCL define cómo se ejecutan los kernels, una aplicación OpenCL consta de dos partes distintas: el programa host y una colección de uno o más kernels. El programa host se ejecuta en el host. OpenCL no define los detalles de cómo funciona el programa host, solo cómo interactúa con los objetos definidos dentro de OpenCL. Los kernels se ejecutan en los dispositivos OpenCL (devices). Ellos hacen el trabajo real de una aplicación OpenCL, Los kernels son típicamente funciones simples que transforman entradas de objetos de memoria en objetos de memoria de salida.

Kernels OpenCL: Funciones escritas con OpenCL C.

Kernels nativos: Funciones creadas fuera de OpenCL y accedidas dentro de OpenCL a través de una función puntero.

2.6.5 Cómo se ejecuta un Kernel en un dispositivo OpenCL

Un kernel está definido en el host. El programa host emite un comando que envía el kernel para su ejecución en un dispositivo (device) OpenCL. Cuando el host emite este comando, el sistema de tiempo de ejecución de OpenCL crea un espacio de índice entero. Se ejecuta una instancia del kernel para cada punto en este espacio de índice. Llamamos a cada instancia de un kernel en ejecución un elemento de trabajo (**work-item)**, que se identifica por sus coordenadas en el espacio de índice. Estas coordenadas son la **ID** global para el elemento de trabajo.

El comando que envía un kernel para su ejecución, por lo tanto, crea una colección de elementos de trabajo, cada uno de los cuales utiliza la misma secuencia de instrucciones definidas por un kernel único. Si bien la secuencia de instrucciones es la misma, el comportamiento de cada elemento de trabajo puede variar debido a las instrucciones de su dependencia de trabajo dentro del código o los datos seleccionados a través del ID global.

Los elementos de trabajo (**Work-items**) están organizados en grupos de trabajo (**work-groups**). Los grupos de trabajo tienen el mismo tamaño en las dimensiones correspondientes y este tamaño divide equitativamente el tamaño global en cada dimensión.

A los work-groups se les asigna una ID única con la misma dimensionalidad que el espacio de índice utilizado para los work-items. Los work-items son asignados con una ID local única dentro de un work-group para que un solo work-item pueda ser identificado de manera única por su ID global o por una combinación de su ID local y su ID de work-group.

Los elementos de trabajo en un grupo de trabajo determinado se ejecutan simultáneamente en los elementos de procesamiento de una sola unidad de cálculo. El espacio de índice abarca un rango de valores N-dimensionados y, por lo tanto, se denomina **NDRange**.

Actualmente, N en este espacio de índice N-dimensional puede ser 1, 2 o 3. Dentro de un programa OpenCL, un **NDRange** está definido por una matriz entera de longitud N que especifica el tamaño del espacio índice en cada dimensión. El ID global y local de cada elemento de trabajo es una tupla N-dimensional. En el caso más simple, los componentes ID globales son valores en el rango de cero a la cantidad de elementos en esa dimensión menos uno.

En la Figura 2.26 se proporciona un ejemplo concreto donde cada cuadro pequeño es un elemento de trabajo. Para este ejemplo, usamos el desplazamiento predeterminado de cero en cada dimensión. El cuadrado sombreado con índice global (6, 5) pertenece al grupo de trabajo con ID (1, 1) e índice local (2, 1).

Por lo tanto, cada elemento de trabajo tiene una coordenada (gx, gy) en un espacio de índice de tamaño global de NDRange (Gx, Gy) y toma los valores $[0\,..\,(Gx - 1), 0\,..\,(Gy - 1)\,]$, e ID del grupo de trabajo (wx, wy).

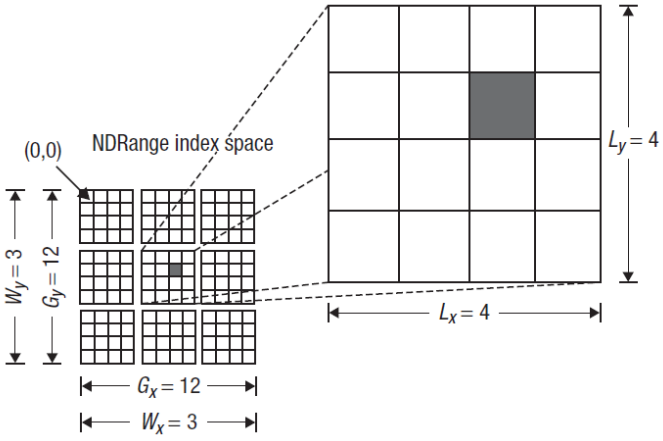


Figura 2.26 Un ejemplo de cómo los identificadores globales (global IDs), ID locales (local IDs) e índices de grupos de trabajo están relacionados para un NDRange bidimensional.

Otros parámetros del espacio de índice se definen en la figura 2.26. El bloque sombreado tiene una ID global de $(g_x, g_y) = (6, 5)$ y un grupo de trabajo $(w_x, w_y) = (1, 1)$ más ID local de y $(l_x, l_y) = (2, 1)$.

2.6.6 Contexto (context)

El anfitrión (Host), sin embargo, juega un papel muy importante en la aplicación OpenCL. Está en el host donde se definen los kernels. El host establece el contexto para los kernels. El host define el NDRange y las colas que controlan los detalles de cómo y cuándo se ejecutan los kernels. Todas estas funciones importantes están contenidas en las API dentro de la definición de OpenCL.

La primera tarea para el host es definir el contexto para la aplicación OpenCL. Como su nombre lo indica, el contexto define el entorno en el que se definen y ejecutan los kernels. Para ser más precisos, definimos el contexto en términos de los siguientes recursos:

• Dispositivos: La colección de dispositivos OpenCL que utilizará el host.

• Kernels: Las funciones OpenCL que se ejecutan en dispositivos OpenCL.

• Objetos de programa: El código fuente del programa y los ejecutables que implementan los kernels.

• Objetos de memoria: un conjunto de objetos en la memoria que son visibles para dispositivos OpenCL y contienen valores que pueden ser operados por instancias de un kernel.

El contexto es creado y manipulado por el host utilizando funciones de la API OpenCL.

2.6.7 Cola de Comandos - (Command-Queues)

La interacción entre el host y los dispositivos OpenCL se produce a través de comandos publicados por el host a la cola de comandos. Estos comandos esperan en la cola de comandos hasta que se ejecutan en el dispositivo OpenCL. Un command-queue es creado por el host y se adjunta a un solo dispositivo OpenCL una vez que se ha definido el contexto. El host coloca los comandos en la cola de comandos, y los comandos se preparan para su ejecución en el dispositivo asociado. OpenCL admite tres tipos de comandos:

• Los comandos de ejecución del kernel, ejecutan un kernel en los elementos de procesamiento de un dispositivo OpenCL.

• Los comandos de memoria, transfieren datos entre el host y diferentes objetos de memoria, mueven datos entre objetos de memoria o asignan y desasignan objetos de memoria del espacio de direcciones del host.

• Los comandos de sincronización ponen restricciones al orden en que se ejecutan los comandos.

2.6.8 Modelo de memoria

El modelo de ejecución establece cómo se ejecutan los kernels, cómo interactúan con el host y cómo interactúan con otros kernels. OpenCL define dos tipos de objetos de memoria: objetos de memoria intermedia y objetos de imagen. Un objeto de memoria intermedia, como su nombre lo indica, es solo un bloque contiguo de memoria disponible para los núcleos.

Un programador puede mapear estructuras de datos en este búfer y acceder al búfer a través de punteros. Esto proporciona flexibilidad para definir casi cualquier estructura de datos que desee el programador (sujeto a las limitaciones del lenguaje de programación del kernel OpenCL).

Los objetos de imagen, por otro lado, están restringidos a la celebración de imágenes. Un formato de almacenamiento de imágenes puede optimizarse para las necesidades de un dispositivo OpenCL específico.

OpenCL también permite a un programador especificar subregiones de objetos de memoria como objetos de memoria distintos (agregados con la especificación OpenCL 1.1). Esto hace que una subregión de un objeto de memoria grande sea un objeto de primera clase en OpenCL que puede manipularse y coordinarse a través de la cola de comandos. Comprender los objetos de memoria en sí mismos es solo un primer paso. También debemos comprender las abstracciones específicas que rigen su uso en un programa OpenCL. El modelo de memoria OpenCL define cinco regiones de memoria distintas:

• Memoria global: esta región de memoria permite el acceso de lectura/escritura a todos los elementos de trabajo en todos los grupos de trabajo. Los elementos de trabajo pueden leer o escribir en cualquier elemento de un objeto de memoria en la memoria global. Las lecturas y escrituras en la memoria global pueden almacenarse en caché dependiendo de las capacidades del dispositivo.

• Memoria constante: esta región de memoria de la memoria global permanece constante durante la ejecución de un kernel. El host asigna e inicializa objetos de memoria colocados en la memoria constante. Los elementos de trabajo tienen acceso de solo lectura a estos objetos.

• Memoria local: esta región de memoria es local para un grupo de trabajo. Esta región de memoria se puede usar para asignar variables que comparten todos los elementos de trabajo en ese grupo de trabajo.

Se puede implementar como regiones de memoria dedicadas en el dispositivo OpenCL. Alternativamente, la región de memoria local puede mapearse en secciones de la memoria global.

• Memoria privada: esta región de memoria es privada para un elemento de trabajo. Las variables definidas en la memoria privada de un elemento de trabajo no son visibles para otros elementos de trabajo.

Las regiones de memoria y cómo se relacionan con la plataforma y los modelos de ejecución se describen en la figura 2.27.

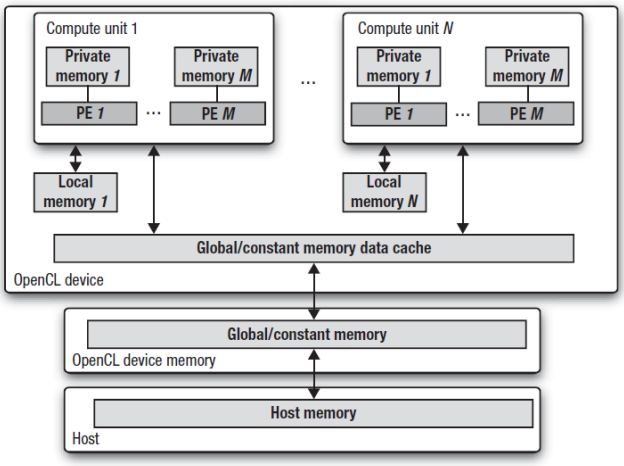


Figura 2.27 Modelo de memoria en OpenCL y cómo las diferentes regiones de memoria interactúan con el modelo de la plataforma.

2.6.9 Modelos de programación

El modelo de ejecución OpenCL define como una aplicación OpenCL se correlaciona con elementos de procesamiento, regiones de memoria y el host. Es un modelo "hardware céntrico". Ahora cambiamos de marcha y describimos cómo mapeamos algoritmos paralelos en OpenCL usando un modelo de programación. Los modelos de programación están íntimamente conectados a cómo los programadores razonan sobre sus algoritmos. Por lo tanto, la naturaleza de estos modelos es más flexible que la del modelo de ejecución definido con precisión.

OpenCL fue definido con dos modelos de programación diferentes en mente: paralelismo de tareas y paralelismo de datos.

2.6.10 Modelo de Programación de Datos en Paralelo

Los problemas bien adaptados al modelo de programación paralela a los datos se organizan en torno a estructuras de datos, cuyos elementos se pueden actualizar simultáneamente. En esencia, una única secuencia lógica de instrucciones se aplica simultáneamente a los elementos de la estructura de datos. La estructura del algoritmo paralelo está diseñada como una secuencia de actualizaciones concurrentes a las estructuras de datos dentro de un problema.

En problemas paralelos de datos más complicados, los elementos de trabajo en un solo grupo de trabajo pueden necesitar compartir datos. Esto se admite a través de los datos almacenados en la región de memoria local. Cada vez que se introducen dependencias entre los elementos de trabajo, se debe tener cuidado de que, independientemente del orden en que se completen los elementos de trabajo, se produzcan los mismos resultados, los elementos de trabajo pueden necesitar sincronizar su ejecución, es decir, los elementos de trabajo en un solo grupo de trabajo pueden participar en **work-group barrier.**

Si el kernel no contiene ninguna instrucción de bifurcación, cada elemento de trabajo ejecutará operaciones idénticas, pero en un subconjunto de elementos de datos seleccionados por su ID global. Este caso define un subconjunto importante del modelo paralelo de datos conocido como Instrucción Única de Datos Múltiples o SIMD. Si bien cada elemento de trabajo utiliza el mismo "programa" (es decir, el kernel), el trabajo real que realiza puede ser bastante diferente. Esto a menudo se conoce como un Programa Único de Datos Múltiples o SPMD.

2.6.11 Modelo de Programación de Tarea en Paralelo

El modelo de ejecución de OpenCL se diseñó claramente con el paralelismo de datos como objetivo principal. Pero el modelo también admite una amplia gama de algoritmos de tareas paralelas.

OpenCL define una tarea como un kernel que se ejecuta como un único elemento de trabajo independientemente del NDRange utilizado por otros kernels en la aplicación OpenCL.

Esto se usa cuando la concurrencia que un programador desea explotar es interna a la tarea. Por ejemplo, el paralelismo puede expresarse únicamente en términos de operaciones vectoriales sobre tipos de vectores. O tal vez la tarea utiliza un kernel definido con la interfaz nativa del núcleo y el paralelismo se expresa utilizando un entorno de programación fuera de OpenCL.

Una segunda versión del paralelismo de tareas cuando los kernels se envían como tareas que se ejecutan al mismo tiempo con una cola fuera de orden. Por ejemplo en una CPU de cuatro núcleos, un núcleo podría ser el host y los otros tres núcleos configurados como unidades de cálculo dentro de un dispositivo OpenCL. La aplicación OpenCL podría poner en cola las seis tareas y dejar que las unidades de cómputo programen dinámicamente el trabajo. Cuando el número de tareas es mucho mayor que el número de unidades de cálculo, esta estrategia puede ser una forma muy efectiva de producir una carga bien equilibrada.

CAPÍTULO 3

Desarrollo experimental
3.1 Elementos ópticos, conceptos básicos

• Láser: La fuente de luz más utilizada dentro de las técnicas dentro de metrología óptica, en este caso Holografía Digital es el láser (Amplificación de la Luz por Emisión Estimulada de radiación). La radiación del láser se obtiene por varios pasos: Primero, los electrones de los átomos del láser son bombeados hasta un estado excitado por una fuente de energía; posteriormente, se les estimula mediante fotones externos para que emitan la energía almacenada en forma de fotones, mediante un proceso conocido como emisión estimulada; los fotones emitidos tienen una frecuencia que depende de los átomos en cuestión y se desplazan en fase con los fotones que los estimulan; los fotones emitidos chocan a su vez con otros átomos excitados y liberan nuevos fotones; la luz se amplifica a medida que los fotones se desplazan hacia atrás y hacia adelante entre dos espejos paralelos desencadenando nuevas emisiones estimuladas, y al mismo tiempo la luz láser intensa, direccional y monocromática, se filtra por uno de los espejos, que es solo parcialmente reflectante. El láser utilizado para esta aplicación es de gas Helio-Neón (He-Ne), la longitud de onda es 633nm (Rojo) y una potencia de salida de 35mW (Figura. 3.1).

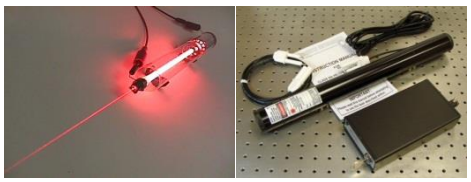


Figura 3.1 Láser Helio-Neón 633nm
Referencia [69].

• Objetivo de microscopio: básicamente se describe como una lente(o varias lentes) de distancia focal muy reducida (alrededor de 1 mm, por ejemplo) y con una apertura numérica muy alta debido a la aplicación para la que fueron diseñados, así entonces la amplificación de imágenes de objetos muy pequeños además de su principal función consiste en colectar la luz proveniente de un espécimen y proyectar una imagen nítida, real y aumentada hacia el plano de observación. Con fines experimentales en el laboratorio de óptica sus características son muy útiles para expandir los haces de luz con que se iluminan los objetos bajo estudio (Colimar el haz: el rayo se expande y mantiene su diámetro a una distancia determinada, así mismo mantiene su eje y altura de (160mm, en este caso) a través de los dispositivos que conforman el arreglo interferométrico). Así, un objetivo de microscopio con amplificación baja (5X) nos proporciona una expansión de haz reducida, de alrededor de 2 cm a dos metros del objetivo, y un objetivo de microscopio con alta amplificación (60X) y alta apertura numérica nos proporciona una expansión notable con un tamaño de mancha de iluminación de unos 30cm a 2 m de distancia. Se utilizó un objetivo Edmund tipo acromático, de aumento de 20x con resorte, una longitud focal efectiva EFL (mm) de 8.55, y apertura numérica de 0.40 (Fig. 3.2).

Figura 3.2 Objetivo de microscopio
Referencia [70].

• Divisor de haz (cube beam splitter) 50/50: su forma más común, un cubo, cada divisor de haz está compuesto por un par de prismas de ángulo recto de precisión de alta tolerancia, cementados junto con un revestimiento dieléctrico sobre la hipotenusa de uno de los prismas, un revestimiento anti reflexión en cada cara del divisor del haz para alcanzar menos de 0.5% de reflexión por la superficie.

Los divisores de haz se utilizan en interferometría para generar dos ondas altamente coherentes entre sí, ya que provienen de la misma fuente cuya amplitud, al pasar por el divisor, se parte en dos. Se utilizaron dos divisores de haz para el arreglo (Fig. 3.3).

Figura 3.3 Divisor de haz 400-700nm, 50/50
Referencia [71].

• Espejos. Elementos básicos de re-direccionamiento luminoso, son utilizados para crear las trayectorias necesarias que llevan la luz de la fuente a los diferentes elementos experimentales utilizados. Como se muestra en la figura 4.7 se montó el interferómetro con cuatro espejos circulares de una pulgada de diámetro, dos para el brazo de referencia y dos para el brazo del objeto, de tal manera que en este último se colocará a un ángulo de incidencia sobre la muestra fuera lo más apropiado para obtener la mejor reflexión posible hacia la cámara CCD (Fig. 3.4).

Figura 3.4 Espejo
Referencia [71].

• Lente biconvexa: una lente es un sistema óptico formado por dos o más interfaces refractoras donde al menos una de éstas está curvada, las lentes que conocemos como convexas, convergentes o positivas, son más gruesas en el centro y así tienden a disminuir el radio de curvatura del frente de onda, es decir, la onda se hace más convergente conforme atraviesa la lente.

Entonces debido a que es un dispositivo que concentran los rayos de luz y forma una imagen del objeto la CCD de la cámara. Se utilizó una lente biconvexa con distancia focal de 100mm (Fig. 3.5).

Figura 3.5 Lente biconvexa
Referencia [71].

3.2 Elementos electrónicos

Originalmente, las distribuciones de HD se registraban en películas fotográficas para su posterior procesado con métodos ópticos, pero estas técnicas eran inadecuadas para la realización de mediciones en ambientes industriales.

Dispositivos de Carga Acoplada

Cámara CCD (Dispositivo de Carga Acoplada). Tal sensor se compone de un gran número de elementos fotosensibles. Durante la fase de acumulación, cada elemento recoge cargas eléctricas, que se generan por fotones absorbidos. Por lo tanto la carga recogida es proporcional a la iluminación. En la fase de lectura, estas cargas son secuencialmente transportadas a través del chip de sensor a sensor y, finalmente, convertidas a una tensión eléctrica.

Desde hace algún tiempo, los sensores de imagen CMOS han estado disponibles. Pero sólo recientemente han atraído la atención de estos dispositivos significativo porque la calidad de la imagen, especialmente la uniformidad de las sensibilidades de los elementos sensores individuales, ahora se acerca a la calidad de los sensores de imagen CCD. Los Sensores CMOS todavía no llegan a la altura de imágenes CCD en algunas funciones, especialmente en los niveles de iluminación bajos (mayor corriente oscura).

Tienen, sin embargo, una serie de ventajas significativas sobre las imágenes CCD. Ellos consumen mucha menos energía, en sub-áreas se puede acceder rápidamente, y se pueden añadir a los circuitos para el pre-procesamiento de imagen y de conversión de señal. De hecho, es posible poner toda una cámara en un solo chip (Fig. 3.6). Por último, pero no menos importante, los sensores CMOS se pueden fabricar de forma más barata y las nuevas áreas de aplicación por lo tanto abiertas.

Una vez ya descrito el funcionamiento básico de la CCD, se describen las características para el grabado de la distribución de intensidad del holograma, se utilizó una CCD Flea3 (FL3-U3-88S2C-C) con una resolución de 8.8 megapíxeles (4096x2160) de 2.5'' contando con un tamaño de pixel de 1.55µm con una interfaz de envío de datos y alimentación de energía a través de un puerto USB 3.0 de alta velocidad. Utilizada sin lente pues el plano del holograma es directamente enfocado en el área sensible de la cámara.

Figura 3.6 Cámara CCD
Referencia [72].

Computadora (PC): En este apartado se utilizó para el procesamiento digital de las imágenes capturadas, así como los videos. En cuanto a las características del equipo un procesador Intel i3 a 3.10Ghz, memoria RAM de 8Gb, Windows 7, con un disco duro de 350Gb:

El equipo utilizado contó con un óptimo rendimiento y prestaciones suficientes de procesamiento, para la adquisición de las imágenes se utilizó una herramienta muy popular MATLAB® que es utilizada para realizar cálculos matemáticos.

En la ingeniería la realización de cálculos con un programa de computación matemática como MATLAB® sustituye la programación de computadoras más tradicional, de este modo también se ha convertido en un entorno de computación técnica preferida por muchos ingenieros y científicos porque es un sistema interactivo único que incluye cómputo numérico, cómputo simbólico y visualización científica.

3.3 Plataforma de desarrollo: DE1-SoC Board

La plataforma de desarrollo utilizada es la placa DE1-SoC de la empresa Terasic® con la que se trabajó en la implementación de algoritmos en esta tesis. Ésta es una plataforma de desarrollo multipropósito, que posee una FPGA Cyclone V, y diversos periféricos que van desde simples interfaces para la interacción con el usuario como switches, pulsadores, LEDs y displays de 7 segmentos, hasta un CODEC de audio, un decodificador de video analógico, un convertidor DAC para VGA, un integrado de la capa física de Ethernet, controladores de memoria micro-SD, sólo por mencionar algunos. El diagrama de bloque de la plataforma señalando los recursos y periféricos disponibles se ilustra en la Figura 3.7.

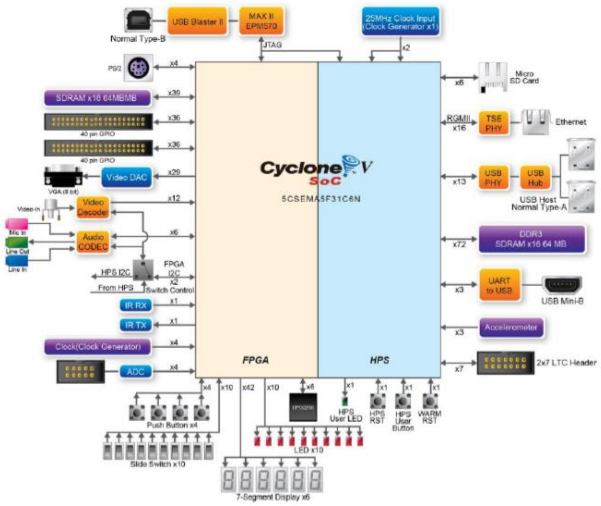


Figura 3.7 Diagrama de bloques del DE1_SoC.

Características del Cyclone V:

• Un procesador embebido en silicio con arquitectura ARM Cortex-A9® de dos núcleos, a 800 MHz, denominado por Altera como Hard Processor System (HPS). Memoria de 1GB DDR SDRAM, 1 GB Ethernet, 2 puertos USB, entrada Micro SD, UART-USB

• Una parte de lógica reconfigurable, la parte propiamente FPGA, con un equivalente de 85k elementos lógicos y 4450 Kbyte de memoria embebida, 64 MB SDRAM, Modo programación JTAG, 10 switches, Display de 7 segmentos

• Elementos de interconexión interna entre el HPS y la FPGA.

La DE1_SoC es una plataforma de diseño robusta construida por Altera, certificada para ser programada mediante OpenCL. Antes de mencionar el modo de programación se explicara que es OpenCL, para más información referencia [73].

3.4 Arreglo experimental para PSDHM

A continuación se describirá el montaje del arreglo experimental empleado para la implementación de la técnica PSDHM.

Primero se comenzó el montaje de un sistema interferométrico clásico, el interferómetro Mach-Zehnder por transmisión, el cual se modificó con el objetivo de colocar la configuración del mismo en vertical, de modo que las muestras que se utilizaran para su reconstrucción pudieran ser colocadas en forma similar que un microscopio convencional.

Como se muestra en la figura 3.8 se implementó el interferómetro de transferencia para la obtención de hologramas digitales mediante la aplicación del método de desplazamiento de fase, con el cual se obtienen 4 imágenes u hologramas, las herramientas ópticas consistieron en cuatro espejos circulares de una pulgada de diámetro, dos para el brazo de referencia y dos para el brazo del objeto.

De manera que en este último se colocara a un ángulo de incidencia sobre la muestra fuera lo más apropiado para obtener la mejor reflexión posible sobre la cámara CCD, además de la alineación adecuada de la altura del láser y espacio entre los dispositivos ópticos de tal manera de evitar que durante la manipulación del sistema se pudiera alterar de algún modo la configuración, y errores de medición. Para la trayectoria que seguirá un haz de luz coherente se eligió un láser He-Ne con una longitud de onda de ʎ=633nm de 35mW ; a la salida de este se colocó un objetivo de microscopio de 20x OB1 figura 3.2, esto con el objeto de aumentar el haz laser, colimar, a continuación se colocó una lente colimadora a una distancia focal de 50mm para aumentar el diámetro del haz de luz L1, con esto obtenemos como resultado que el área iluminada de las muestras sea mayor, un pinhole de 25µm para acentuar el diámetro del haz y eliminar ruido; se utilizaron dos divisores de haz de 50/50, el primero tiene como objetivo para dividir el haz en dos campos BS1, el primero de ellos se utiliza para obtener un campo de onda de fase constante, llamado de referencia el cual es reflejado por dos espejos M4 y M5 en él se encuentra el transductor piezoeléctrico que introducirá los desplazamientos para el método de desplazamiento de fase en $0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}, 2\pi$ finalmente este frente de onda pasa atravez del objetivo OB2.

El segundo haz es reflejado y direccionado por el espejo M2, que se utilizara para que pase a través de la muestra semitransparente colocada sobre una base con movimiento en eje (x, y e z), después un objetivo de microscopio 4x con apertura numérica de N.A=0.4 OB3 (mismas características para el OB2), y obtener un frente de onda objeto que contendrá la información de la muestra, así mismo el segundo divisor de haz BS2 será el que realizara la interferencia de los dos campos, y obtener los hologramas a la entrada del dispositivo de captura CCD cuyas características se han descrito anteriormente, un polarizador de 52mm se utilizó para atenuar la intensidad de la luz proveniente del brazo de referencia.

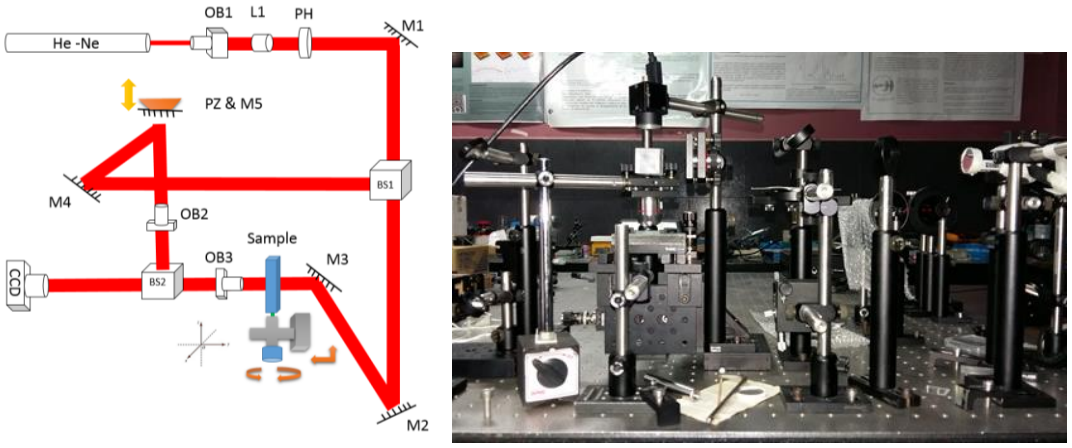


Figura 3.8 Arreglo experimental para el método de PSDHM.

3.5 Desarrollo metodológico

A continuación se hará la descripción de los pasos que se llevaron a cabo para la implementación y evaluación del algoritmo de desplazamiento de fase (PS) que fue implementado en C y posteriormente programado en OpenCL, así mismo de algoritmos para la eliminación de ruido en las imágenes como es el filtro bilateral y la transformada rápida de Fourier ya escritas en C y OpenCL, algoritmos utilizados para la reconstrucción de imágenes utilizando el FPGA.

En la implementación de la Microscopía Holográfica Digital con Desplazamiento de Pase (PSDHM), en cuanto a las imágenes utilizadas se presentaron en formato BMP, debido a que Altera SDK no soporta esas características para esta tarjeta, se tomó una librería de [74], formato utilizado para las pruebas realizadas tanto en PC como en la FPGA; la resolución de las imágenes fue 1024x1024, 512x512 y 256x256 pixeles en RGB. El procedimiento a seguir consiste en lo siguiente.

- Obtención de las imágenes. Implementación de un arreglo interferómetro experimental tipo Mach-Zehnder por transmisión para (PSDHM), en el que la adquisición será utilizando la PC y CCD.
- Evaluación del algoritmo de PS en matlab®.

- Configuración e instalación de Visual Studio 2010 y OpenCL en la PC para implementar el algoritmo de PS, filtro bilateral y la transformada rápida de Fourier escrito en lenguaje C y OpenCL, sobre las imágenes almacenadas.
- Pruebas con el FPGA y el algoritmo PS.
- Evaluación de los algoritmos en OpenCL PS, FB y la FFT en PC.
- Pruebas del FPGA con algoritmos en OpenCL del Filtro Bilateral (FB) y la Transformada Rápida de Fourier (FFT), para la eliminación de ruido en las imágenes de HD.
- Descripción del método de obtención de archivos ejecutables *.aocx con la herramienta de OpenCL de Altera.
- Resultados con el FPGA y los algoritmos PS, FB y FFT escritos en C y OpenCL, tiempos de procesamiento.

3.6 Obtención de las imágenes

Una vez ya implementado el arreglo interferómetrico se realizaron las primeras pruebas de obtención de imágenes HD, para realizar este paso se utilizó una PC en la cual se capturaron los HD con la ayuda de la CCD.

En la obtención de cada imagen se fue necesario introducir un desplazamiento de fase por medio del transductor piezoeléctrico en el orden de 0, π/2, π, 3π/2, es decir, que se obtuvieron 4 imágenes desplazadas en fase como tenemos en la figura 3.9, esto debido a la utilización de un algoritmo de desplazamiento de fase de cuatro pasos descrito en la ecuación 2.12 del capítulo anterior; en estas pruebas se utilizó un polímero con algunas imperfecciones en la superficie como muestra.

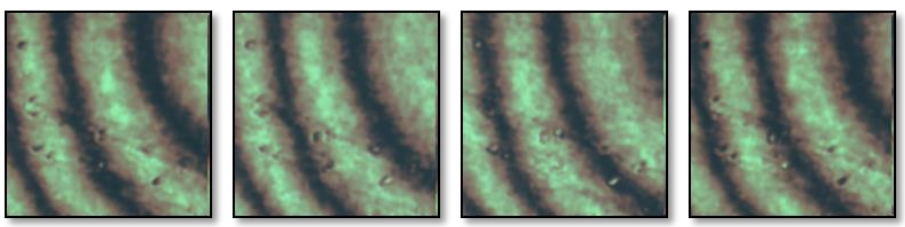


Figura 3.9 Hologramas obtenidos con desplazamiento de fase cada π/2.

3.7 Evaluación del algoritmo de PS (Phase Shift) en Matlab®

Un algoritmo sencillo para el desenvolvimiento de fase fue implementado cuyo método que describe que al considerar un mapa de fase envuelta $\Delta\varphi_{wp}(x,y)$ contiene discontinuidades cada 2π, por lo que la corrección de estas discontinuidades y obtención un mapa fase desenvuelto se describe a continuación; primero una distribución de fase envuelta en una dimensión es considerada, una variable $n(x)$ considera los valores obtenidos de la diferencia entre los valores de fase de los píxeles adyacentes, esta diferencia de valores de fase se expresa como $\Delta\varphi_{dp(x)} = \Delta\varphi_{wp}(x+1,y) - \Delta\varphi_{wp}(x,y)$, adicionando a su vez el valor de la variable $n(x)$ en $n(x-1)$, tenemos entonces en la Ec. 3.1; para cada punto $n(x)$ se almacenaran los valores que contienen los pixeles donde se localizan los saltos en 2π, esto se realiza para cada fila en x que en un rango de 1 a N puntos que se están desenvolviendo de la muestra.

$$n(x) = \Delta\varphi_{dp(x)} + n(x-1) \qquad \text{(Ec. 3.1).}$$

Finalmente cada valor entero de $n(x)$ se resta al valor de la fase envuelta $\Delta\varphi_{wp}(x,y)$; de este modo obtenemos el valor de fase desenvuelta para cada pixel en $\Delta\varphi_{uwp}(x,y)$ de una fila.

$$\Delta\varphi_{uwp}(x,y) = \Delta\varphi_{wp}(x,y) - 2\pi[n(x)] \qquad \text{(Ec. 3.2).}$$

Si graficamos la variable $n(x)$ en una fila obtendremos una función escalonada, que acumula los saltos 2π para todos los píxeles. La distribución de fase continua se calcula a continuación sustrayendo esta función escalonada a la distribución de fase no envuelta de la Ec. 3.2 con $\Delta\varphi_{wp}(x,y) - 2\pi[n(x)]$, este esquema de desenvolvimiento unidimensional puede ser transferido a dos dimensiones. Se desenvuelve primero cada fila del mapa de fase bidimensional con el algoritmo descrito anteriormente.

Los píxeles de cada fila desenvuelta actúan entonces como puntos de partida para el desenvolvimiento posterior sobre las columnas, es decir, se desenvuelve primero por filas, se aplica una operación inversa sobre estos los valores y las columnas se convertirán en filas, posteriormente se desenvolverá sobre estos valores, cabe mencionar que el algoritmo descrito es similar a los conocidos como algoritmos de seguimiento de camino (path-following).

Aplicando la formula de la ecuación 2.12 descrita en el capítulo 2; el cual es un algoritmo en el que obtenemos el mapa de fase envuelta, lo aplicamos sobre las cuatro imágenes HD obtenidas de la figura 3.10.

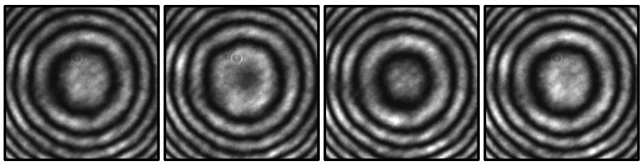


Figura 3.10 Imágenes de prueba utilizadas de 256x256 pixeles.

La información de la fase envuelta es representada como una matriz de datos en X e Y en 2D, utilizamos la función mesh () contenida matlab®, para mostrar la reconstrucción sobre las filas y columnas, y obtener una representación en 3D donde se localizan las discontinuidades cada 2π en los ejes X e Y.

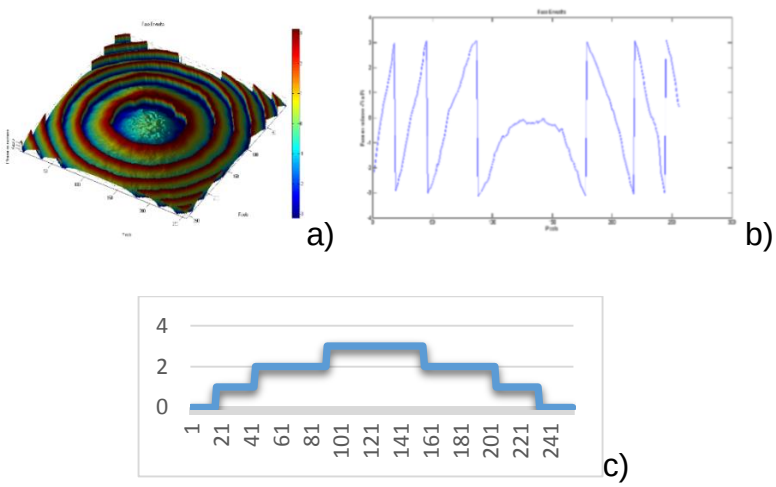


Figura 3.11 a) Fase envuelta completa, b) perfil de la imagen y c) localización de saltos.

Una vez obtenido el mapa de fase, se realiza una prueba de desenvolvimiento, se implementó el siguiente código que es un algoritmo propuesto utilizando Matlab®:

```
%%% Algoritmo de desenvolvimiento de fase
%%% IMpha=atan2(I3-I1,I4-I2)
%%% p=IMpha

%%%%%%%%%%%%%%%PASO 1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1. for x=1:i
2.    for y=2:j
3.
4.       n(y)=floor((p(x,y)-p(x,y-1))/(2*pi)+0.5)+n(y-1);
5.       phase(x,y)=p(x,y)-2*pi*n(y);
6.    end
7. end
8.
9. p=phase';
10.
11.
12. n(1)=0;
13.
%%%%%%%%%%%%%%%PASO 2%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

14. for x=1:i
15.    for y=2:j
16.       n(y)=floor((p(x,y)-p(x,y-1))/(2*pi)+0.5)+n(y-1);
17.       phase2(x,y)=p(x,y)-2*pi*n(y);
18.    end
19 end
```

Figura 3.12 Algoritmo de desenvolvimiento implementado en Matlab®.

A continuación se realiza el desenvolvimiento en las filas de la matriz de la imagen obtenida PS Figura 3.11a, indicado en las líneas del Paso 1 del algoritmo de la Figura 3.12.

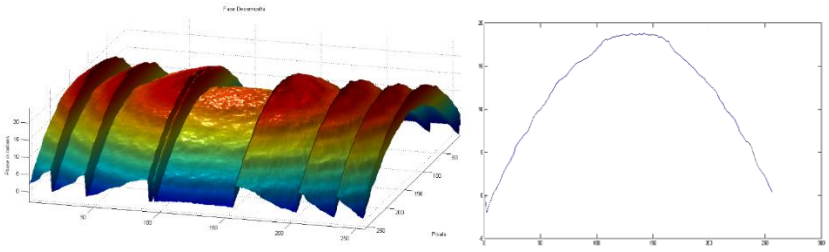


Figura 3.13 Imagen del algoritmo PS sobre las filas de la fase envuelta, sin discontinuidades en 2π para las filas.

Como parte del proceso se aplica una función traspuesta sobre las filas de la matriz y las filas serán columnas y viceversa, finalmente se aplica el Paso 2 del algoritmo Figura 3.12, obtenemos el resultado en la Figura 3.14.

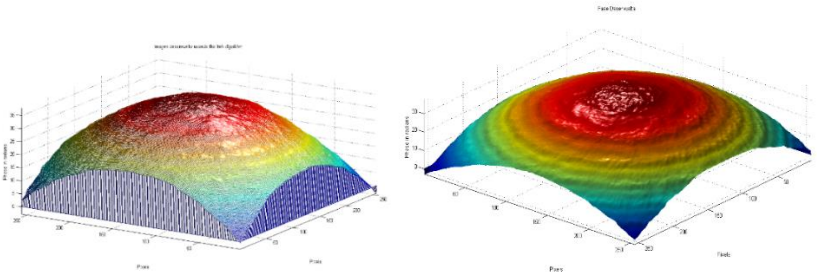


Figura 3.14 Fase Desenvuelta.

Dentro de Matlab® se encuentra una función que también realiza el desenvolvimiento de la fase, este es Unwrap () y el código para desenvolver es el siguiente, referencia [75].

```
%%% Algoritmo de desenvolvimiento de fase
%%% IMpha=atan2(I3-I1,I4-I2)
%%% p=IMpha

%%%%%%%%%%%%%%%PASO 1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

unwrappedItho = p;
for i=1:w
    unwrappedItho(i,:) = unwrap( unwrappedItho(i,:));
end

%%%%%%%%%%%%%%%PASO 2%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:w
     unwrappedItho(:,i) = unwrap( unwrappedItho(:,i));
end
```

Figura 3.15 Algoritmo función Mesh ().

Este algoritmo es más sencillo en esta función, de modo que se realizó una comparación sobre los datos de la matriz que contiene la información de la fase envuelta figura 3.14, es decir sobre las filas con el fin de corroborar que sea equivalente la información, el algoritmo propuesto a la derecha de la figura, para el desenvolvimiento de fase.

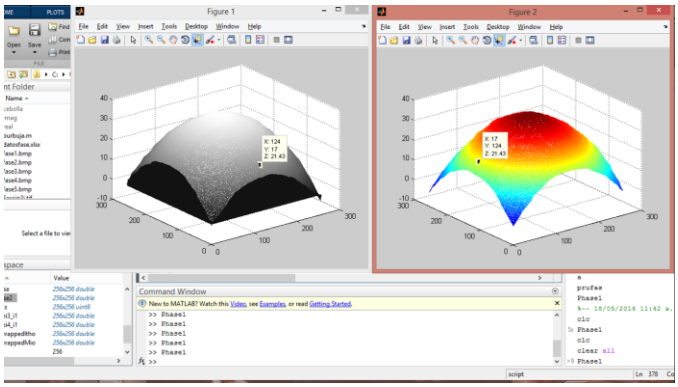


Figura 3.16 Comparación de los datos e imágenes de fase, después de aplicar la función Unwrap ().

Una vez que se compararon los datos para las filas y columnas de la imagen de fase desenvuelta se demostró similitud en los resultados obtenidos aplicando el algoritmo propuesto y la función Unwrap () de matlab®.

3.8 Configuración e instalación Software

Como se describió en secciones anteriores, una característica importante de OpenCL es su portabilidad, que permite escribir códigos en cualquier dispositivo hardware que cuente con un compilador OpenCL, de acuerdo con esto se ocupó una PC convencional para instalar OpenCL en un procesador Intel; así mismo Microsoft® Visual Studio 2010 (muy utilizado para escribir códigos en C y C++) para comparar los algoritmos escritos en C y OpenCL; en el caso se OpenCL se utilizó el mismo procesador como Host y Device para evaluar la calidad de los algoritmos sobre las imágenes de prueba; en el caso de los algoritmos escritos en código C éstos se ejecutan directamente en el mismo procesador. A continuación se describe la instalación de Microsoft® Visual Studio 2010(MVS2010) y OpenCL en una PC con las siguientes características: Sistema Operativo Windows 8.1, procesador Intel® Celeron ® N2830 con dos núcleos a 2.16GHz, a 64 bits con una memoria RAM de 4 GB.

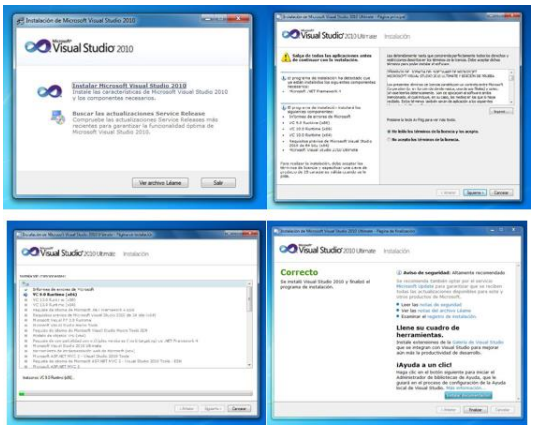


Figura 3.17 M. Visual Studio 2010

Las imágenes de la figura 3.17 corresponden a la instalación de MVS2010 cuyas herramientas y librerías son fundamentales para hacer pruebas sobre los algoritmos de PS, Filtro Bilateral y la Transformada Rápida de Fourier como herramientas de filtrado para las imágenes utilizadas. Se instaló este programa ya que de acuerdo con los requisitos de preinstalación de Altera® debemos contar con la instalación de AOCL y Quartus Prime Software compatible con el compilador C para un sistema Windows®, aunque también se puede utilizar un Sistema Operativo Linux con un compilador compatible con C, como el GCC, referencia [76].

Después se instala el OpenCL para el PC que es una versión de Intel 2015, el Intel® Media Server Studio 2017 R6 – Essentials Edition para Windows, que cuenta con la herramienta Intel® SDK for OpenCL™ Applications 2016 R3 for Windows 6.3.0.1904®, mismo programa que se vincula directamente con MVS2010, los cuales necesitamos para probar nuestros algoritmos.

CAPÍTULO 4

Resultados

4.1 Pruebas de calibración del arreglo para PSDHM.

Las primeras pruebas realizadas en PSDHM se obtuvieron imágenes correspondientes a la muestra de un polímero colocado en el arreglo experimental figura 3.8, que corresponden a las imágenes holográficas de la figura. 3.9 con desplazamiento de fase cada π/2. Para la obtención de la reconstrucción en 3D de las muestras obtenidas, se debe tomar en cuenta la compensación de la curvatura introducida por el objetivo de microscopio; para resolver este detalle, el término de aberración puede eliminarse si uno adquiere otro holograma sin el objeto en su lugar, ahora, si este holograma se resta del primero, el holograma numérico resultante tiene el campo de objeto sin aberración, en estas pruebas se utilizó una PC convencional con las características descritas en el capítulo anterior y con el uso de Matlab® (Figura 4.1).

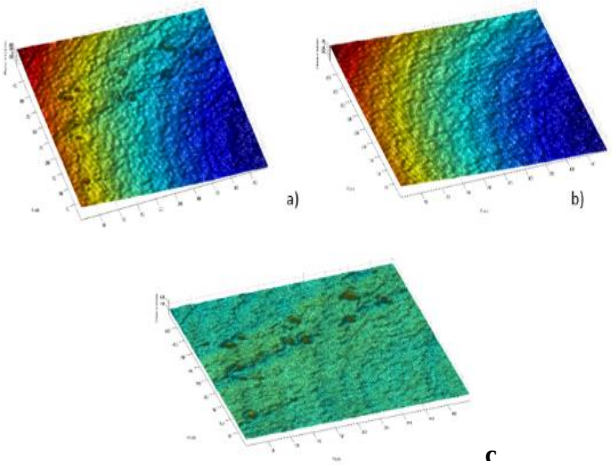


Figura 4.1 Fase desenvuelta de un polímero. a) Fase del objeto y b) fase de referencia sin compensación de curvatura, c) fase desenvuelta con compensación de curvatura.

Se realizó una segunda prueba con una muestra de aluminio (Al), previamente sumergida en una solución de $NaCl_2$ para inducirle corrosión electroquímica, observada por la formación de picaduras superficiales (figura 4.2). En esta prueba se modificó uno de los caminos ópticos (onda objeto) ya que los hologramas se obtuvieron con una configuración por reflexión.

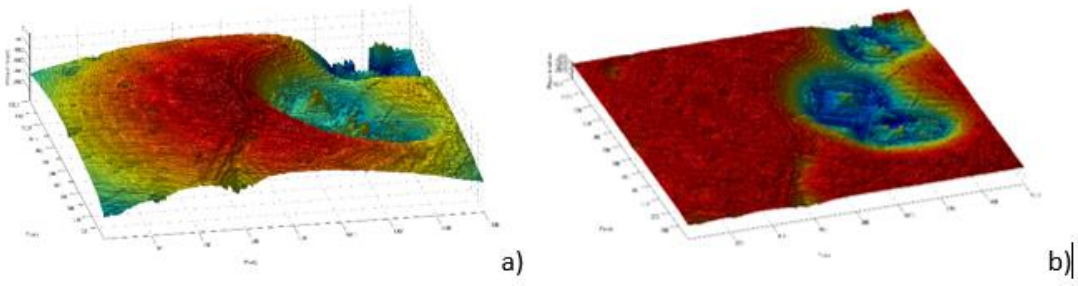


Figura 4.2 Fase desenvuelta muestra de Aluminio sumergido en $NaCl_2$. a) Fase sin compensación de curvatura, b) Fase con compensación de curvatura.

4.2 Pruebas del algoritmo PS en el FPGA

La realización de estas pruebas que consistieron en 3 etapas, la primera comenzó con evaluar el algoritmo PS en matlab® y posteriormente probarlo con el compilador de MVS2010 con código en C; la segunda etapa consistió en compilar y ejecutar el algoritmo PS en OpenCL también sobre MVS2010, finalmente como tercera etapa en el FPGA, además se realizaron pruebas de desplegado de las imágenes directamente en la tarjeta DE1_SoC que se pueden consultar en el Anexo 2. Las imágenes de la figura 4.3 se tomaron de un arreglo interferómetro HD para una sencilla prueba con espejos; a continuación las imágenes de la figura 4.4 representan las etapas que se evaluaron del funcionamiento del algoritmo PS, de esta manera obtener los primeros resultados en MVS2010.

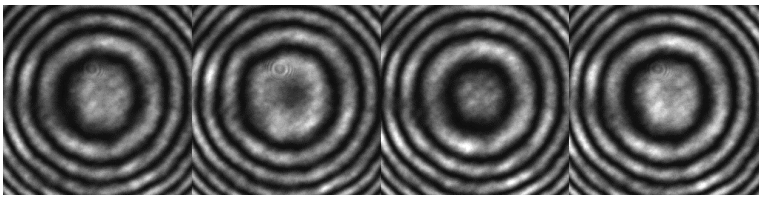


Figura 4.3 Imágenes de prueba PS.

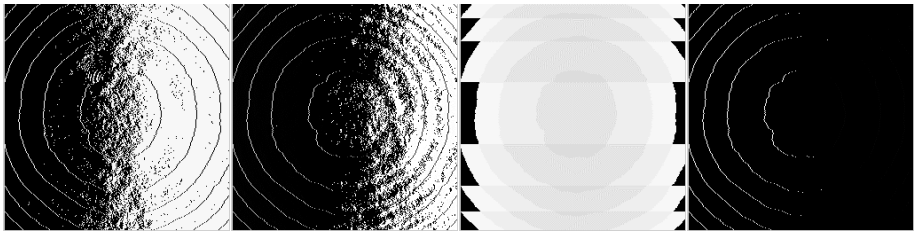


Figura 4.4. Resultados evaluación del algoritmo PS.

Una vez evaluado el algoritmo PS, obtenemos el resultado para este grupo de imágenes (Figura 4.5).

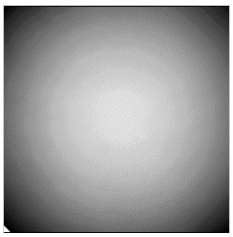
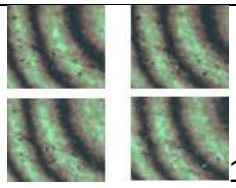
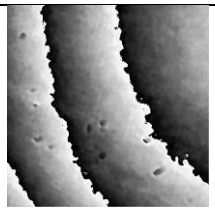
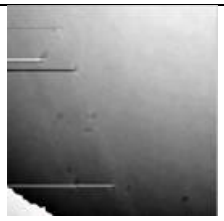
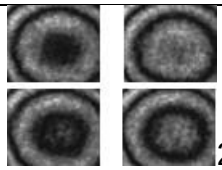
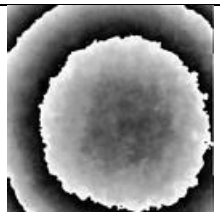
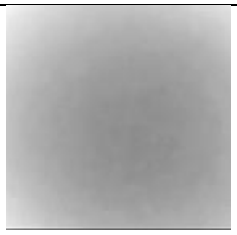


Figura.4.5. Resultado de algoritmo PS en MVS2010.

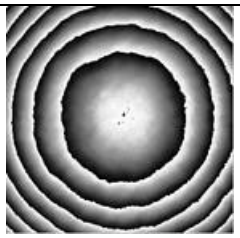
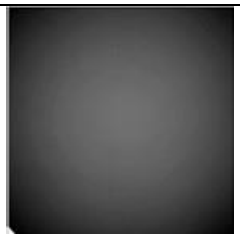
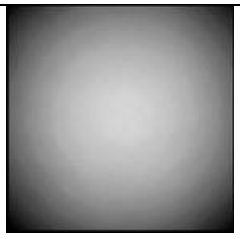
Tabla 1. Pruebas de imágenes con desplazamiento de fase.

Desplazamiento 0,π/2, π,3π/2	Fase Envuelta	Fase Desenvuelta
 1		
 2		

Se comprobó el funcionamiento del algoritmo con buena respuesta, como se muestra en las imágenes de la tabla 1, obtenidas con el arreglo interferómetro para PSDHM, en la primera fila en la fase desenvuelta se observan errores de desenvolvimiento, esto debido a que las imágenes cuentan con una superficie con irregularidades, además de que se aprecia en la parte inferior izquierda tiene una error de desenvolvimiento, lo mismo que la imagen de la figura 4.5, el cual más adelante se soluciona este problema de código.

Las imágenes en la tabla 2, corresponden a la implementación del algoritmo PS en el FPGA, la primer fila (1) corresponden a el algoritmo en C (ejecutado por el Procesador ARM), la segunda (2) corresponde al algoritmo en OpenCL (FPGA).

Tabla 2. Implementación en FPGA.

Fase Envuelta	Fase Desenvuelta
 1	
 2	

4.3 Pruebas del FPGA con algoritmos escritos en C y OpenCL del Filtro Bilateral y la Transformada Rápida de Fourier.

La guía Altera SDK para programación en OpenCL proporciona la descripción, información de recomendaciones y uso para el compilador y herramientas de Altera® Software Development Kit (SDK) for OpenCL™ (AOCL).

A continuación se describirán los primeros pasos para comenzar a hacer pruebas en la tarjeta DE1_SoC.

La tarjeta DE1_SoC es un plataforma de diseño hardware robusta con Altera System-on-Chip SoC FPGA, especialmente diseñada para trabajar con el software OpenCL. Primero se descargó una imagen DE1-SoC_openCL_BSP.zip que nos permitirá ejecutar los programas en OpenCL desde la línea de comandos de Linux, los requerimientos para configurar OpenCL en la tarjeta DE1_SoC son los siguientes:

Tabla 3. Requerimientos de la tarjeta.

Tarjeta Terasic DE1-SoC	
• Tarjeta microSD con al menos 4GB de capacidad • Lector de tarjeta microSD • Cable USB (tipo A o mini-B) • Cable Ethernet • USB Host port	• Una PC con: • 32GB de Memoria • Windows 7 64-bit o Linux • Win32 Disk Imager • La herramienta PuTTY ó Minicom(Linux) • Altera QuartusII v14.0 instalado con licencia valida. • Altera OpenCL v14.0 instalado.

En este trabajo se utilizó la versión 15.0.0.145(Quartus Prime 15.0) de Altera QuartusII y OpenCL para Windows®, ya que la versión señalada en la tabla 3, aunque funcional no genero nuestros archivos necesarios. Recordando un proyecto OpenCL consiste de un Kernel OpenCL y un programa Host mostrado en la figura 4.6, el Kernel es ejecutado en el FPGA parte de la tarjeta DE1-SoC y el programa Host en el Procesador ARM.

El programa Host se compila de manera cruzada por Altera SoC EDS, y el Kernel es desarrollado por el Quartus y el OpenCL SDK que es instalado en Windows® o Linux.

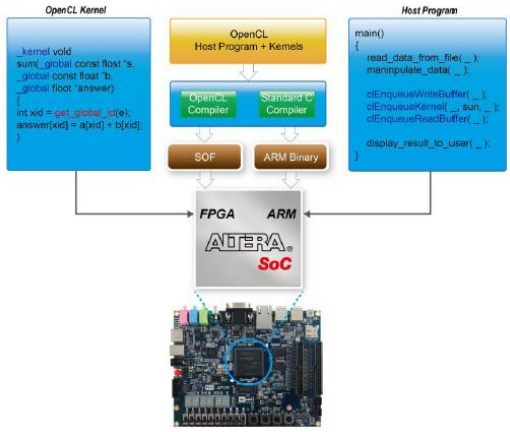


Figura.4.6. Arquitectura Altera SoC OpenCL.
Referencia [77].

4.4 Ejecución del demo sobre DE1_SoC

En esta parte se describe la prueba del demo OpenCL sobre la tarjeta DE1_SoC del archivo de imagen de Linux incluido en el paquete de soporte DE1_SoC (BSP). En primer lugar se instala sobre la PC con Windows® la siguiente paquetería.

El Disk Imager: Este programa está diseñado para escribir una imagen de disco sin formato en un dispositivo extraíble o hacer una copia de seguridad de un dispositivo extraíble en un archivo de imagen sin formato. Es muy útil para el desarrollo integrado, es decir, los proyectos de desarrollo de ARM (Android, Ubuntu en ARM, etc.).

Putty: Es un cliente SSH, Telnet, rlogin, y TCP raw con licencia libre. Disponible originalmente sólo para Windows®, ahora también está disponible en varias plataformas Unix, y se está desarrollando la versión para Mac™ OS clásico y Mac™ OS X.

Después de haber instalado estos programas solo se tiene que instalar en la memoria la imagen de Linux que contiene el OpenCL y es el DE1_SoC BSP anterior mente comentado que se descargan de la página de Altera®, en esta imagen puedes ejecutar algunos programas que ya están cargados en la misma como son el Hello_World, BoardTest, Swapper y Vector_add, que cada uno de ellos contiene tanto el ejecutable binario del Host así como el archivo *.aocx que se ejecutara en el FPGA mismo que contiene el Kernel. Es importante comentar que lo anterior también puede realizarse desde una PC que contenga un sistema operativo Linux [78].

Una vez realizado lo anterior solo hay que instalar paquetería de Altera (Quartus Prime 15.0) en Windows® lo siguiente:

- Altera Quartus II and OpenCL SDK
- Altera SoC EDS
- DE1-SoCOpenCLBoard Support Package(BSP)
- AOCL_RTE
- DSP Builder Setup
- JNEye Setup

Después de instalar y configurar los paquetes anteriores se realizó la prueba de los archivos OpenCL demos que vienen con la tarjeta como el programa ejemplo Hello_world figura 4.7.

Figura.4.7. Prueba Hello_World.

A continuación se realiza un acondicionamiento del código del Filtro Bilateral y la FFT que primero serán probados con el compilador de C y OpenCL sobre MSV2010 en el PC, cabe mencionar que el código de la FFT solo se tiene en OpenCL, así una vez que los algoritmos realizan la tarea de filtrado para la eliminación de ruido sobre las imágenes obtenidas de la técnica PSDHM serán implementados en el FPGA. A continuación se muestra el método seguido.

Para comenzar es necesario comentar que la tarjeta DE1_SoC no tiene soporte para imágenes, incluso aunque las librerías para procesamiento de imágenes de OpenCL contienen soporte para objetos de imagen 2D y 3D, Altera SDK en esta tarjeta no soporta estas características; las librerías soportan imágenes con formato *.bmp(formato de imágenes obtenidas por PSDHM) se obtuvieron en [79], hecho esto se procede a trabajar en el entorno de MVS2010 en la PC para probar el algoritmo del Filtro Bilateral(FB) y la Transformada de Fourier(FFT) (Figura 4.8).

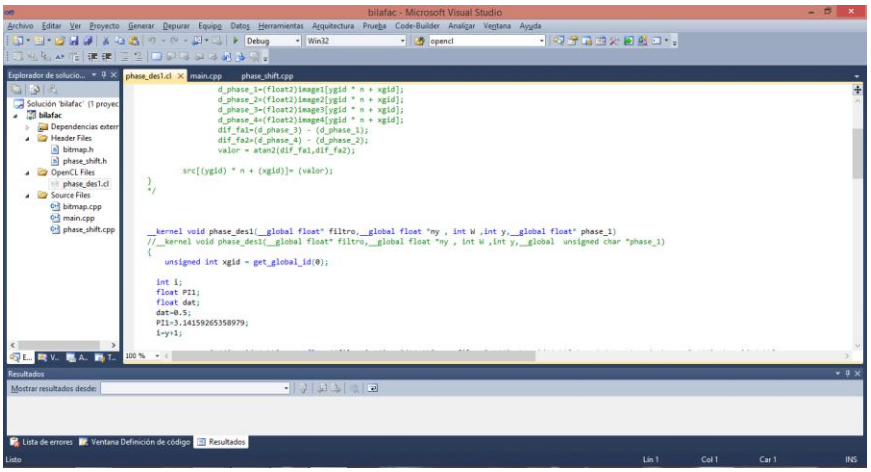


Figura.4.8. Interfaz de edición para MVS2010.

Para los primeros resultados en la implementación de los algoritmos FB Y la FFT (algoritmos tomados de la referencia [79,80]), tenemos para la imagen clásica de Lena que tiene de dimensiones 512x512 en la que se aplica el FB con un tamaño de ventana de 24 y un valor de sigma de 0.0025 (Figura 4.9), el funcionamiento de este filtro tanto los valores como las variables están descritas en el Capítulo 3.

 a)  b)

Figura 4.9. Filtro bilateral. a) Lena sin filtro, b) Lena con Filtro.

El FB suaviza y conserva bien los bordes de la imagen y actúa eliminando las diferencias atribuibles al ruido de esta manera se comprueba el funcionamiento del filtro, además de estas pruebas, otras imágenes se sometieron a este filtro con diferentes valores para la ventana de filtrado. A continuación se implementa la FFT sobre las mismas imágenes de Lena con un valor de radio de máscara de 12, tanto para el filtro pasa-bajo como para el pasa-alto (Figura 4.10).

 a)  b)

Figura 4.10. Filtro FFT. a) Pasa-Bajo y, b) Pasa-Alto.

Después de la realización de estas pruebas, se iniciaron en el FPGA. En esta parte tenemos que primero que generar un archivo ejecutable *.aocx, este es el archivo de configuración de hardware que contiene la información necesaria para su ejecución en el FPGA nuestro código-Kernel, y proviene de un archivo fuente en OpenCL con extensión *.cl que contiene nuestro código principal programado en C para OpenCL.

Para generar este archivo tenemos un programa ejecutable “AOC” (mencionado en la instalación de paquetería Altera en Windows®) en el Símbolo del Sistema (Command Prompt) que es una consola para ejecutar comando MS-DOS en Windows®.

Este programa AOC agrupa uno o más Kernels en un archivo temporal para compilarlo y entonces generar dos archivos y carpetas, un archivo Altera Offline Compiler Object (.aoco) y un archivo Altera Offline Compiler Executable (.aocx)**,** mas información consultar, Flujo de programación Altera SDK para OpenCL (anexo3).

Antes de generar nuestro primer archivo *.aocx para nuestro Filtro Bilateral, dentro del soporte que tenemos de Altera, se genera un reporte de los recursos lógicos programables (RLP) en un archivo con el nombre de nuestro kernel, producto del programa AOC que nuestra compilación arrojara, lo anterior es importante porque el código escrito en OpenCL será sometido en consideración de los RPL con que cuente la tarjeta, el caso de la DE1_SoC tenemos 85k RPL, una vez compilado nuestro archivo con la instrucción:

aoc <your_kernel_filename>**.cl** [**--report**]

Tendremos un archivo *.txt con el nombre que acordamos al archivo *.cl con el reporte siguiente (Figura 4.11).

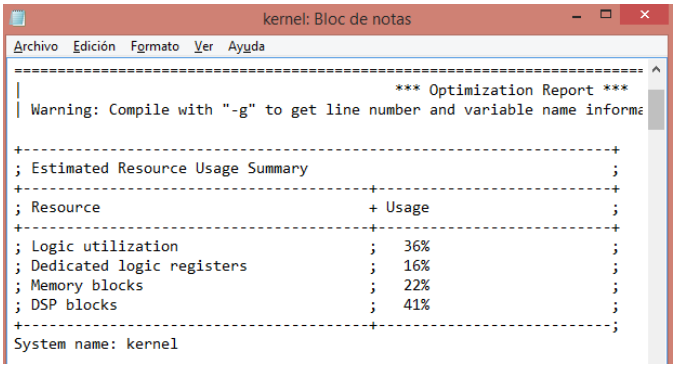


Figura 4.11. Reporte de utilización de los Recursos Lógicos Programables del DE1_SoC del FB.

Como podemos apreciar, la figura anterior es el resultado de la compilación de nuestro primer programa por AOC y la obtención del ejecutable *.aocx del Filtro Bilateral.

En primer lugar tenemos la línea que tiene más importancia para este trabajo ya que con el código adecuado podemos agregar a este 36% de RPL, el algoritmo que tenemos para PS; así bien tenemos un 64% de espacio disponible en los recursos de nuestra tarjeta DE1_SoC.

Ahora, como resultado de la utilización de recursos lógicos para la el archivo *.aocx de la FFT tenemos (Figura 4.12):

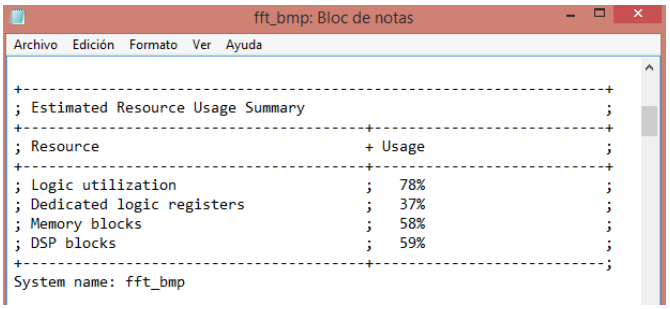


Figura 4.12. Reporte de utilización de los Recursos Lógicos Programables del DE1_SoC para la FFT.

En este caso tenemos que los RLP están al 78% por lo que tenemos que considerar que el algoritmo de PS tendrá que solo ocupar el 22% para que ocupemos todos los recursos lógicos disponibles, en todo caso tendríamos que sumar los tiempos de ejecución de estos algoritmos PS y FFT, pero lo que se pretende es que el archivo *.aocx que se genere tendrá que contener los dos algoritmos juntos el de PS con FB y PS con la FFT.

Además de estas pruebas se realizaron como primeros resultados para el FB y la FFT en el FPGA con los siguientes datos (Tablas 4 y 5).

Tabla 4. Resultados con filtros.

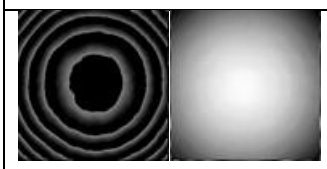
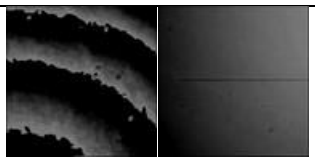
Filtro Bilateral	Filtro Fourier
	

Tabla 5. Resultados de las primeras pruebas en el DE1_SoC.

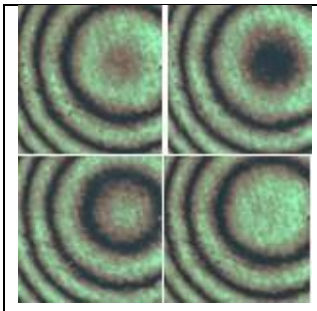
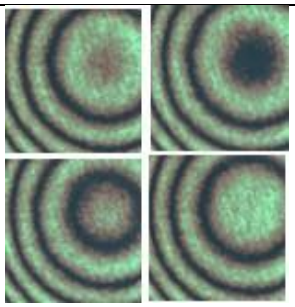
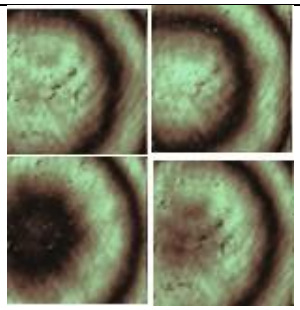
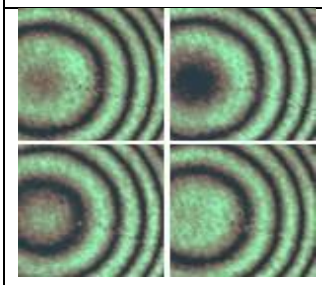
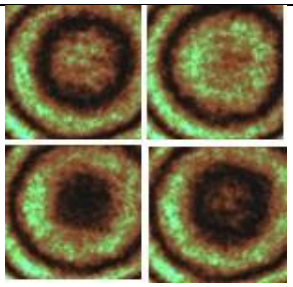
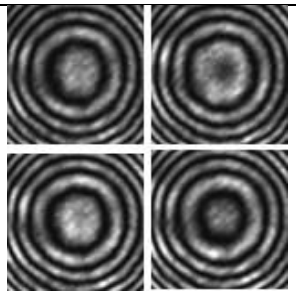
Tiempo (sec)	1 iteración	5 iteración
	Imagen 256x256	Imagen 256x256
Fase Desenvuelta ARM	0.162	0.676
Fase Desenvuelta - Filtro Bilateral ARM	9.818	47.409
Fase Desenvuelta - Filtro Bilateral FPGA OpenCL	2.053	10.271
	Imagen 499x499	Imagen 499x499
Fase Desenvuelta ARM	0.636	2.649
Fase Desenvuelta - Filtro Bilateral ARM	36.756	182.758
Fase Desenvuelta - Filtro Bilateral FPGA OpenCL	3.838	11.856
	Imagen 1024x1024	Imagen 1024x1024
Fase Desenvuelta ARM	2.6	10.841
Fase Desenvuelta - Filtro Bilateral ARM	152.158	752.839
Fase Desenvuelta - Filtro Bilateral FPGA OpenCL	7.702	49.369
Fase Desenvuelta - Filtro Fourier FPGA OpenCL	11.816	NULL

4.5 Implementación en el FPGA con algoritmos en OpenCL de PS, FB y la FFT como filtros para la eliminación de ruido en las imágenes de HD.

En la implementación se utilizaron hologramas obtenidos por PSDHM, usando algoritmos compilados y ejecutados en MVS2010. El objetivo fue filtrar los errores de desenvolvimiento en las imágenes debido al ruido generado por irregularidades superficiales en las muestras, aplicando los filtros FB o FFT según fuera el caso, y mejorar la representación de la fase desenvuelta.

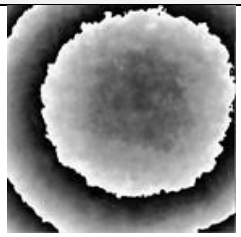
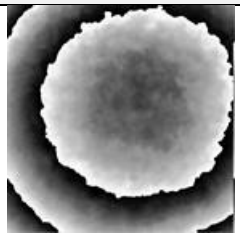
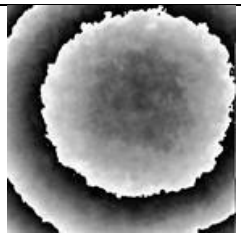
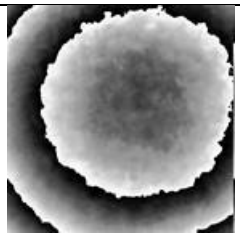
Concluidas las pruebas en MVS2010, los algoritmos se adaptaron para ejecutarlos en el FPGA. A continuación se enlistan los grupos de imágenes holográficas obtenidas por PSDHM, que se utilizaron en las pruebas para MVS2010 y en el FPGA.

Tabla 6. Grupo de imágenes de prueba.

 Grupo 1	 Grupo 2	 Grupo 3
 Grupo 4	 Grupo 5	 Grupo 6

Enseguida se muestran algunos resultados realizados en MSV2010.

Tabla 7. Resultados en la implementación del Filtro Bilateral en C y OpenCL para MVS2010 en PC.

Grupo 5 Filtro Bilateral en C		
Grupo 5 Filtro Bilateral en OpenCL		

En la Tabla 7 se muestran los resultados obtenidos de los algoritmos del FB y PS en C y OpenCL en la PC. Es necesario comentar que el primer procedimiento fue aplicar el filtrado sobre los datos de la imagen de la fase envuelta, lo cual no arrojó buenos resultados, esto debido a que la información de la fase envuelta era modificada es decir que se perdía mucha información, la imagen era demasiado suavizada por el filtro; por lo tanto las pruebas del filtrado se realizaron sobre cada uno de los cuatro hologramas variando el valor de la ventana de filtrado para el FB. Lo anterior generaría un pequeño aumento en los tiempos de procesamiento ya que el filtro sería sobre cada una de las imágenes HD utilizadas y no como se pretendía hacer con los datos de la fase envuelta; sin embargo el desempeño de los algoritmos del FB sobre las imágenes tanto para C y OpenCL muestra resultados muy similares y aceptables en cuanto a calidad. La Figura 4.13 representa la secuencia de filtrado; para continuar se procede a hacer pruebas con la aplicación de la FFT.

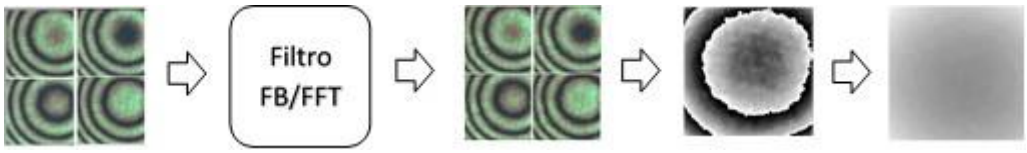
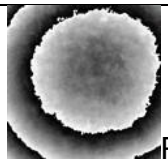
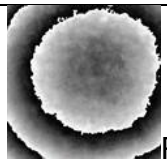
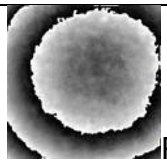
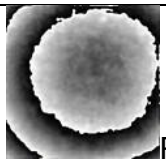
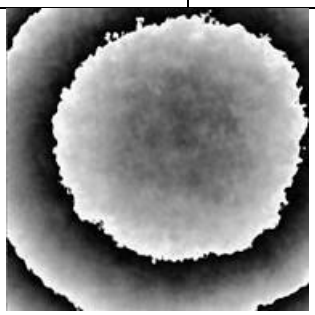
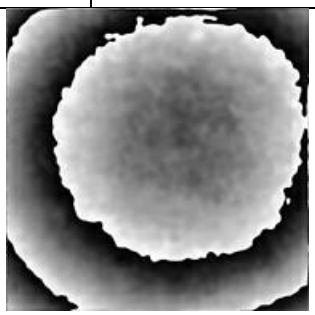


Figura.4.13. Secuencia de filtrado.

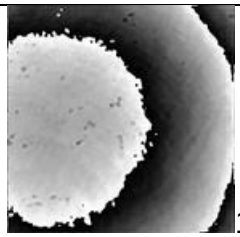
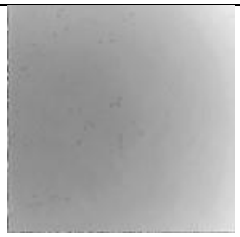
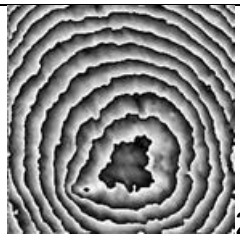
En la Tabla 8 se aprecia la aplicación de la FFT, en la que sólo se modificó el tamaño del radio que utiliza la máscara con valores utilizados en un rango de 2 a 32. Aunque las imágenes de prueba no presentan mucho ruido visible, este algoritmo se aplicará posteriormente a imágenes con más ruido en la superficie.

Tabla 8. Implementación de la FFT en MVS2010.

 R=2	 R=12	 R=22	 R=32
Grupo 5 Filtro Fourier			

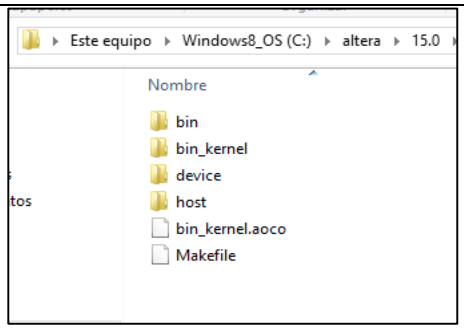
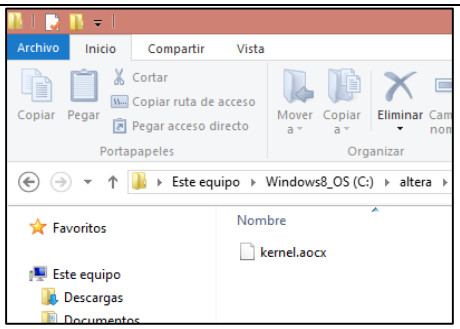
El resultado de la prueba para la FFT de la Tabla 9, indica buen desempeño para otro tipo de imágenes con más ruido presente. Es importante recordar que tanto el filtro FB y la FFT el archivo generado (*.cl) agrupará uno o más Kernels de configuración de hardware en el mismo archivo, esto es porque el archivo *.aocx generado por AOC deberá contener dos algoritmos tanto para el filtro FB como la FFT contendrán el algoritmo PS, claro que esto estará determinado por el uso de los RPL den como resultado su utilización en la DE1_SoC.

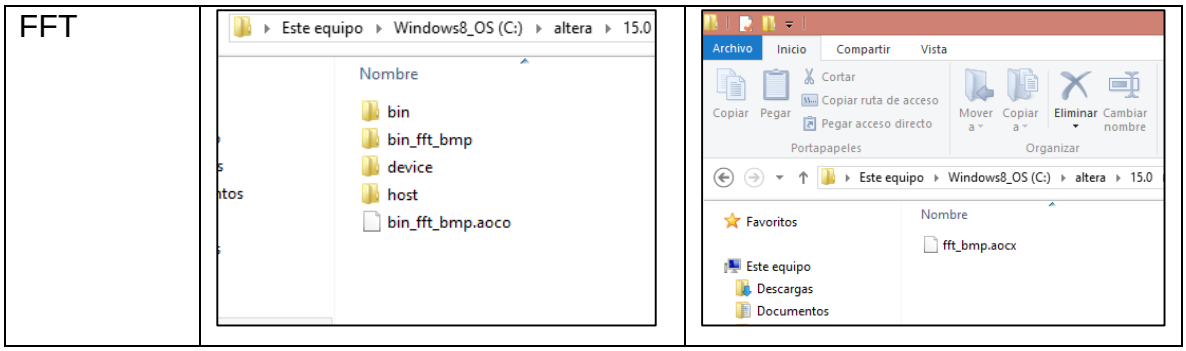
Tabla 9. Filtro Fourier en MVS2010 en PC.

Fase envuelta	Fase Desenvuelta/Sin Filtro	Fase Desenvuelta/Filtrada
 1)		
 2)		

Comprobados los algoritmos en MVS2010 en PC, se generarán los archivos *.aocx (Tabla 10), que contienen la configuración de hardware para el filtrado de nuestras imágenes en el FPGA (Figura. 4.14).

Tabla 10. Archivos *.aocx, del Filtro bilateral y Transformada Rápida de Fourier.

Filtro Bilateral		



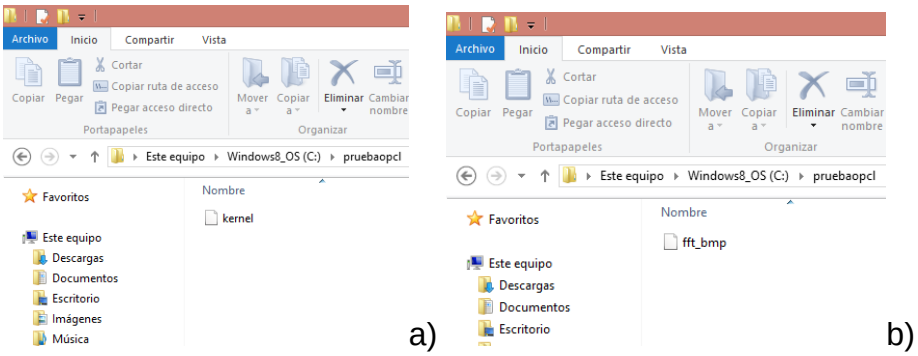


Figura 4.14. Archivos *aocx generados.

Recordando que OpenCL trabaja con una archivo Host que controla las instrucciones y la comunicación de los datos que seran procesados por el FPGA, ademas del archivo *.aocx , también es necesario generar un archivo ejecutable por parte del host, es decir que tenderemos un arhivo *.aocx que se ejecutara en el FPGA(Device) y un archivo Host que se ejecutara en el procesador ARM de la tarjeta DE1_SoC.

Una vez contando con estos dos archivos se comenzo con las pruebas del filtro FB figura 4.15, tenemos que el tiempo para una iteración en una imagen de 256x256 pixeles del grupo 6, es decir para la aplicación del FB con PS es de 2.053s, el tamaño de ventana para esta prueba fue de 5, y un valor de sigma de 0.0025.

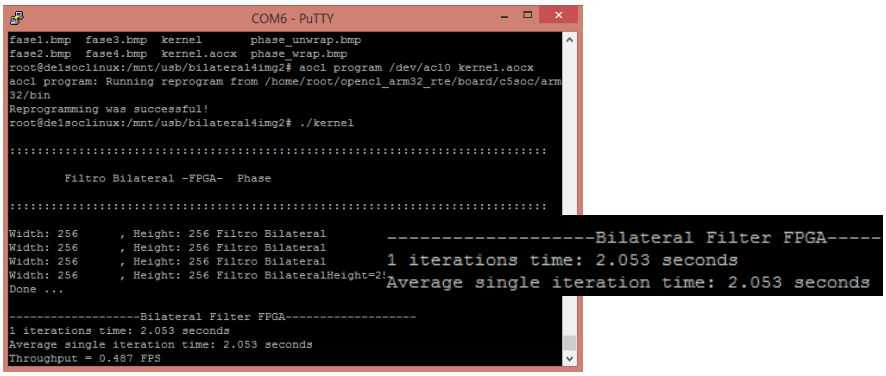


Figura 4.15. Tiempo resultante para implementar el FB en el FPGA.

A continuación se realizó la prueba para la FFT con PS en el FPGA con una imagen de 1024x1024 pixeles. El resultado para una iteración es de 11.816s en la se utilizó una radio de máscara de 2.

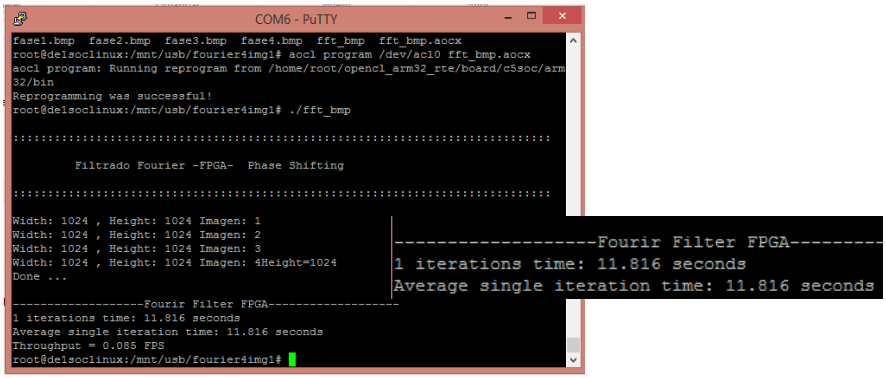


Figura.4.16. Tiempo en la implementacion de la FFT en el FPGA.

Los resultados en la implementacion del filtro de la FFT y FB fueron favorables en los tiempos de procesamiento posteriormente se realizaran pruebas con diferentes tamaños para las imágenes HD utilizadas. En un ultimo dato los resultados de los recursos RLP utilizados para generar el archivo *.aocx de nuestro algoritmo PS, implementado con el FB y la FFT para las pruebas anteriores semuestra en la figura 4.17 un reporte, el cual muestra que los recursos utilizados son del 64% muy util ya que el FB genero una utilizacion del 36% por lo teoricamente tenemos al 100% de RLP(sin embargo una posterior optimizacion final se logro un 82% de RLP en conjunto) es decir que el archivo final *.aocx que contendra los algoritmos de FB y PS cumplio con los requisitos para implementacion.

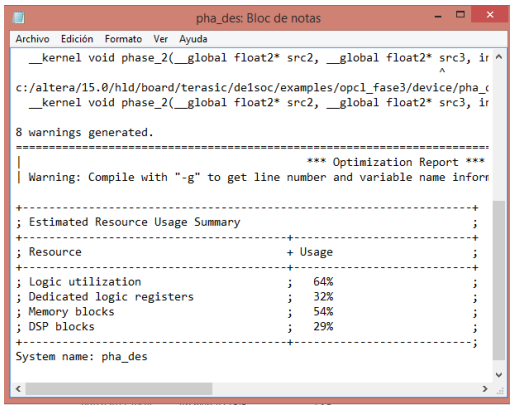


Figura 4.17 Reporte RLP del algoritmo PS.

4.6 Resultados con el FPGA y los algoritmos PS, FB y FFT escritos en C y OpenCL, tiempos de procesamiento.

Una vez que los algoritmos OpenCL fueron probados como herramientas de filtrado y desenvolvimiento de fase, se calcularon sus tiempos de procesamiento, es decir, el tiempo que puede obtener un mapa de fase desenvuelta a partir de una secuencia de filtrado.

Para este propósito, ambos procedimientos de filtrado y desenvolvimiento de fase se combinaron en un nuevo algoritmo, y sus tiempos de ejecución se calcularon de acuerdo con el número de iteraciones necesarias para obtener desenvolvimiento adecuado dependiendo de la cantidad de ruido que presentaran los interferogramas. El tiempo resultante (tiempo en segundos, utilizando la función getCurrentTimestamp de AOCLUtils) se calculó de acuerdo con los seis procesos que se describen a continuación, utilizando el procesador ARM y FPGA. Se utilizaron cuatro grupos de cuatro imágenes (hologramas) desplazadas en π/2, interferogramas previamente obtenidos por PSDHM de 24 bits RGB (figura 4.18). Los grupos 1 y 3 corresponden a imágenes grabadas de muestras de películas de polímeros (en pruebas separadas); segundo grupo son imágenes de referencia, es decir, sin muestra en el haz del objeto, y el cuarto conjunto se registró a partir de la muestra de microesferas (Grupo 4) de 10µm aproximadamente. Las imágenes se han ajustado a 256x256, 512x512 y 1024x1024 píxeles, tal como se indica.

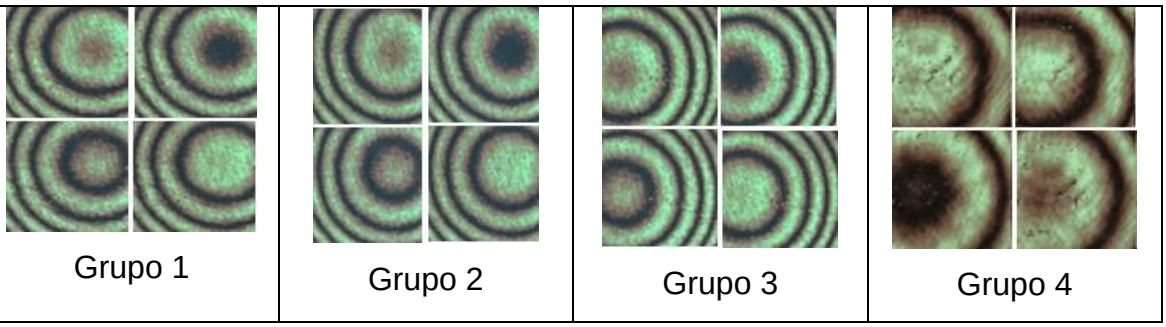


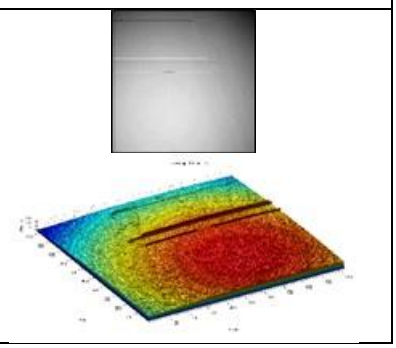
Figura 4.18. Conjuntos de imágenes desplazadas π/2, 24 bits RGB obtenidas por PSDHM; Los grupos 1 y 3 corresponden a la película de polímero, el grupo 2 son interferogramas de referencia y el grupo 4 corresponde a microesferas.

En las Tablas 11-15, los tiempos de desenvolvimiento mencionados anteriormente se ordenan de acuerdo con los procesos correspondientes, mostrados por pares en cada tabla. Además, se incluye información de las tareas realizadas por el procesador ARM y la tarjeta FPGA por separado, los grupos de imágenes PSDHM considerados y el tamaño de los interferogramas.

Aun lado de cada tabla, las imágenes 2D de las distribuciones de fase desenvueltas obtenidas con la tarjeta FPGA; adicionalmente se utilizó un software de aplicación (Matlab®) para generar gráficas de representación en 3D, de las fases desenvueltas.

Proceso 1: En este proceso se generó un algoritmo PS similar en C y en OpenCL (ejecutados para el procesador ARM y el FPGA), por separado, se aplicó en los interferogramas de los grupos 1 a 2 (Figura 4.18), es decir se realizaron dos pruebas; los resultados se muestran en la Tabla 11. Como se puede ver, aunque el tiempo de ejecución del desenvolvimiento usando el ARM es más corto que el obtenido con FPGA, podemos decir que todos los tiempos son generalmente en orden aceptable para realizar las tareas indicadas ya que la evaluación del tiempo será importante en cuanto se prueben con los dos algoritmos el de filtrado y el de desenvolvimiento, como vemos los errores se observan para la representación de la película del polímero en su fase desenvuelta reconstruida.

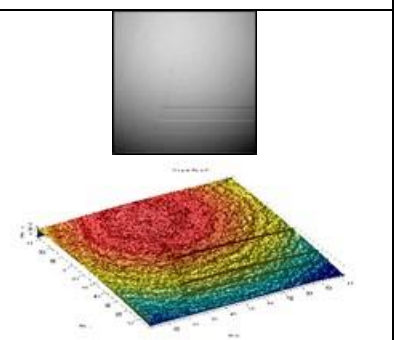
Tabla 11. Tiempos de desenvolvimiento de fase obtenido proceso 1 y representación de la fase desenvuelta.

Proceso 1		Grupo 1 Tiempo/s	Grupo 2 Tiempo/s	Represetacion 2D y 3D en Matlab® abajo
C ARM	1024x1024	1.92	2.716	
	512x512	0.469	0.492	
	256x256	0.117	0.12	
OpenCL FPGA	1024x1024	5.473	5.401	
	512x512	1.725	1.734	
	256x256	0.794	0.782	

Proceso 2: Para el desenvolvimiento de la fase se utilizó el conjunto de imágenes del Grupo 3 (Figura 4.18), se combinó con la aplicación previa del filtro Bilateral, el tamaño de ventana para esta prueba fue de 5, y un valor de sigma de 0.0025, para una sola prueba. Ambas tareas fueron realizadas por el procesador ARM y el FPGA por separado, siguiendo la secuencia de filtrado descrita en la figura 4.13.

Podemos ver cómo el desenvolvimiento de tiempos de procesamiento de fase en una imagen de 1024x1024 píxeles es 12X más rápido usando OpenCL en el FPGA, en lugar de C sobre el procesador ARM, 9X y 3X más rápido para otros tamaños de imagen (Tabla 12). Aunque se ha introducido el filtrado, los errores en la reconstrucción de la imagen de referencia son aún se aprecian, para 1 iteración.

Tabla 12. Tiempos de desenvolvimiento de fase obtenido proceso 2 y representación de la fase desenvuelta.

Proceso 2		Grupo 3 Tiempo/s	Tamaño MB	Represetación 2D y 3D en Matlab® abajo
Filtro Bilateral + C ARM	1024x1024	161.488	3MB	
	512x512	41.552	768KB	
	256x256	10.289	192KB	
Filtro Bilateral + OpenCL FPGA	1024x1024	13.389	3MB	
	512x512	4.727	768KB	
	256x256	2.978	192KB	

Proceso 3: Una forma de reducir el ruido y obtener imágenes sin envolver con menos errores fue aplicando el procedimiento de filtrado bilateral repetidamente (es decir, realizando iteraciones) en las imágenes HI obtenidas. Como se indica en la Tabla 13, se aplicaron 4 iteraciones con un tamaño de ventana de 5, prueba 1 (columna azul) y una prueba 2 de 1 iteración (columna naranja), para un tamaño de ventana de 10, ambas pruebas sobre las imágenes del Grupo 3 (Figura 4.18), sobre 3 tamaños de imagen diferentes y el mismo valor de σ = 0.0025. Se puede obtener una notoria mejora en los tiempos de procesamiento para FPGA en comparación del procesador ARM.

Por ejemplo, el procesamiento fue 10 veces más rápido para imágenes de 1024x1024 usando 4 iteraciones en la prueba 1 y ≈30 veces más rápido que el procesador ARM para el mismo tamaño de imagen con solo 1 iteración en la prueba 2. También observamos cómo no hay diferencia significativa en los tiempos con 4 o 1 iteraciones usando el procesador ARM, aun cambiando el tamaño de la ventana; con FPGA, el número de iteraciones informa diferencias claras en la velocidad de procesamiento al solo cambiar la ventana.

Una mejora evidente en cuanto a la representación (ausencia de errores) de la reconstrucción de la película de polímero se muestra en las Figuras 4.19, usando más iteraciones (4 iteraciones, Figura 4.19 (a)) o cambiando el tamaño de filtración de la ventana (de 5 a 10, Figura.4.19 (b)), en comparación con su representación de la misma muestra de la tabla 12 anterior. No se aprecian diferencias muy significativas entre las Figs.4.19 (a) y (b) al cambiar las iteraciones o el tamaño de la ventana ya que no se observan errores en el desenvolvimiento.

Tabla 13 Tiempo de desenvolvimiento de fase obtenidos, proceso 3.

Proceso 3		Grupo 3 4 it Tiempo/s	Grupo 3 1 it Tiempo/s	Tamaño en MB
Filtro Bilateral + iteraciones+ Desenvolvimiento de fase, código en C para el P. ARM	1024x1024	638.928	673.974	3MB
	512x512	162.557	166.496	769KB
	256x256	44.232	43.868	193KB
Filtro Bilateral + iteraciones+ Desenvolvimiento de fase, código OpenCL en FPGA	1024x1024	62.411	22.548	3MB
	512x512	24.384	7.544	769KB
	256x256	19.521	3.74	193KB

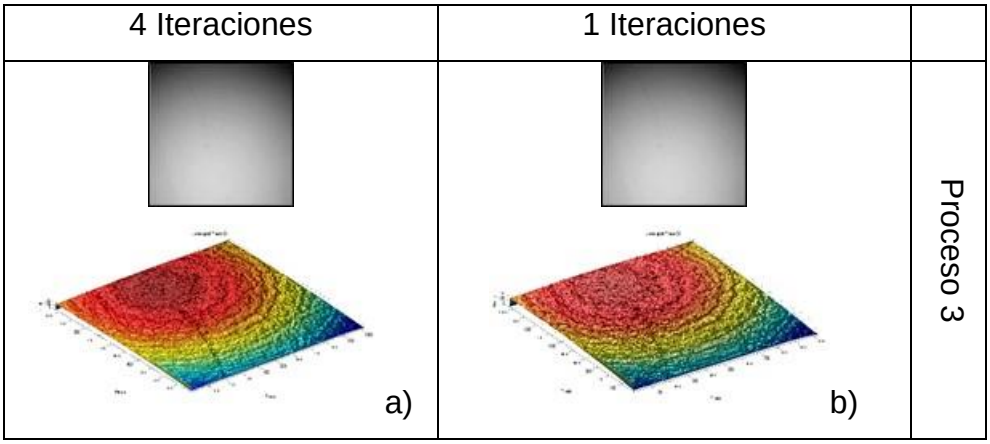


Figura 4.19. Gráficas 2D y de superficie 3D en Matlab® representando las distribuciones de fase desenvueltas. Película de polímero

Proceso 4: Se realizó una comparación en tiempo/s, entre los filtros Fourier y Bilateral sobre el FPGA usando las imágenes HD contenidas en el Grupo 4 (Figura 4.18), realizando 2 pruebas columna azul prueba 1 y columna naranja prueba 2(para la prueba 1 el radio de la máscara en el filtro FFT fue de 2, en el FB la ventana de 5; en la prueba 2 el radio del filtro FFT se aumentó a 22 y para el FB la ventana a 10).

Ambos filtros compilados y obtenidos en OpenCL; cabe mencionar que para el procedimiento de desenvolvimiento se utilizó el algoritmo programado en C (es decir que los algoritmos de filtrado fueron en OpenCL y el desenvolvimiento de la fase en C); esto se debe a que en la compilación y obtención del archivo ejecutable (kernel) *.aocx necesario para el filtro FFT, los recursos en cuanto a elementos lógicos de la tarjeta no fueron suficientes, como se describe en el reporte obtenido en la figura 4.12 anterior que consumían el 78% del total con el que cuenta esta tarjeta FPGA.

Los resultados en tiempo de procesamiento que se muestran en la Tabla 14 son ligeramente similares comparando al filtrado de Fourier y el FB para una primera prueba con sombreado azul, por ejemplo en los casos de la imagen de 1024x1024 con 9.811s en el filtrado FFT y 12.199s para el FB respectivamente, en una imagen de 512x512 tenemos 3.745s para la FFT y 2.923s para el FB.

Tabla 14. Tiempo de desenvolvimiento de fase, proceso 4.

Proceso 4		Grupo 4 Tiempo/s	Grupo 4 Tiempo/s	Tamaño en MB
Filtro Fourier en OpenCL + Desenvolvimiento de fase, código C sobre FPGA	1024x1024	**9.811**	9.907	3MB
	512x512	**3.745**	3.654	769KB
	256x256	**2.199**	2.126	193KB
Filtro Bilateral en OpenCL + Desenvolvimiento de fase, código C sobre FPGA	1024x1024	**12.737**	**39.359**	3MB
	512x512	**2.923**	10.887	769KB
	256x256	**1.874**	3.654	193KB

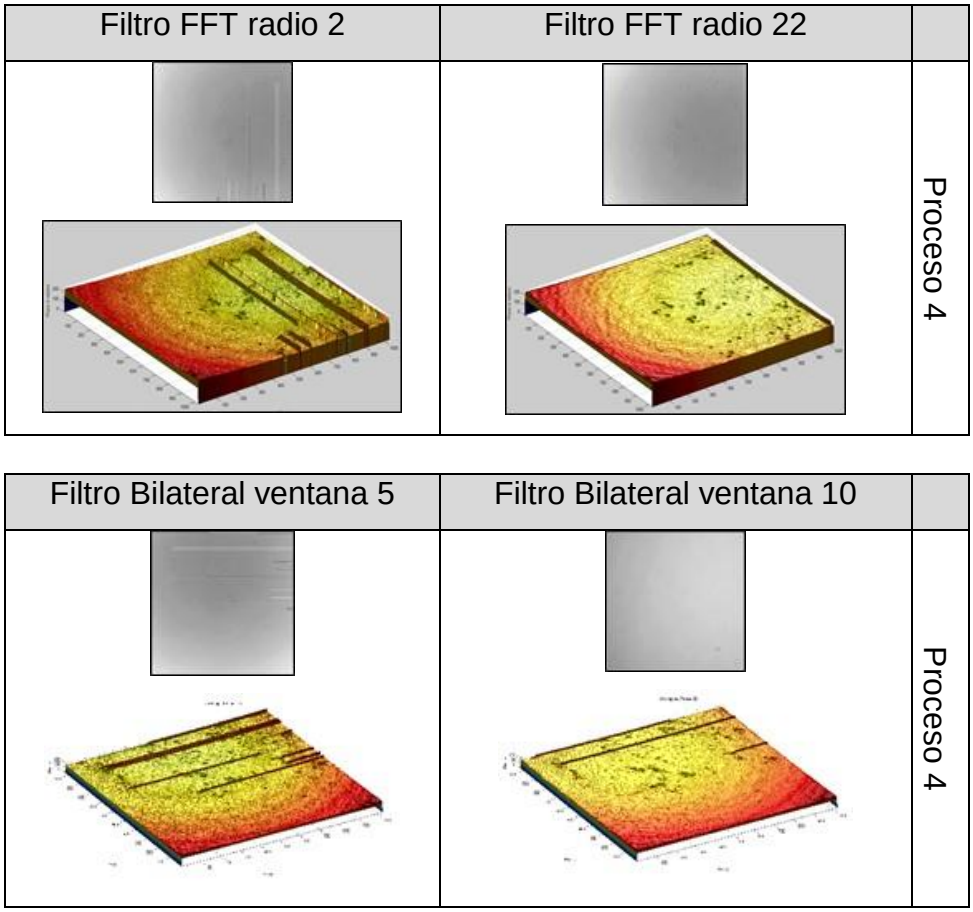


Figura 4.20. Gráficas 2D y de superficie 3D en Matlab® representando las distribuciones de fase desenvueltas para el proceso 4. Esferas microcristalinas.

Una observación importante en el Proceso 4 es que, aunque para el filtrado de Fourier, el radio de la máscara se inició con 2 en la primera prueba (columna azul) a 22 en columna naranja, no hubo cambios apreciables que afectaran en el tiempo de procesamiento de 9.811s y 9.907s.

Como se puede observar con el filtrado Bilateral, donde el tamaño de la ventana de filtrado se cambió de 5 en la primer prueba, a 10 en la segunda respectivamente, el tiempo de procesamiento se vio claramente afectado ya que aumenta 3 veces, aproximadamente (por ejemplo la imagen de 1024x1024 con el filtro bilateral con ventana 5 tenemos un tiempo de 12.737s y al aumentar la ventana a 10 tenemos un tiempo de 39.359s). Como resultado, podemos decir que el filtrado de Fourier conserva casi los mismos tiempos de procesamiento para ambos valores de radio de máscara, este procedimiento es más eficiente que el filtrado bilateral, ya que realiza menos operaciones durante este proceso, en comparación con el aumento del tamaño de la ventana de filtrado para FB el número de cálculos se incrementa.
Con respecto a la representación de la fase desenvuelta, es evidente una mejora en la eliminación de errores del estado inicial figura 4.20 mediante el uso del filtrado de Fourier. Aunque hay una ligera mejora en el filtrado, el resultado del FB no fue tan notable.

Finalmente se muestran los resultados para dos últimas pruebas utilizando la compensación de la aberración sobre la fase desenvuelta obtenida de la imagen del objeto y referencia respectivamente, en estas pruebas solo se utilizó el Filtro Bilateral como herramienta de filtrado.

Proceso 5: Un algoritmo implementado para la compensación de aberración, comúnmente presente durante la reconstrucción holográfica [80-82], se implementó en base al concepto de placa de corrección, donde la resta entre las fases desenvueltas de la imagen del objeto (película de polímero, Grupo 1) y la imagen de referencia (Grupo 2) se realiza. Esto es importante tenerlo en cuenta por qué se debe considerar la compensación de las curvaturas de frente de onda introducidas por el sistema de lentes PSDHM durante el proceso de obtención imágenes, especialmente para aplicaciones metrológicas microscópicas.

En la Tabla 15 se muestra una comparación entre los tiempos de procesamiento que utilizan la compensación de aberración combinada solo con algoritmos de filtrado Bilateral y desenvolvimiento de fase, con un tamaño de ventana de filtrado de 5, implementado en OpenCL y C por separado en el FPGA (el proceso de desenvolvimiento, compensación de la aberración y el filtrado FB se implementaron en el procesador ARM (en C) y sobre el FPGA(en OpenCL) en dos pruebas (columna azul prueba 1 con una iteración y prueba 2 columna naranja con 3 iteraciones).
Para un tamaño de imagen de 1024x1024, el FPGA muestra que es casi 8x veces más rápido que el procesador ARM para realizar las tareas descritas. La prueba 2 adicional para eliminar el ruido con 3 iteraciones mostraron que para el caso del algoritmo C, el tiempo de procesamiento resultó demasiado largo (≈ 489s); se realizaron pruebas adicionales con más iteraciones pero no dieron resultados favorables, probablemente debido a un consumo excesivo de memoria relacionado con la limitación de las características de la tarjeta.

Las representaciones correspondientes de la fase desenvuelta de la película de polímero se muestran en las Figuras 4.21 (a y b), para 1 iteración y 3 iteraciones respectivamente; aunque no hay diferencias apreciables entre ellos en cuanto a su representación, muestran una corrección en la curvatura del frente de onda en comparación con su representación anterior para la misma muestra del polímero representado en la tabla 11.

Proceso 6: Se realizó una prueba final considerando la compensación de aberración, con un ajuste a un tamaño de ventana de filtrado FB de 10 para 1 iteración. Las imágenes HD de los Grupos 1-2 se usaron nuevamente. Se obtuvo un tiempo de procesamiento de FPGA 23x veces más rápido que el uso del ARM para un tamaño de 1024x1024, y también se obtuvieron mejoras notables para tamaños de 512x512 (≈17X) y 256x256 (≈10X).

Esta representación particular de la fase desenvuelta del polímero, muestra tanto la corrección en la curvatura del frente de onda así como una reconstrucción limpia del ruido (Figura 4.21 (c)), con respecto al proceso 5 que aun presentaba errores de desenvolvimiento. Tenemos un resultado aceptable usando el FB, siguiendo la combinación de tareas descritas.

Tabla 15. Tiempos de desenvolvimiento de fase obtenidos en los procesos 5 y 6, como se describe en el texto.

Proceso 5		Grupo 1,2 1 it Tiempo/s	Grupo 1,2 3 it Tiempo/s	**Proceso 6**		Grupo 1,2 1 it Tiempo/s	Tamaño
Compensación de Aberración + Filtro Bilateral + Ventana 5 C ARM	1024x1024	173.356	489.707	Compensación de Aberración + Filtro Bilateral + Ventana 10 C ARM	1024x1024	673.769	3MB
	512x512	50.893	119.644		512x512	162.372	768KB
	256x256	11.368	30.784		256x256	39.545	192KB
Compensación de Aberración + Filtro Bilateral + Ventana 5 OpenCL FPGA	1024x1024	22.467	76.834	Compensación de Aberración + Filtro Bilateral + Ventana 10 OpenCL FPGA	1024x1024	29.503	3MB
	512x512	7.349	26.592		512x512	9.584	768KB
	256x256	3.634	12.006		256x256	3.962	192KB

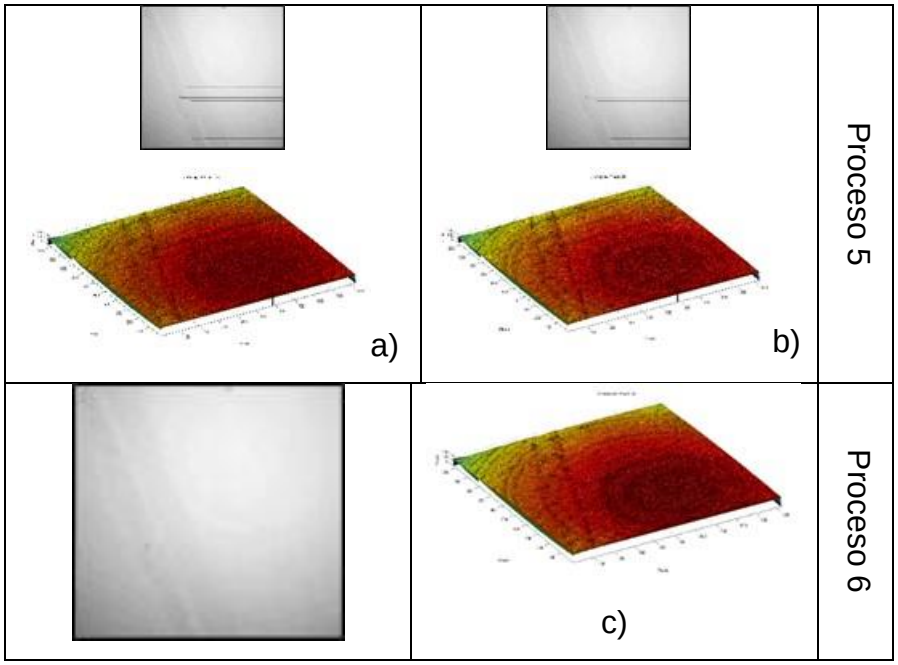


Figura 4.21. Gráficas 2D y de superficie 3D en Matlab® representando las distribuciones de fase desenvueltas para la película de polímero. Procesos 5: (a) 1 iteración, (b) 3 iteraciones. Proceso 6: (c) 1 iteración.

Conclusiones y Trabajo Futuro

- Los procedimientos de desenvolvimiento de fase y filtrado de imágenes realizados por separado por el procesador ARM y el FPGA, la implementación y evaluación de los algoritmos OpenCL presentados anteriormente comparando sus tiempos de procesamiento, evidentemente reveló las capacidades de su implementación cuando se usan con un FPGA en lugar del procesador ARM, los casos particulares de algoritmos de filtrado de Fourier y Bilaterales también programados en OpenCL también muestran su eficiencia para reducir el ruido de imagen, muy útil al combinarlos con el desenvolvimiento de fase, mostrando tiempos de procesamiento para un tamaño de imagen de 1024x1024 de hasta ≈30X para el proceso 3 y 23x más rápido el FPGA que el procesador ARM en el proceso 6, también se obtuvieron mejoras notables para tamaños de 512x512 (≈17X) y 256x256 (≈10X) de este último, por mencionar que en el proceso 5 y 6 se implementó el proceso de desenvolvimiento, compensación de la aberración y el filtrado FB en el procesador ARM (en C) y sobre el FPGA(en OpenCL). Podemos agregar que la implementación de OpenCL puede considerarse como una herramienta muy importante para PSDHM en el FPGA y en otros métodos similares; una investigación adicional podría estar orientada a la adaptación y optimización de los algoritmos OpenCL combinados con el filtrado FFT, considerando la gran diversidad de filtros de Fourier para varias aplicaciones.
- De acuerdo con las perspectivas comentadas anteriormente cabe mencionar que el FPGA se está convirtiendo cada día en una buena opción de implementación de hardware en muchos sentidos, ya sea programada directamente con lenguajes de programación de bajo nivel como VHDL o de alto nivel como OpenCL.

Existen muchas aplicaciones donde se puede utilizar la PSDHM, en este trabajo se utilizó para reconstrucción 3D, mientras que para un trabajo futuro se pretende implementar en la determinación cuantitativa y cualitativa de cualquier muestra que se tenga bajo estudio. Debido a que el objetivo principal de cualquier área de trabajo ya sea en la industria o en la investigación es necesario obtener información lo más rápido posible, la actual tendencia de estos dispositivos de hardware es que continuamente se están convirtiendo en herramientas cada vez más usada gracias a los avances tecnológicos que siguen teniendo así como su desempeño, de ahí que Intel adquiriera de la empresa Altera en 2015; en este trabajo se comprobó que el FPGA es una herramienta eficiente para la implementación de algoritmos en PSDHM y de acuerdo con la literatura en muchos otros campos, además de que continuamente los fabricantes se agregan nuevas mejoras en las arquitecturas de los FPGA como grandes bloques RAM, transceptores de alta velocidad, DSPs, microprocesadores multinúcleo entre otras, que permiten escalar y mejorar cualquier aplicación a implementar, también posibilita reducir costos por su capacidad de ser un dispositivo reconfigurable además de las características propias que posee; una de las propuestas de avance para el futuro de esta tesis también se centra en la implementación de monitoreo de muestras a través de la implementación de un laboratorio remoto [83-91], que permita obtener información como la topografía, altura, rugosidad u otras características importantes de una muestra bajo estudio; esto es, que la implementación de la PSMHD permita ser controlada vía remota para la supervisión y monitoreo, ya sea desde muestras biológicas, efectos de corrosión o en muestras que estén siendo sometidas a tención, intenso calor o abrasión, por lo que resulta de interés la trascendencia que pudiera tener con la adquisición, procesamiento y muestra de resultados a través de un dispositivo FPGA.

Bibiliografía

[1] 3D imaging and statistics of red blood cells in multiple deformation states, Mona Mihailescu, Alexandru Gheorghiu, Roxana-Cristina Popescu, the publishing house of the romanian academy proceedings of the romanian academy, Series A, Volume 14, Number 3/2013, pp. 211–218.

[2] 3D nuclear track analysis by digital holographic microscopy, F. Palacios , D. Palacios Fernández , J. Ricardo , G.F. Palacios , L. Sajo-Bohus ,E. Goncalves , J.L. Valin , F.A. Monroy, 1350-4487 ,Elsevier, 2010.

[3]Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology, Stephen M. (Steve) Trimberger, Fellow IEEE, 0018-9219, 2015 IEEE.

[4]Fast and robust camera-based motion tracking using FPGAs in microrobotics, Dissertation zur Erlangung des Grades eines Doktors der Ingeneurswissenschaften (Dr.-Ing.), Dipl.-Inform. Claas Diederichs. 28. November 2014.

[5] Using low cost FPGAs for realtime video processing, Master's Thesis Filip Roth, Brno, spring 2011.

[6] Hitting the accelerator: the next generation of machine-learning chips, Deloitte Touche Tohmatsu Limited, London. J14285 (2017).

[7] Digital Image Processing, Rafael C. Gonzalez, Richard E. Woods, Prentice Hall, Second Edition, p. 147-159 (2001).

[8]Gordon E. Moore. "Cramming more components onto integrated circuits" Electronics, Volume 38, Number 8, Apri1965.

[9]http://www.expansion.com/economiadigital/innovacion/2017/07/05/5953a11c268e3e1d718b4873.html

[10] https://www.altera.com/about.html

[11] https://www.nextplatform.com/2016/03/14/intel-marrying-fpga-beefy-broadwell-open-compute-future/

[12] Heterogeneous Computing with OpenCL, Benedict Gaster, Lee Howes, David R. Kaeli, Perhaad Mistry.

[13] From opencl to high-performance hardware on fpgas, Tomasz S. Czajkowski, Utku Aydonat, Dmitry Denisenko, John Freeman, Michael Kinsner, David Neto, Jason Wong, Peter Yiannacouras and Deshanand P. Singh, 978-1-4673-2256-0/12/ IEEE.(2012).

[14] Invited paper: using opencl to evaluate the efficiency of cpus, gpus and fpgas for information filtering, Doris Chen, Deshanand Singh, Altera Toronto Technology Center, Toronto, Ontario, Canada, 978-1-4673-2256-0/12/ IEEE(2012).

[15] Intel® FPGA product catalog versión 17.1. SG-PRDCT-17.1 (2017).

[16] Intel® User-customizable SoC FPGAs, Altera Intel Corporation. (2017).

[17] Gabor D (1948) A new microscopic principle. Nature 161:777-778.

[18] Digital Holography. Ulf Schnars, Werner Jüptner: digital hologram recording, numerical reconstruction and related techniques, p. 37-39, Springer-Verlag, Berlin-Heidelberg, 2005.

[19] Image reconstruction in digital holographic microscopy on GPU, Bachelor Thesis, Andrej Krejˇcír, 2013.

[20] Interferometría Holográfica Digital en Tiempo Real: Aplicación en la Cuantificación de Deformaciones Mecánicas, Natalia Múnera Ortiz, Universidad Nacional de Colombia Facultad de Ciencias, Escuela de Física, Medellín, Colombia 2013.

[21] Leith E. N., Upatnieks J (1962) Reconstructed wavefronts and communication theory. Jour Opt Soc Amer 52:1123-1130.

[22] E. N. Leith, and J. Upatnieks, “Wavefront reconstruction with diffused illumination and threedimensional objects,” J. Opt. Soc. Am. 54, 1295–1301 (1964).

[23] J.W. Goodman and R.W. Lawrence. “Digital image formation from electronically detected holograms”, Applied Physics Letters 11, 77-79 (1967).

[24] W.S. Haddad, D. Cullen, J.C. Solem, J.W. Longworth, A. McPherson, K. Boyer, and C.K. Rhodes, “Fourier-transform holographic microscope”, Applied Optics 31, 4973-4978 (1992).

[25] B.R. Brown, A.W. Lohmann. “Computer-generated binary holograms”. IBM J. Resp. Develop, 160-168 (1969).

[26] A study and simulation of computer generated holograms, Divya P.S., Sheeja M.K,M. Tech Student, SCT College of Engineering, Pappanamcode, Thiruvananthapuram18, Kerala, India, International Journal of Advances in Engineering & Technology, July 2013.

[27] E. Cuche, F. Bevilacqua, and C. Depeursinge, “Digital holography for quantitative phase-contrast imaging,” Optics Letters 24, 291–293 (1999).

[28] C. J. Mann, L. F. Yu, and M. K. Kim, “Movies of cellular and sub-cellular motion by digital holographic microscopy,” Biomed. Eng. Online 5, 21 (2006).

[29] J. Kuhn, F. Charriere, T. Colomb, E. Cuche, F. Montfort, Y. Emery, P. Marquet, and C. Depeursinge, “Axial sub-nanometer accuracy in digital holographic microscopy,” Measurement Science & Technology 19, (2008).

[30] Aplicación de la microscopía holográfica digital en transmisión para la caracterización del espesor de recubrimientos delgados, Scientia et Technica Año XIII, No 36, Septiembre 2007. Universidad Tecnológica de Pereira. ISSN 0122-1701.

[31] Cell Identification with, 3-D Holographic, Inkyu Moon, Mehdi Daneshpanah, Arun Anand and Bahram Javidi, OPN Optics & Photonics News, 1047-6938/11/06/18/6, OSA.(2011).

[32] Digital Holographic Microscopy A New Method for Surface Analysis and Marker-Free Dynamic Life Cell Imaging, Dr. Björn Kemper, Patrik Langehanenberg, Prof. h.c. (Acad. Sci. UA) Gert von Bally, Laboratory of Biophysics University of Münster Robert-Koch-Straße 45, ptik & Photonik June 2007.

[33] Fast phase reconstruction in white light diffraction phase microscopy, Hoa V. Pham,Christopher Edwards, Lynford L. Goddard, and Gabriel Popescu, Optical Society of America, January 2013 / Vol. 52, No. 1 / Applied optics (2013).

[34] Topographic measurements using digital holographic microscopy combined with photorefractive single-shot holography, Doris Grosse aus Essen, Ruhr Universitat Bochum, Termin der mündlichen Prüfung: 31.01.2014.

[35] Visualization of fast-moving cells in vivo using digital holographic video microscopy, Hongyue Sun, Bing Song, Hongpai Dong, Brian Reid, Michael A. Player John Watson, Journal of Biomedical Optics 13(1), 014007 (January/February 2008).

[36] Adaptive filter to improve the performance of phase-unwrapping in digital holography, Francisco Palacios, Edison Goncalves, Jorge Ricardo, Jose L. Valin. Elsevier, Optics Communications 238 (2004) 245–251.

[37] I. Yamaguchi, and T. Zhang, “Phase-shifting digital holography,” Optics Letters 22,1268–1270 (1997).

[38] G. Indebetouw, “Properties of a scanning holographic microscope: improved resolution, extended depth-of-focus, and/or optical sectioning,” J. Mod. Opt. 49, 1479–1500 (2002).

[39] G. Indebetouw, Y. Tada, J. Rosen, and G. Brooker, “Scanning holographic microscopy with resolution exceeding the Rayleigh limit of the objective by superposition of off-axis holograms,” Applied Optics 46, 993–1000 (2007).

[40] O. Matoba, T. Nomura, E. Perez-Cabre, M. S. Millan, and B. Javidi, “Optical Techniques for Information Security,” Proc. IEEE 97, 1128–1148 (2009).

[41] Non-scanning motionless fluorescence three-dimensional holographic microscopy, Joseph rosen and gary brooker, nature photonics | VOL 2 |March 2008 | www.nature.com/naturephotonics.

[42] Microscopia holografica digital mediante desplazamiento de fase con fuentes parcialmente coherentes, tesis profesional para obtener el grado de Maestría en ingeniería y ciencias aplicadas, Ing. Hector Hugo Gutierrez Payan, Ciicap (2015).

[43] Direct phase determination in hologram interferometry with use of digitally recorded holograms Schnars U (1994). Journ Opt Soc Am A 11(7):2011-2015, reprinted (1997) In: Hinsch K, Sirohi R (eds). SPIE Milestone Series MS 144, pp 661 – 665.

[44]. I. Yamaguchi, and T. Zhang, “Phase-shifting digital holography,” Optics Letters 22, 1268–1270 (1997).

[45] K. Creath, “Phase measurements interferometry techniques,” in Progress in Optics, E. Wolf, ed. (Elsevier Science, New York, 1988), pp. 349–393.

[46] T. Kreis, Handbook of holographic interferometry: Optical and digital methods (Wiley-VCH, 2005).

[47] Kebbel V, Hartmann HJ, Jüptner W (2001) Application of digital holographic microscopy for inspection of micro-optical components. In: Proc SPIE vol 4398, pp 189-98.

[48] Optical sectioning with a low-coherence phase-shifting digital holographic microscope, Yu-Chih Lin, Chau-Jern Cheng, and Ting-Chung Poon, Optical Society of America, 0003-6935/11/070B25-06.(2011).

[49] Principles of Digital Image Processing, Fundamental Techniques, Wilhelm Burger, Mark J. Burge, ISBN 978-1-84800-190-9, Springer-Verlag London Limited 2009.

[50] Digital image processing, Piks Scientific Inside, Fourth Edition, William K. Pratt PixelSoft, Inc. Los Altos, California, ISBN: 978-0-471-76777-0, pp.269, Wiley-interscience. (2007).

[51] Image Processing Toolbox, for user with Matlab®, User's Guide, version 5, Section 7.4,The MathWorks, Inc. (2004).

[52] Principles of Digital Image Processing, Fundamental Techniques, Wilhelm Burger • Mark J. Burge, pp. 97, ISBN 978-1-84800-190-9, Springer. (2009).

[53] Sylvain Paris, Pierre Kornprobst, Jack Tumblin and Frédo Durand. Bilateral Filtering: Theory and Applications. Foundations and Trends in Computer Graphics and Vision, Vol. 4, No. 1, Now Publishers, Hanover, MA, USA (2009).

[54] Bilateral filtering for gray and color images, Tomasi, C., & Manduchi, R., IEEE International Conference on Computer Vision, (págs. 839–846).(1998)

[55] Multiresolution Bilateral Filtering for Image, Zhang, M., & Gunturk, B. Denoising. IEEE Transactions on Image Processing, 17(12), 2324 - 2333. (2008).

[56] A Fast Approximation of the Bilateral Filter Using a Signal Processing Approach, Paris, S., & Durand, F. Int J Comput Vis, 81(1), 24-52.(2009).

[57] Digital Image Processing, Rafael C. Gonzalez, Richard E. Woods, Prentice Hall, Second Edition, p147-159 (2001).

[58] Image Processing Handbook, Sixth Edition, John C. Russ, North Carolina State University, ISBN: 978-1-4398-4063-4, pp. 337, Taylor & Francis Group.(2011).

[59]Altera measurable advantage, Architecture Brief, http://www.altera.com/ - literature/br/br-soc-fpga.pdf.

[60] Smith-Waterman Protein Search with OpenCL on an FPGA, E. Rucci and A. De Giusti and M. Naiouf, C. García and G. Botella and M. Prieto-Matias, 2015 IEEE Trustcom/BigDataSE/ISPA.

[61] An efficient KNN algorithm implemented on FPGA based heterogeneous computing system using OpenCL, Yuliang Pu, Jun Peng, Letian Huang, John Chen, and 2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines.

[62] Software-Based High-Level Synthesis Design of FPGA Beamformers for Synthetic Aperture Imaging, João Amaro, , Billy Y. S. Yiu, Gabriel Falcão, Marco A. C. Gomes and Alfred C. H. Yu, IEEE Transactions on Ultrasonics, Ferroelectrics, and Fr 862 equency Control, vol. 62, no. 5, May 2015.

[63] Energy-Efficient FPGA Implementation for Binomial Option Pricing Using OpenCL, Valentin Mena Morales, Pierre-Henri Horrein, Amer Baghdadi, Erik Hochapfel, Sandrine Vaton, IEEE978-3-9815370-2-4/DATE14/ 2014 EDAA.

[64] Real-time 3D Reconstruction for FPGAs: A Case Study for Evaluating the Performance, Area, and Programmability Trade-offs of the Altera OpenCL SDK, Quentin Gautier, Alexandria Shearer, Janarbek Matai, Dustin Richmond, Pingfan Meng and Ryan Kastner, University of California, San Diego, 978-1-4799-6245-7/14/,2014 IEEE.

[65] On the Portability of the OpenCL Dwarfs on Fixed and Reconfigurable Parallel Platforms, Konstantinos Krommydas, Muhsen Owaida, Christos D. Antonopoulos, Nikolaos Bellas and Wu-chun Feng, 19th IEE International Conference on Parallel and Distributed Systems 1521-9097/13, IEEE(2013).

[66] Modulador/demodulador OFDM reconfigurable, implementado en FPGA, Marcos F. Guerra Medina, Oswaldo González, Francisco Delgado, José Rabadán, Dpto. Física Fundamental y Experimental, Electrónica y Sistemas, Universidad de La Laguna (ULL), e Instituto para el Desarrollo Tecnológico y la Innovación en

Comunicaciones (IDeTIC), Universidad de Las Palmas de Gran Canaria (ULPGC).España(2013).

[67] Design and implementation in VHDL code of the two-dimensional fast Fourier transform for frequency filtering, convolution and correlation operations, Juan M Vilardy1, F. Giacometto, C. O. Torres and L. Mattos, XVII Reunion Iberoamericana de Óptica & X Encuentro de Óptica, Láseres y Aplicaciones IOP Publishing, Journal of Physics: Conference Series 274 (2011) 012048.

[68] Transformada de fourier para el estándar dtmb de televisión digital, fourier transform for dtmb standard, Nelson García Rodríguez, Lacetel, Cuba, XV Convención y Feria Internacional, ISBN: 978-959-7213-02-04.

[69] Meredith Instruments, http://mi-lasers.com/hene-lasers.

[70]https://www.edmundoptics.com/c/microscopy/625/.

[71] http://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=754.

[72] Point Grey, Innovation in Imaging Grasshopper3 U3 USB 3.0 Camera Technical Reference Version 13.0. (2012-2015).

[73]DE1-SoC User Manual, Altera University Program, Terasic. (2014).

[74]Stereoscopic Depth on an FPGA via OpenCL, Ahmed Kamel, Aashish Agarwal, Master Degree Thesis, School of Electrical and Computer Engineering, Cornell University, Design Project Report (2015).

[75]K. Itoh, “Analysis of the phase unwrapping problem,” Applied Optics, Vol. 21, No. 14, p. 2470, July15, 1982.

[76] Bernd Jähne, Digital Image Processing, 5th revised and extended edition. Springer 2002. [Pág 21-22].

[77]DE1-SoC OpenCL, Altera University Program, Terasic. (2014).

[78] Altera SDK for OpenCL Programming Guide. Last updated for Quartus Prime Design Suite: 15.1, 101 Innovation Drive San Jose, CA 95134 (2015).

[79] Stereoscopic Depth on an FPGA via OpenCL, Ahmed Kamel, Aashish Agarwal, Master Degree Thesis, School of Electrical and Computer Engineering, Cornell University, Design Project Report (2015).

[80] https://wgww.fixstars.com/en/opencl/book/OpenCLProgrammingBook/mersenne-twister.

[81] T. Colomb, E. Cuche, F. Charrière, J. Kühn, N. Aspert, F. Montfort, P. Marquet, and C. Depeursinge, "Automatic procedure for aberration compensation in digital holographic microscopy and applications to specimen shape compensation", Appl. Opt. 45,851-863 (2006).

[82] Advanced Holography–Metrology and Imaging, Edited by Izabela Naydenova, Published by InTech, p. 183-184 (2011).ISBN 978-953-307-729-1.

[83]Diseño de Laboratorios virtuales y/o remotos. J.M Andujar Márquez, T. J. Mateo Sanguino, Dpto. De Ingeniería electrónica, sistemas informáticos y automática (DIESIA), Elsevier.es. , www.revista–riai.org

[84] Integración de dispositivos físicos en un laboratorio remoto de control mediante diferentes plataformas: Labview, Matlab y C/C++, R. Costa-Castelló M. Vallés L. M. Jimenéz L. Diaz-Guerra, A. Valera, R. Puerto, Comité Español de Automática, ISSN: 1697-7912, ScienDirect. (2010).

[85] Laboratorios remotos, análisis, características y su desarrollo como alternativa a la práctica en la facultad de ingeniería, Ronald Zamora Musa, Revista Inge-CUC / Vol. 6 - No. 6 / Octubre 2010 / Barranquilla - Colombia / ISSN 0122-6517.

[86] Un Laboratorio de Ingeniería de Control Basado en Internet Nourdine Aliane, Javier Fernández, Aída Martínez y Jaime Ortiz , Universidad Europea de Madrid,

Departamento de Arquitectura de Computadores y Automática, Información Tecnológica – Vol. 18 Nº 6 – 2007 https://scielo.conicyt.cl/pdf/infotec/v18n6/art04.pdf.

[87] Experimentación Remota para el Análisis de Dispositivos Ópticos y de Potencia, Duván F. Garcia, Alexander Vera, Álvaro Bernal, Eleventh LACCEI Latin American and Caribbean Conference for Engineering and Technology (LACCEI'2013) "Innovation in Engineering, Technology and Education for Competitiveness and Prosperity" August 14 - 16, 2013 Cancun, Mexico.

[88] Entornos virtuales de aprendizaje: nuevos retos educativos, Virtual Learning Environments: New educational challenges, Adrián Segura-Robles, Miguel Ángel Gallardo-Vigil, http://www.ugr.es/~sevimeco/revistaeticanet/numero132/Articulos/Formato/177.pdf

[89] Laboratorios Remotos de Automatización y Control, Dr. Manuel E. Macías Depto. de Ingeniería Eléctrica ITESM, Campus Monterrey. http://www.cudi.edu.mx/primavera_2008/presentaciones/laboratorios_manuel_macias.pdf.

[90] Desarrollo básico de un Laboratorio Virtualde Control de Procesos basado en Internet, Casallas Ricardo, Chacón Rafael, Posso Fausto, Departamento de Ingeniería Electrónica – UNET, pp. 58-65 Universidad del Táchira / Universidad de Los Andes Táchira – Venezuela. (2005).

[91] Laboratorios virtuales y remotos para la práctica a distancia, de la automática, S. Dormido, J. Sánchez, F. Morilla, Dpto. de Informática y Automática, UNED, Avda. Senda del Rey s/n, 28040 Madrid. (2007).

Bibliografia anexos

[1] https://www.top500.org.
[2] http://www.abacus.cinvestav.mx.
[3] https://www.intel.la/content/www/xl/es/products/processors/core/x-series/i9-

7960x.html

[4] Intel® User-customizable SoC FPGAs, Altera Intel Corporation. (2017).

[5] An OpenCL implementation for the solution of the time-dependent Schrödinger equation on GPUs and CPUs, Cathal Ó Broin, L.A.A. Nikolopoulos, School of Physical Sciences, Dublin City University, Collins Ave and National Center for Plasma Science & Technology,Elsevier,(2012).

[6] A portable OpenCL Lattice Boltzmann code for multi- and many-core processor architectures, Enrico Calore, Sebastiano Fabio Schifano, and Raffaele Tripiccione, Italia, Elsevier (2014).

[7] FPGA-GPU communicating through PCIe, Yann Thoma, Alberto Dassatti, Daniel Molla, Enrico Petraglio, Reconfigurable and Embedded Digital Systems Institute, REDS HEIG-VD, School of Business and Engineering Vaud,Elsevier(2015).

[8] Comparison of High Level FPGA Hardware Design for Solving Tri-Diagonal Linear Systems, David J. Warne, Neil A. Kelson, and Ross F. Hayward, Queensland University of Technology, Elsevier.(2014).

[9] Fractal Video Compression in OpenCL: An Evaluation of CPUs, GPUs, and FPGAs as Acceleration Platforms, Doris Chen, Deshanand Singh, Altera Toronto Technology Center , Ontario, Canada, 978-1-4673-3030-5/13, IEEE.(2013).

[10] Real-time 3D Reconstruction for FPGAs: A Case Study for Evaluating the Performance, Area, and Programmability Trade-offs of the Altera OpenCL SDK, Quentin Gautier, Alexandria Shearer, Janarbek Matai, Dustin Richmond, Pingfan Meng and Ryan Kastner,University of California, San Diego, 978-1-4799-6245-7/14,IEEE.(2014).

[11] Improving Data Partitioning Performance on OpenCL-based FPGAs, Zeke Wang, Bingsheng He, Nanyang Technological University, Singapore and Wei Zhang-Hong Kong University of Science and Technology, 2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines. 978-1-4799-9969-9/15. IEEE (2015).

ANEXO 1. Óptica Física.

Ondas de luz.

La luz se puede describir como una onda electromagnética o como una corriente de partículas llamados fotones. El modelo a ser referido depende del experimento bajo investigación. Ambos modelos se contradicen entre sí, pero son necesarios para describir el espectro completo de fenómenos de luz. Interacción de la luz con la estructura atómica de la materia es descrita por la óptica cuántica, la teoría que trata con los fotones. Refracción, la difracción y la interferencia están perfectamente descritas por el modelo de onda, que se basa en la teoría del electromagnetismo clásico. La interferencia y la difracción forman la base de la holografía. La teoría apropiada es por lo tanto el modelo de onda. Las cantidades oscilantes son el eléctrico y los campos magnéticos. Las amplitudes de campo oscilan perpendicularmente a la propagación dirección de la luz y perpendicularmente entre sí, es decir, las ondas de luz son transversales fenómenos. Las ondas de luz se pueden describir ya sea por el campo eléctrico o magnético. La propagación de la luz se describe mediante la ecuación de onda, que se desprende de Ecuaciones de Maxwell. La ecuación de onda en el vacío es:

$$\nabla^2 \vec{E} - \frac{1}{c^2}\frac{\partial^2 \vec{E}}{\partial t^2} = 0 \qquad \text{Ec. 1.1}$$

Aquí $\vec{E}$ es el campo eléctrico y ∇^2 es el operador de Laplace definido como:

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \qquad \text{Ec. 1.2}$$

C es la velocidad de la luz en el vacío:

$$c = 299792458 m/s \qquad \text{Ec. 1.3}$$

El campo eléctrico $\vec{E}$ es una cantidad vectorial, lo que significa que podría vibrar en cualquier dirección, que es perpendicular a la propagación de la luz. Sin embargo, en muchas aplicaciones la onda vibra solo en un solo plano.

Tal luz se llama linealmente polarizada. En este caso, es suficiente considerar la ecuación de onda escalar

$$\frac{\partial^2 E}{\partial z^2} - \frac{1}{c^2}\frac{\partial^2 E}{\partial t^2} = 0 \qquad \text{Ec. 1.4}$$

Donde la luz se propaga en la dirección z. Se puede verificar fácilmente que se describió una onda plana armónica polarizada linealmente por

$$E(x, y, z, t) = a\cos(\omega t - \vec{k}\vec{r} - \varphi_0) \qquad \text{Ec. 1.5}$$

Es una solución de la ecuación de la onda.

$E(x, y, z, t)$ es el modulo del vector campo eléctrico en un punto con vector espacial $\vec{r} = (x, y, z)$en el tiempo t. La cantidad a se llama amplitud. La onda vector $\vec{k}$ describe la dirección de propagación de la onda:

$$\vec{k} = k\vec{n} \qquad \text{Ec. 1.6}$$

$\vec{n}$ es un vector unitario en dirección de propagación. Los puntos de la misma fase se encuentran en planos paralelos, que son perpendiculares a la dirección de propagación. El módulo de $\vec{k}$ es nombrado el número de onda calculado por:

$$\vec{k} \equiv k = \frac{2\pi}{\lambda} \qquad \text{Ec. 1.7}$$

La frecuencia angular ω corresponde a la frecuencia de f de la onda de luz por

$$\omega = 2\pi f \qquad \text{Ec. 1.8}$$

La frecuencia f y la longitud de onda λ están relacionadas con la velocidad de la luz c :

$$c = f\lambda \qquad \text{Ec. 1.9}$$

El término que varía espacialmente:

$$\varphi = -\vec{k}\vec{r} - \varphi_0 \qquad \text{Ec. 1.10}$$

Es nombrado fase, con fase constante φ_0. Debe señalarse que esta definición no está estandarizada. Algunos autores designan el argumento completo de la función coseno, $\omega t - \vec{k}\vec{r} - \varphi_0$ como la fase. La definición Eq. (1.10) es favorable para describe el proceso holográfico y, por lo tanto, se usa en este libro.

La longitud de la onda de la luz visible en el vacío está en el rango de $400nm$(violeta) a $780nm$(color rojo oscuro). El correspondiente rango de frecuencia es $7.5x10^{14}Hz$ a $3.8x10^{14}Hz$. Los sensores de luz como por ejemplo el ojo humano, fotodiodos, películas fotográficas o CCD no son capaces de detectar tales altas frecuencias debido a la técnica y razones físicas. La única cantidad directamente mensurable es la intensidad. Es proporcional al promedio de tiempo del cuadrado del campo eléctrico:

$$I = \varepsilon_0 c\langle E^2\rangle_t = \varepsilon_0 c \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} E^2\, dt \qquad \text{Ec. 1.11}$$

$\langle\ \ \rangle_t$ Representa el promedio de tiempo durante muchos períodos de luz. El factor constante $\varepsilon_0 c$ resulta, si la intensidad se deriva formalmente de ecuaciones de Maxwell. La constante ε_0 es la permisividad del vacío.

Para una onda plana Eq. (1.5) tiene que ser insertada:

$$I = \varepsilon_0 c a^2 \langle cos^2\,(\omega t - \vec{k}\vec{r} - \varphi_0)\rangle_t = \frac{1}{2}\ \varepsilon_0 c a^2 \qquad \text{Ec. 1.12}$$

De acuerdo con Eq. (1.12) la intensidad se calcula por el cuadrado de la amplitud. La expresión (1.5) se puede escribir en forma compleja como

$$E(x,y,z,t) = a\ Re\{\exp(i\ (\omega t - \vec{k}\vec{r} - \ \varphi_0))\} \qquad \text{Ec. 1.13}$$

Donde 'Re' denota la parte real de la función compleja. Para cálculos esto 'Re' puede ser omitido. Sin embargo, solo la parte real representa la onda física:

$$E(x,y,z,t) = a\ \exp(i\ (\omega t - \vec{k}\vec{r} - \ \varphi_0)) \qquad \text{Ec. 1.14}$$

Una ventaja de la representación compleja es que las partes espacio y el tiempo se factorizan:

$$E(x,y,z,t) = a\ \exp(i\varphi) exp\ (i\omega t) \qquad \text{Ec. 1.15}$$

En muchos cálculos de óptica, solo la distribución espacial de la onda es de interés. En este caso, solo la parte espacial del campo eléctrico, llamada amplitud compleja, tiene que ser considerada:

$$A(x,y,z) = a\ \exp(i\varphi) \qquad \text{Ec. 1.16}$$

Las ecuaciones (1.15) y (1.16) no son solo válidas para ondas planas, sino en general para ondas tridimensionales cuya amplitud a y fase φ pueden ser funciones de x, y y z. En notación compleja, la intensidad ahora se calcula simplemente tomando el cuadrado del módulo de la amplitud compleja

$$I = \frac{1}{2}\varepsilon_0 c|A|^2 = \frac{1}{2}\varepsilon_0 c A^* A = \frac{1}{2}\varepsilon_0 c a^2 \qquad \text{Ec. 1.17}$$

Donde * denota el complejo conjugado. En muchos cálculos prácticos donde el valor absoluto de I no es de interés, se puede despreciar el factor $\frac{1}{2}\ \varepsilon_{0\ c}$, cual significa que la intensidad simplemente se calcula por $I = \ |A|^2$

3.2 Interferencia

La superposición de dos o más ondas en el espacio se denomina interferencia. Si cada uno onda única descrita por $\vec{E}_i = (\vec{r}, t)$ es una solución de la ecuación de onda, también la superposición de la Ec. 1.18 es una solución.

$$\vec{E}(\vec{r}, t) = \sum_i \vec{E}_i(\vec{r}, t) \qquad i = 1,2 \ldots \qquad \text{Ec. 1.18}$$

Esto se debe a que la ecuación de onda es un diferencial lineal ecuación. Si consideramos la interferencia de dos ondas monocromáticas con frecuencias y longitudes de onda iguales. Las ondas deben tener las mismas direcciones polarización, por ejemplo el formalismo escalar se puede usar. Las amplitudes complejas de las ondas son:

$$A_1(x, y, z) = a_1 \exp(i\varphi_1) \qquad \text{Ec. 1.19}$$

$$A_2(x, y, z) = a_2 \exp(i\varphi_2) \qquad \text{Ec. 1.20}$$

La amplitud compleja resultante se calcula luego por la suma las amplitudes individuales:

$$A = A_1 + A_2 \qquad \text{Ec. 1.21}$$

De acuerdo con la ecuación (1.17) la intensidad se convierte:

$$I = |A_1 + A_2|^2 = (A_1 + A_2) + (A_1 + A_2)^* \qquad \text{Ec. 1.22}$$

$$= a_1^2 + a_2^2 + 2a_1 a_2 \cos(\varphi_1 - \varphi_2)$$

$= I_1 + I_2 + 2\sqrt{I_1 I_2} \cos \Delta\varphi$ $\quad I_1$, I_2 son las intensidades individuales y

$$\Delta\varphi = \varphi_1 - \varphi_2 \qquad \text{Ec. 1.23}$$

La intensidad resultante es la suma de las intensidades individuales más el término de interferencia $2\sqrt{I_1 I_2} \cos \Delta\varphi$ que depende de la diferencia de fase entre las ondas. La intensidad alcanza su máximo en todos los puntos a los que se aplica:

$$\Delta\varphi = 2n\pi \ \ para \ \ n = 0,1,2 \ldots \qquad \text{Ec. 1.24}$$

Llamada interferencia constructiva. La intensidad alcanza su mínimo dónde:

$$\Delta\varphi = (2n+1)\pi \ \ para \ \ n = 0{,}1{,}2 \ldots \qquad \text{Ec. 1.25}$$

Esto se llama interferencia destructiva. El entero n es el orden de interferencia. Un patrón de interferencia consiste en franjas oscuras y brillantes como resultado de e interferencia destructiva. La teoría escalar aplicada aquí también puede usarse para ondas con diferentes direcciones de polarización, si los componentes del campo eléctrico vector son considerados. La superposición de dos ondas planas que se cruzan bajo un ángulo θ con respecto a otra resultando resultado en un patrón de interferencia con espaciado equidistante, figura 1.1. El espaciado de franjas d es la distancia de un máximo de interferencia al siguiente y se puede calcular por consideraciones geométricas. La figura 3.1 muestra evidentemente eso

$$sen\ \theta_1 = \frac{\Delta l_1}{d} \ ; sen\ \theta_2 = \frac{\Delta l_2}{d} \qquad \text{Ec. 1.26}$$

Las cantidades $\theta_1\ y\ \theta_2$ son los ángulos entre las direcciones de propagación de los frentes de onda y la pantalla vertical. La longitud Δl_2 es la diferencia de camino del frente de onda W2 con respecto al frente de onda W1 en la posición de la interferencia P1 máximo (W2 tiene que viajar un camino más largo a P1 que W1). En la vecindad máxima P2 se intercambian las condiciones: Ahora W1 tiene que viajar un camino más largo; la diferencia de trayectoria de W2 con respecto a W1 es $-\Delta l_1$

La variación entre las diferencias de camino en los máximos vecinos es por lo tanto $\Delta l_1 + \Delta l_2$ .Esta diferencia es igual a una longitud de onda. Por lo que la condición de interferencia es:

$$\Delta l_1 + \Delta l_2 = \lambda \qquad \text{Ec. 1.27}$$

Combinando la ecuación 1.26 con la 1.27 resulta:

$$d = \frac{\lambda}{sen\ \theta_1 + sen\ \theta_2} = \frac{\lambda}{2\ sen\frac{\theta_1+\theta_2}{2}\ cos\frac{\theta_1-\theta_2}{2}} \qquad \text{Ec. 1.28}$$

La aproximación $\cos(\theta_1 - \theta_2)/2 \approx 1$ y $\theta = \theta_1 + \theta_2$ tenemos:

$$d = \frac{\lambda}{2\, sen\frac{\theta}{2}} \qquad \text{Ec. 1.29}$$

En lugar del espaciado de franjas d, el patrón de franjas también puede describirse por frecuencia espacial f, que es el recíproco de d:

$$f = d^{-1} = \frac{2}{\lambda}\, sen\, \frac{\theta}{2} \qquad \text{Ec. 1.30}$$

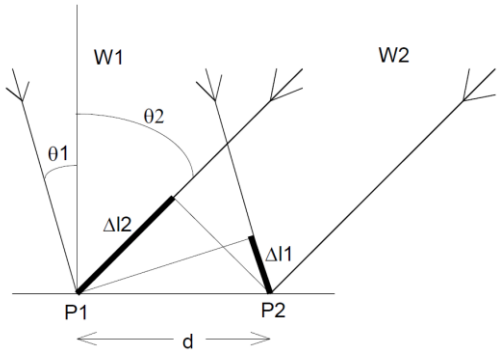


Figura 1.1. Interferencia de dos ondas planas.

3.3 Coherencia

3.3.1 General Coherencia espacial y temporal

En general, la intensidad resultante de dos fuentes diferentes, por ejemplo dos bombillas eléctricas de luz dirigidas en una pantalla, es aditiva. En lugar de franjas oscuras y brillantes como se esperaba por Eq. (1.22) solo un brillo uniforme según la suma de las intensidades individuales se vuelven visibles. Para generar franjas de interferencia, las fases de las ondas individuales deben ser correlacionadas de una manera especial. Esta propiedad de correlación se denomina coherencia. La coherencia es la capacidad de la luz para interferir. Los dos los aspectos de coherencia son la coherencia temporal y espacial. Coherencia temporal describe la correlación de una onda consigo misma en diferentes instantes. La Coherencia espacial representa la correlación mutua de diferentes partes del mismo frente de onda.

3.3.2 Coherencia temporal

El prototipo de un interferómetro de dos haces es el interferómetro de Michelson, figura 1.2. La luz emitida por la fuente de luz S se divide en dos ondas parciales por el divisor de haz BS. Las ondas parciales viajan a los espejos M1 respectivamente M2, y se reflejan nuevamente en las direcciones del incidente. Después de pasar el divisor de haz nuevamente se superponen en una pantalla. Por lo general, las ondas parciales superpuestas no son exactamente paralelas, sino que interfieren con un ángulo pequeño. Como resultado, una doble dimensión patrón de interferencia se hace visible.

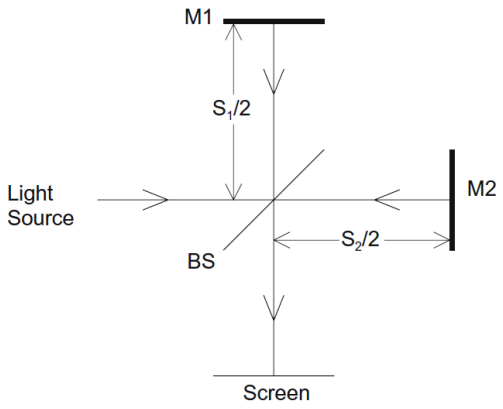


Figura 1.2. Interferómetro de Michelson.

La longitud del camino óptico de BS a M1 y de vuelta a BS es S_1, la longitud de camino óptico de BS a M2 y de vuelta a BS es S_2. Los experimentos prueban que la interferencia solo puede ocurrir si la diferencia de trayectoria óptica $S_1 - S_2$ no excede una cierta longitud L. Si la diferencia de trayectoria óptica excede este límite, las franjas de interferencia desaparecen y solo un brillo uniforme se vuelve visible en la pantalla. La explicación cualitativa de este fenómeno es la siguiente: las franjas de interferencia solo pueden desarrollar si las ondas superpuestas tienen una relación de fase bien definida (constante). La diferencia de fase entre las ondas emitidas por diferentes fuentes varía aleatoriamente y por lo tanto las ondas no interfieren.

Los átomos de la fuente de luz emiten trenes de ondas con longitud finita L. Si la diferencia de trayectoria óptica excede esta longitud del tren de ondas, las ondas parciales que pertenecen juntas no se superponen después de pasar las diferentes formas e interferencia no es posible.

La diferencia de longitud de trayectoria crítica o, de manera equivalente, la longitud de un tren de ondas es llamada longitud de coherencia L. El tiempo de emisión correspondiente para el tren de ondas, La ec. 1.31 llamada tiempo de coherencia

$$\tau = \frac{L}{c} \qquad \text{Ec. 1.31}$$

De acuerdo con las leyes del análisis de Fourier, un tren de ondas con una longitud finita L corresponde a la luz con un ancho espectral finito Δf:

$$L = \frac{c}{\Delta f} \qquad \text{Ec. 1.32}$$

La luz con una longitud de coherencia larga se denomina altamente monocromática. La coherencia la longitud es por lo tanto una medida para el ancho espectral. Longitudes de coherencia típicas de luz irradiada de fuentes térmicas, por ejemplo las ordinarias bombillas eléctricas, están en el rango de algunos micrómetros. Eso significa, interferencia solo se puede observar si los brazos del interferómetro son casi iguales longitudes. Por otro lado, los láseres tienen longitudes de coherencia de unos pocos milímetros (por ejemplo, un láser de diodo multimodo) a varios cientos de metros (por ejemplo, un estabilizador único modo Nd: YAG-laser) hasta varios cientos de kilómetros para el gas especialmente estabilizado láseres utilizados con fines de investigación. La visibilidad

$$V = \frac{I_{max} - I_{min}}{I_{max} + I_{min}} \qquad \text{Ec. 1.33}$$

Es una medida para el contraste de un patrón de interferencia. $I_{max}\ e\ I_{min}$ son dos intensidades cercanas máxima y mínima. Se calculan insertando $\Delta\phi = 0$, respectivamente $\Delta\phi = \pi$ en Eq. (1.22). En el caso ideal de longitud coherencia infinita, la visibilidad es por lo tanto

$$V = \frac{2\sqrt{I_1 I_2}}{I_1 + I_2} \qquad \text{Ec. 1.34}$$

Para considerar el efecto de la longitud de coherencia finita, la auto coherencia compleja $\Gamma(\tau)$ es introduce:

$$\Gamma(\tau) = \langle E(t+\tau)E^*(t)\rangle \qquad \text{Ec. 1.35}$$

$$\lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} E(t+\tau)E^*(t)dt$$

$E(t)$ es el campo eléctrico (para ser precisos: la señal analítica compleja) de uno onda parcial mientras que $E(t+\tau)$ es el campo eléctrico de la otra onda parcial. El último se retrasa en el tiempo por τ. Eq. (1.35) es la autocorrelación de E. La cantidad normalizada define el grado de coherencia.

$$\gamma(\tau) = \frac{\Gamma(\tau)}{\Gamma(0)} \qquad \text{Ec. 1.36}$$

Con una longitud de coherencia finita, la ecuación de interferencia (1.22) debe ser reemplazada por

$$I = I_1 + I_2 + 2\sqrt{I_1 I_2}|\gamma| \cos\Delta\varphi \qquad \text{Ec. 1.37}$$

La intensidad máxima y mínima ahora es calculada:

$$I = I_1 + I_2 + 2\sqrt{I_1 I_2}|\gamma| \qquad \text{Ec. 1.38}$$

$$I = I_1 + I_2 - 2\sqrt{I_1 I_2}|\gamma|$$

Introduciendo esas cantidades a la ecuación 1.33 tenemos:

$$V = \frac{2\sqrt{I_1 I_2}}{I_1 + I_2}|\gamma| \qquad \text{Ec. 1.39}$$

Para dos ondas parciales de la misma intensidad $I_1 = I_2$, la ecuación 3.39 se convierte en

$$V = |\gamma| \qquad \text{Ec. 1.40}$$

$|\gamma|$ es igual a la visibilidad y, por lo tanto, una medida de la capacidad de los dos campos de ondas para interferir $|\gamma| = 1$ describe idealmente luz monocromática o, del mismo modo, luz con una longitud de coherencia infinita. $|\gamma| = 0$ es verdadero para luz completamente incoherente. La luz parcialmente coherente se describe con $0 < |\gamma| < 1$.

3.4 Difracción

Se considera una onda de luz que golpea un obstáculo. Esta podría ser una pantalla opaca con algunos agujeros transparentes, o viceversa, un medio transparente con opaco estructuras. Desde la óptica geométrica se sabe que la sombra se vuelve visible en una pantalla detrás del obstáculo. Mediante un examen más detallado, uno encuentra que esto no es estrictamente correcto. Si las dimensiones del obstáculo (por ejemplo, el diámetro de los agujeros en una pantalla opaca o tamaño de partículas opacas en un volumen transparente) están en el rango de la longitud de onda, la distribución de la luz no está claramente delimitada, sino que forma un patrón de regiones oscuras y brillantes. Este fenómeno se llama difracción. La difracción se puede explicar cualitativamente con el principio de Huygens:

"Cada punto de un frente de onda se puede considerar como un punto de origen para secundaria ondas esféricas. El frente de onda en cualquier otro lugar es la superposición coherente de estas ondas secundarias".

El principio de Huygens se explica gráficamente en la figura 1.3.

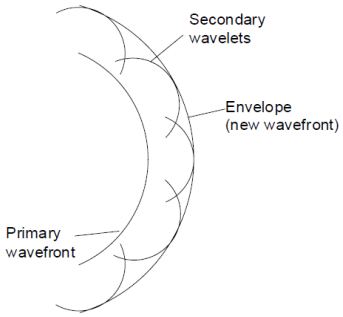


Figura 1.3. Principio de Huygens

La integral de Fresnel-Kirchhoff describe cuantitativamente la difracción.

$$\Gamma(\xi', \eta') = \frac{i}{\lambda} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} A\,(x, y) \frac{exp\left(-i\frac{2\pi}{\lambda}\rho'\right)}{\rho'} \varrho dx dy \qquad \text{Ec. 1.41}$$

Con

$$\rho' = \sqrt{(x - \xi') + (y - \eta') + d^2} \qquad \text{Ec. 1.42}$$

Y

$$\varrho = \frac{1}{2}\,(\cos\theta\ + cos\theta') \qquad \text{Ec. 1.43}$$

$A\,(x, y)$ es la amplitud compleja en el plano de la apertura de flexión, ver sistema coordenadas definido en la figura 1.4. $\Gamma(\xi', \eta')$ es el campo en el plano de observación. ρ' representa la distancia entre un punto en el plano de apertura y un punto en el plano de observación. Eq. (1.41) se puede entender como la formulación matemática del principio de Huygens:

La fuente de luz S situada en el plano de origen con coordenadas (ξ, η) irradia ondas esféricas. $A(x, y)$ es la amplitud compleja de dicha onda en el plano de apertura. Al principio se considera una abertura opaca con un solo orificio en la posición (x, y). Tal agujero es ahora la fuente de ondas secundarias. El campo en el puesto (ξ', η') del plano de difracción es proporcional al campo en el lado de la entrada de la apertura $A(x, y)$ y al campo de la onda esférica secundaria que emerge de (x, y), descrita por $\exp\left(-\frac{i2\pi'}{\lambda\rho'}\right)/\rho'$. Ahora toda la apertura como un plano que consiste en muchas fuentes para las ondas secundarias se considera. Todo el campo resultante en el plano de difracción es, por lo tanto, la integral sobre todas las ondas esféricas secundarias, emergiendo del plano de apertura.

El principio Huygens permitiría que las ondas secundarias no solo se propagaran en dirección hacia adelante, pero también hacia atrás en la dirección de la fuente. Todavía, el experimento demuestra que los frentes de onda siempre se propagan en una dirección. Para excluir formalmente esta situación no física, se define el factor de inclinación ϱ en Eq. (1.43) se introduce en la integral de Fresnel-Kirchhoff.

ϱ depende del ángulo θ entre el rayo incidente de la fuente y el vector unitario $\vec{n}$ perpendicular al plano de apertura, y en el ángulo θ 'entre el rayo doblado y $\vec{n}$, ver figura 1.4. ϱ es aproximadamente cero para $\theta \approx 0$ 0 y $\theta' \approx \pi$. Esto previene las ondas viajando hacia atrás. En la mayoría de las situaciones prácticas, tanto θ como θ' son muy pequeños y $\varrho \approx 1$. El factor de inclinación se puede considerar como una corrección a la integral de difracción, como se hace aquí, o derivarse en la difracción formal teoría.

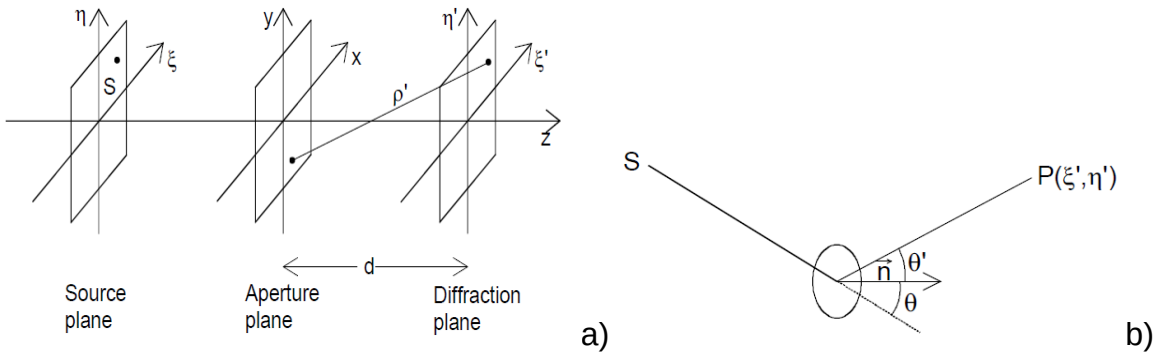


Figura 1.4. a) Sistema de Coordenadas y b) Ángulos.

3.5 Grabación y reconstrucción de hologramas

Los hologramas generalmente se graban con una configuración óptica que consiste en una fuente de luz (láser), espejos y lentes para guiar el haz y un dispositivo de grabación, por ejemplo una placa fotografía. Una configuración típica se muestra en la figura 1.5a. Luz con suficiente la longitud de coherencia se divide en dos ondas parciales por un divisor de haz (BS). La primera onda ilumina el objeto. Está dispersa en la superficie del objeto y se refleja al medio de grabación. La segunda onda, llamada onda de referencia, ilumina el medio sensible a la luz directamente. Ambas ondas interfieren. El patrón de interferencia se registra, por ejemplo desarrollo químico de la placa fotográfica.

El grabado patrón de interferencia se llama holograma. La onda del objeto original se reconstruye iluminando el holograma con la onda de referencia, figura 1.5b. Un observador ve una imagen virtual, que es indistinguible del objeto original. La imagen reconstruida exhibe todos los efectos de perspectiva y profundidad de foco. El proceso holográfico se describe matemáticamente, la amplitud compleja de la onda del objeto se describe por

$$E_O(x,y) = a_O(x,y)\exp(i\varphi_O(x,y)) \qquad \text{Ec. 1.44}$$

Donde la amplitud $a_0(x,y)$ y la fase $\varphi_0(x,y)$

$$E_R(x,y) = a_R(x,y)\exp(i\varphi_R(x,y)) \qquad \text{Ec. 1.45}$$

Es la amplitud compleja de la onda de referencia con real amplitud a_R y fase φ_R.

Ambas ondas interfieren en la superficie del medio de grabación. La intensidad es calculada por

$$I(x,y) = |E_O(x,y) + E_R(x,y)|^2 \qquad \text{Ec. 1.46}$$

$$\big(E_O(x,y) + E_R(x,y)\big)\big(E_O(x,y) + E_R(x,y)\big)^*$$

$$E_R(x,y)E_R^*(x,y) + E_O(x,y)E_O^*(x,y) + E_O(x,y)E_R^*(x,y) + E_R(x,y)E_O^*(x,y)$$

La transmisión de amplitud $h(x,y)$ de la placa fotográfica desarrollada (o de otros medios de grabación) es proporcional a $I\,(x,y)$:

$$h(x,y) = h_0 + \beta\tau I(x,y) \qquad \text{Ec. 1.47}$$

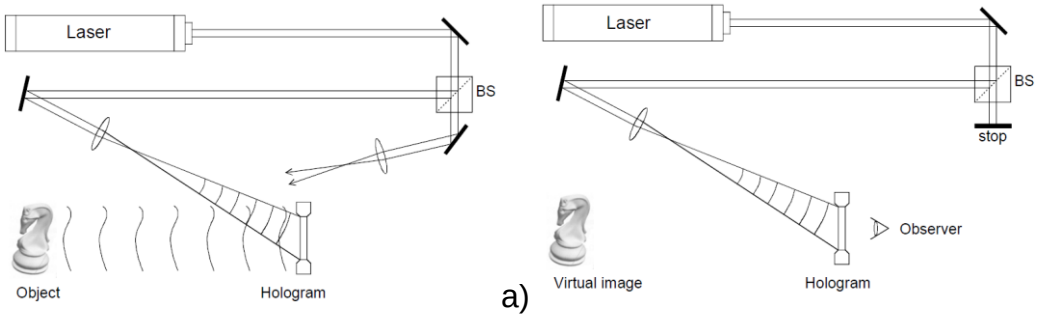


Figura 1.5. a) Grabación de Holograma, b) Reconstrucción del holograma.

La constante β es la pendiente de la transmitancia de amplitud frente a la exposición característica del material sensible a la luz. Para las emulsiones fotográficas β es negativo. La variable τ es el tiempo de exposición y h_0 es la transmisión de amplitud la placa no expuesta. $h(x,y)$ Se llama función de holograma. En Holografía digital usando CCD como el medio de grabación h_0 puede ser descuidado.

Para la reconstrucción de hologramas, la transmisión de amplitud debe multiplicarse con la amplitud compleja de la onda de reconstrucción (referencia):

$$E_R(x,y)h(x,y) = \qquad \text{Ec. 1.48}$$

$$[h_0 + \beta\tau(a_R^2 + a_O^2)]E_R(x,y) + \beta\tau a_R^2 E_O(x,y) + \beta\tau E_R^2(x,y)E_O^*(x,y)$$

El primer término en el lado derecho de esta ecuación es la onda de referencia, multiplicada por un factor. Representa la onda no difractada que pasa el holograma (difracción cero orden). El segundo término es la onda de objeto reconstruida, formando la imagen virtual. El verdadero factor $\beta\tau a_R^2$ solo influye en el brillo de la imagen. El tercer término genera una imagen real distorsionada del objeto. Para holografía fuera del eje la imagen virtual, la imagen real y la onda no difractada están espacialmente separadas. La razón de la distorsión de la imagen real es el factor complejo que varía espacialmente E_R^2, que modula la imagen formando la onda del objeto conjugada E_O^*. Se puede generar una imagen real sin distorsión utilizando el haz de referencia conjugado E_R^* para la reconstrucción:

$$E_R^*(x,y)h(x,y) = \qquad \text{Ec. 1.49}$$

$$[h_0 + \beta\tau(a_R^2 + a_O^2)]E_R^*(x,y) + \beta\tau a_R^2 E_O^*(x,y) + \beta\tau E_R^2(x,y)E_O(x,y)$$

3.6 El Interferómetro Mach-Zehnder

Existen varios interferómetros de dos ondas con innumerables aplicaciones en óptica, metrología, diagnóstico de plasma y otros campos relacionados. La mayoría de estos interferómetros son variantes del histórico interferómetro Michelson.

Como se muestra en la Figura 1.6, el interferómetro Mach-Zehnder usa dos divisores de haz y dos espejos para dividir y recombinar los haces. El espaciado de las franjas está controlado variando el ángulo entre los rayos que emergen del interferómetro. Además, para cualquier ángulo dado entre los haces, la posición del punto de intersección de un par de rayos que se originan desde el mismo punto en la fuente puede ser controlado variando la separación lateral de los haces. Con una extensión fuente, esto permite obtener franjas de interferencia localizadas en cualquier plano. El interferómetro Mach-Zehnder tiene dos características atractivas. Una es que dos caminos están ampliamente separados y atravesados solo una vez; el otro es que la región de localización de las franjas se puede hacer coincidir con el objeto de prueba, para que se pueda usar una fuente extendida de alta intensidad. El interferómetro de Mark-Zehnder es ampliamente utilizado para estudios de flujo de fluidos, calor transferencia, y la distribución de temperatura en plasmas.

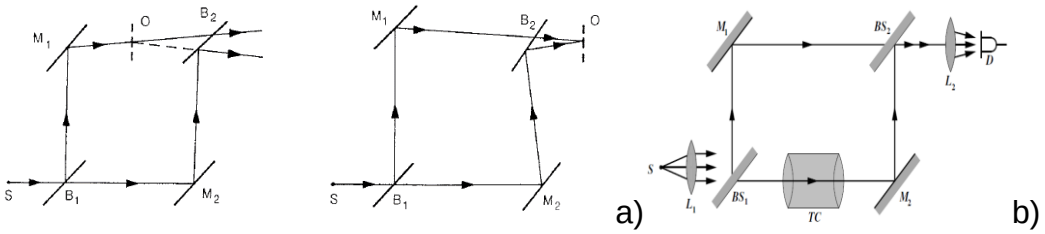


Figura 1.6. a) Localización de franjas interferómetro Mach-Zehnder, b) BS1 y BS2 son divisores de haz, M1 y M2 son espejos, TC es una cámara de prueba, D es detector.

ANEXO 2.

Coherencia espacial

La coherencia espacial describe la correlación mutua de diferentes partes de la misma frente de onda Esta propiedad se mide con el interferómetro Young, figura 2.4. Una fuente de luz extendida emite luz desde diferentes puntos. Posibles interferencias se observan en una pantalla. Una abertura con dos agujeros transparentes se monta entre fuente de luz y pantalla. Bajo ciertas condiciones, que se derivarán en este capítulo, las interferencias son visibles en la pantalla. Las franjas son el resultado de la luz rayos que viajaban de diferentes maneras a la pantalla, ya sea a través de la parte superior o a través del agujero inferior en la abertura. El patrón de interferencia desaparece si la distancia entre los agujeros a excede el límite crítico a_k. Este límite se llama coherencia distancia. El fenómeno no está relacionado con el ancho espectral de la fuente de luz, pero tiene la siguiente causa: Las ondas emitidas por diferentes puntos de origen de la extensión fuente de luz se superponen en la pantalla. Puede suceder que un especial el punto fuente genera un máximo de interferencia en un cierto punto de la pantalla, mientras que otro punto de origen genera un mínimo en el mismo punto de pantalla. Esto es porque la longitud del camino óptico es diferente para los rayos de luz que emergen de diferentes puntos de origen En general, las contribuciones de todos los puntos de origen compensan ellos mismos y el contraste desaparece. Esta compensación se evita si sigue la condición se cumple para cada punto de la fuente de luz:

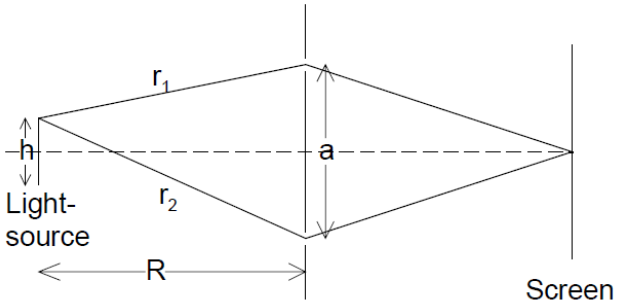


Figura 2.4. Interferómetro de Young.

$$r_1 - r_2 < \frac{\lambda}{2} \qquad \text{Ec. 2.1}$$

Esta condición definitivamente se cumple, si se mantiene para los bordes de la luz fuente. Las siguientes relaciones son válidas para puntos en los bordes:

$$r_1^2 = R^2 + \left(\frac{a-h}{2}\right)^2 ; \quad r_1^2 = R^2 + \left(\frac{a+h}{2}\right)^2 \qquad \text{Ec. 2.2}$$

h es el ancho de la fuente de luz. Uso de los supuestos $a \ll R$ y $h \ll R$ resulta a

$$r_1 - r_2 < \frac{ah}{2R} \qquad \text{Ec. 2.3}$$

Combinando la ecuación 2.1 y 2.3 tenemos

$$\frac{ah}{2R} < \frac{\lambda}{2} \qquad \text{Ec. 2.4}$$

La distancia de coherencia es entonces:

$$\frac{a_k h}{2R} < \frac{\lambda}{2} \qquad \text{Ec. 2.5}$$

En contraste con la coherencia temporal, la coherencia espacial depende no solo de propiedades de la fuente de luz, sino también en la geometría del interferómetro. Una fuente de luz puede generar inicialmente interferencia, lo que significa Eq. (2.4) se cumple. Si la distancia entre los agujeros aumenta o la distancia entre la luz fuente y la apertura disminuyen, Eq. (2.4) se viola y la interferencia la figura desaparece. Para considerar la coherencia espacial, la función de autocorrelación definida en:

$$\Gamma(\overrightarrow{r_1}, \overrightarrow{r_2}, \tau) = \langle E(\overrightarrow{r_1}t + \tau)E^*(\overrightarrow{r_2}, t)\rangle \qquad \text{Ec. 2.6}$$

$$= \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} E(\overrightarrow{r_1}t + \tau)E^*(\overrightarrow{r_2}, t)dt$$

r_1 y r_2 son los vectores espaciales de los agujeros en la apertura del interferómetro Young. Esta función se denomina función de correlación cruzada. La función normalizada es

$$\gamma(\overrightarrow{r_1}, \overrightarrow{r_2}, \tau) = \frac{\Gamma(\overrightarrow{r_1}, \overrightarrow{r_2}, \tau)}{\sqrt{\Gamma(\overrightarrow{r_1}, \overrightarrow{r_1}, 0)\Gamma(\overrightarrow{r_2}, \overrightarrow{r_2}, 0)}} \qquad \text{Ec. 2.7}$$

donde $\Gamma(\overrightarrow{r_1},\overrightarrow{r_1},0)$ es la intensidad en r_1 y $\Gamma(\overrightarrow{r_2},\overrightarrow{r_2},0)$ es la intensidad en r_2. Eq. (2.7) describe el grado de correlación entre el campo de luz en r_1 en el tiempo $t+\tau$ con el campo de luz en r_2 en el tiempo t. La función especial $\gamma(\overrightarrow{r_1},\overrightarrow{r_2},\tau=0)$ es una medida para la correlación entre las amplitudes de campo en r_1 y r_2 al mismo tiempo y se llama grado complejo de coherencia. El módulo de la coherencia normalizada la función $|\gamma(\overrightarrow{r_1},\overrightarrow{r_2},\tau)|$ se mide con el interferómetro Young.

Interferómetros de dos haces

Para hacer mediciones usando interferencia, generalmente necesitamos una disposición óptica en el cual dos haces que viajan a lo largo de caminos separados están hechos para interferir. Uno de estos caminos es la ruta de referencia, mientras que el otro es la prueba o medida, camino. La diferencia de camino óptico entre los frentes de onda interferentes es entonces:

$$\Delta p = p_1 - p_2 \qquad \text{Ec. 2.8}$$

$$= \Sigma(n_1 d_1) - \Sigma(n_2 d_2)$$

donde n es el índice de refracción, y d la longitud, de cada sección en las dos rutas. Para producir un patrón de interferencia estacionario, la diferencia de fase entre las dos ondas interferentes no deberían cambiar con el tiempo. Los dos haces interfiriendo deben, por lo tanto, tener la misma frecuencia. Este requisito puede cumplirse solo si provienen de la misma fuente Dos métodos se usan comúnmente para obtener dos haces de una sola fuente son:

• División de frente de onda

• División de amplitud

División de Frente de Onda

La división de frente de onda usa aperturas para aislar dos haces de porciones separadas del frente de onda primario. En la configuración mostrada en la Figura 2.9, utilizada en el experimento de Young para demostrar la naturaleza ondulatoria de la luz, los dos agujeros pueden considerarse como fuentes secundarias. Las franjas de interferencia se ven en una pantalla colocado en la región de

superposición de los haces difractados de los dos aperturas. La división de frente de onda se usa en el interferómetro Rayleigh.

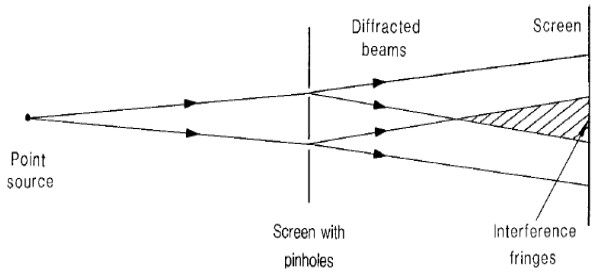


Figura 2.9. Interferencia de dos haces formados por la división de un frente de onda.

División de Amplitud

En la división de amplitud, dos haces se derivan de la misma porción del frente de onda original. Algunos elementos ópticos que se pueden usar para la división de amplitud se muestran en Figura 2.10. El dispositivo más utilizado es una placa transparente recubierta con un reflejo parcial película que transmite un haz y refleja al otro (comúnmente referido como un divisor de haz). Una película parcialmente reflectante también se puede incorporar en un cubo compuesto por dos prismas en ángulo recto con sus caras de hipotenusa cementadas juntas. Otro dispositivo que se ha utilizado es una rejilla de difracción, que produce, además del haz transmitido directamente, uno o más haces difractados. Otro dispositivo que se puede usar es un prisma de polarización, que produce dos rayos polarizados ortogonalmente Un divisor de haz polarizador también puede ser construido incorporando en un cubo divisor de haces una película multicapa que refleja una polarización y transmite la otra. En ambos casos, los vectores eléctricos deben ser traídos de vuelta al mismo plano, generalmente por medio de otro polarizador, para los dos haces a interferir. Algunos tipos comunes de interferómetros de dos haces son:

• El interferómetro Rayleigh

• El interferómetro Michelson (Twyman-Green)

• El interferómetro Mach-Zehnder

• El interferómetro Sagnac

Más adelante hablaremos del interferómetro Mach-Zehnder que utilizaremos en este trabajo para hacer las pruebas con las imágenes obtenidas.

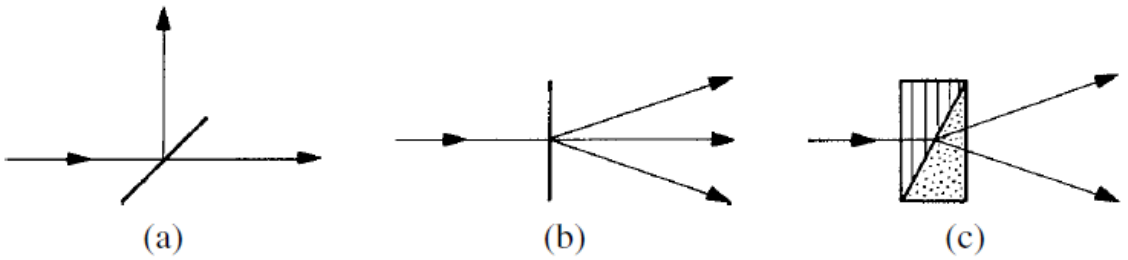


Figura 2.10. Técnicas para la división de amplitud: (a) un divisor de haz, (b) una red de difracción, y (c) un prisma polarizante.

Expansión de laser-beam

El haz de un láser típicamente tiene un diámetro que van desde una fracción de milímetro a unos pocos milímetros y un gaussiano perfil de intensidad dado por la relación:

$$I(r) = \exp\left(-\frac{2r^2}{w_0^2}\right) \qquad \text{Ec. 2.9}$$

donde r es la distancia radial desde el centro del has. En una distancia radial $r = w_0$, la intensidad cae a $1 / e^2$ de eso en el centro del haz.

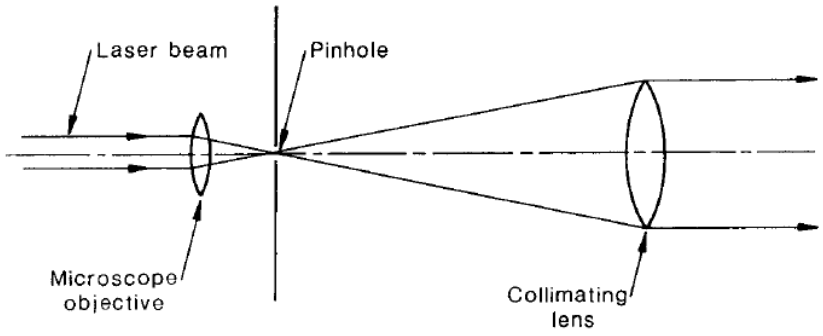


Figura 2.13. Arreglo utilizado para expandir y filtrar espacialmente un rayo láser.

Tal has retiene su perfil gaussiano a medida que se propaga, pero su diámetro efectivo aumenta debido a la difracción Después de atravesar una distancia z desde la cintura del haz (el punto cuyo diámetro del haz es mínimo), se da la distribución de la intensidad por la misma relación con w_0 reemplazado por $w(z)$, donde:

$$w(z) = w_0\left[1 + (\frac{\lambda z}{\pi w_0^2})^2\right]^{1/2} \qquad \text{Ec. 2.10}$$

También se puede mostrar que, a una gran distancia de la cintura del rayo, el ángulo de la divergencia del haz es:

$$\theta = \lambda/\pi w_0 \qquad \text{Ec. 2.11}$$

Muchos interferómetros requieren un haz colimado que llene una apertura mucho más grande. En tal caso, el rayo láser se enfoca con un objetivo de microscopio; luego se puede usar una lente con una abertura adecuada, como se muestra en la Figura 2.13, para obtener un haz colimado. Debido al alto grado de coherencia de la luz láser, el haz expandido exhibe comúnmente patrones de difracción aleatorios producidos por arañazos o polvo en la óptica. Estos efectos se pueden minimizar colocando un agujero de alfiler (filtro espacial) en el foco del objetivo del microscopio. Si la apertura de la el objetivo del microscopio es mayor que $2w_0$, el diámetro del punto focal es:

$$d = 2\lambda f/\pi w_0 \qquad \text{Ec. 2.12}$$

Donde f es la distancia focal del objetivo del microscopio; sin embargo, si solo el parte central del rayo láser (diámetro D) se transmite, el diámetro del punto focal está dado por la relación:

$$d = 2.44\lambda f/D \qquad \text{Ec. 2.13}$$

Las ecuaciones de imagen

La imagen virtual aparece en la posición del objeto original si el holograma es reconstruido con los mismos parámetros como los utilizados en el proceso de grabación. Sin embargo, si uno cambia la longitud de onda o las coordenadas de la reconstrucción punto fuente de onda con respecto a las coordenadas de la fuente de onda de referencia punto utilizado en el proceso de grabación, la posición de la imagen reconstruida se mueve. El desplazamiento de coordenadas es diferente para todos los puntos, por lo tanto, la forma de la reconstrucción el objeto está distorsionada. La ampliación de la imagen puede estar influenciada por los parámetros reconstrucción, también.

Las ecuaciones de imágenes relacionan las coordenadas de un objeto con el punto O con el del punto correspondiente en la imagen reconstruida. Estas ecuaciones son citadas aquí sin derivación, porque se necesitan para explicar técnicas específicas como la Microscopía Holográfica Digital.

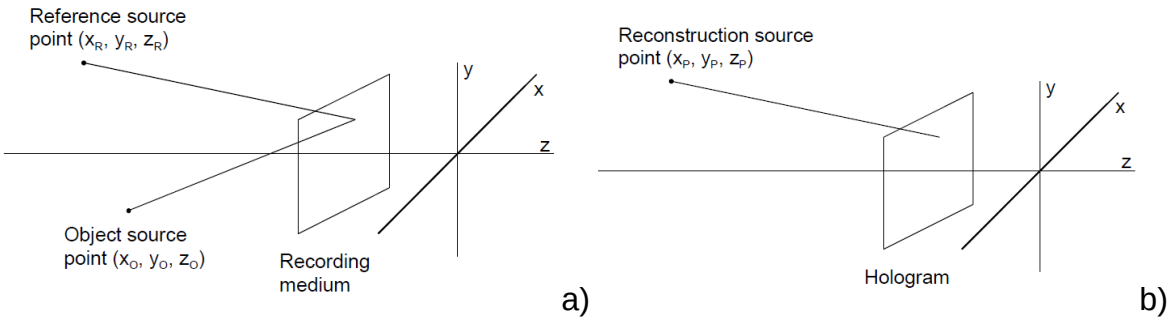


Figura 2.14. a) Grabado de Holograma, b) Reconstrucción.

El sistema de coordenadas se muestra en la figura 2.14 (x_O, y_O, z_O) son las coordenadas del punto del objeto O, (x_R, y_R, z_R)son las coordenadas del punto fuente del onda de referencia utilizada para la grabación de hologramas y (x_P, y_P, z_P) son las coordenadas del punto de origen de la onda de reconstrucción. $\mu = \lambda_1/\lambda_2$ denota la relación entre la longitud de onda de registro λ_1 y la longitud de onda de reconstrucción λ_2 . Las coordenadas de ese punto en la imagen virtual reconstruida, que corresponde al punto del objeto O, son:

$$x_1 = \frac{x_P z_O z_R + \mu x_O z_P z_R - \mu x_R z_P z_O}{z_O z_R + \mu z_P z_R - \mu z_P z_O} \quad \text{Ec. 2.14}$$

$$y_1 = \frac{y_P z_O z_R + \mu y_O z_P z_R - \mu y_R z_P z_O}{z_O z_R + \mu z_P z_R - \mu z_P z_O} \quad \text{Ec. 2.15}$$

$$z_1 = \frac{z_P z_O z_R}{z_O z_R + \mu z_P z_R - \mu z_P z_O} \quad \text{Ec. 2.16}$$

Las coordenadas de ese punto en la imagen real reconstruida, que corresponde al punto del objeto O, son:

$$x_2 = \frac{x_P z_O z_R - \mu x_O z_P z_R + \mu x_R z_P z_O}{z_O z_R - \mu z_P z_R + \mu z_P z_O} \quad \text{Ec. 2.17}$$

$$y_2 = \frac{y_P z_O z_R - \mu y_O z_P z_R + \mu y_R z_P z_O}{z_O z_R - \mu z_P z_R + \mu z_P z_O} \quad \text{Ec. 2.18}$$

$$z_2 = \frac{z_P z_O z_R}{z_O z_R - \mu z_P z_R + \mu z_P z_O} \quad \text{Ec. 2.19}$$

Se puede considerar que un objeto extendido está compuesto por varios objetos puntuales. Las coordenadas de todos los puntos de superficie están descritas por las ecuaciones anteriormente mencionadas. La ampliación lateral de toda la imagen virtual se representa:

$$M_{lat,1} = \frac{dx_1}{dx_0} = \left[1 + z_O\left(\frac{1}{\mu z_P} - \frac{1}{z_R}\right)\right]^{-1} \quad \text{Ec. 2.20}$$

La magnificación lateral de la imagen real resulta:

$$M_{lat,2} = \frac{dx_2}{dx_0} = \left[1 + z_O \left(\frac{1}{\mu z_P} - \frac{1}{z_R}\right)\right]^{-1} \quad \text{Ec. 2.21}$$

La magnificación longitudinal de la imagen virtual es dada:

$$M_{long,1} = \frac{dz_1}{dz_0} = \frac{1}{\mu} \; {M_{lat,1}}^2 \quad \text{Ec. 2.22}$$

La magnificación longitudinal de la imagen real es:

$$M_{long,2} = \frac{dz_2}{dz_0} = \frac{1}{\mu} \; {M_{lat,2}}^2 \quad \text{Ec. 2.23}$$

Hay una diferencia entre la imagen real y la imagen virtual que se debe mencionar: Desde que la imagen real está formada por la onda del objeto conjugada O*, tiene la curiosa propiedad que su profundidad está invertida. Los puntos correspondientes de la imagen virtual (que coinciden con los puntos del objeto original) y de la imagen real se encuentran a distancias iguales desde el plano de holograma, pero en lados opuestos de él. El fondo y el primer plano de la imagen real se intercambian por lo tanto. La imagen real aparece con la "perspectiva equivocada". Se llama imagen pseudoescópica, en contraste con imagen normal u ortoscópica.

Holografía en el eje

Para comprender la aparición simultánea de imágenes reales y virtuales en holografía, tomamos un enfoque que está más cerca de nuestra noción habitual de formación de imágenes con lentes.

Grabación de hologramas

Para simplificar, primero consideramos la grabación de un holograma de un objeto puntual. Un objeto extendido se puede considerar como una colección de objetos puntuales. Dejemos que el punto objeto O estar situado en el eje de la placa fotográfica a una distancia z_1 de (Fig. 2.7a).

La onda de referencia es una onda plana monocromática $R(A_1, k_1)$, incidente normalmente en la placa fotográfica. La onda esférica emitida por el objeto puntual interfiere con la referencia ola en todas partes, en particular, en el plano de la placa fotográfica.

Es obvio, pero aún puede señalarse que, a diferencia del plano de imagen de una lente, luz del objeto puntual (y también de un objeto extendido) en holografía alcanza cada punto de la placa fotográfica. En consecuencia, cada porción del holograma adquiere información completa sobre las distribuciones de amplitud y fase del campo de objeto en forma de un patrón de interferencia. Esto parece introducir una gran cantidad de redundancia en imágenes holográficas, pero las imágenes reconstruidas desde un holograma de gran tamaño son más brillantes y poseen mayor resolución. La disposición de la figura 2.7a es algo similar a la originalmente empleada por Gabor para grabar un holograma (Fig. 2.7b). Usó un filtro óptico y un orificio para mejorar la coherencia de su fuente de luz. Gabor usó un pequeño objeto semitransparente. La luz transmitida por el objeto actúa como la onda y luz de referencia difractada por el objeto constituye la onda del objeto.

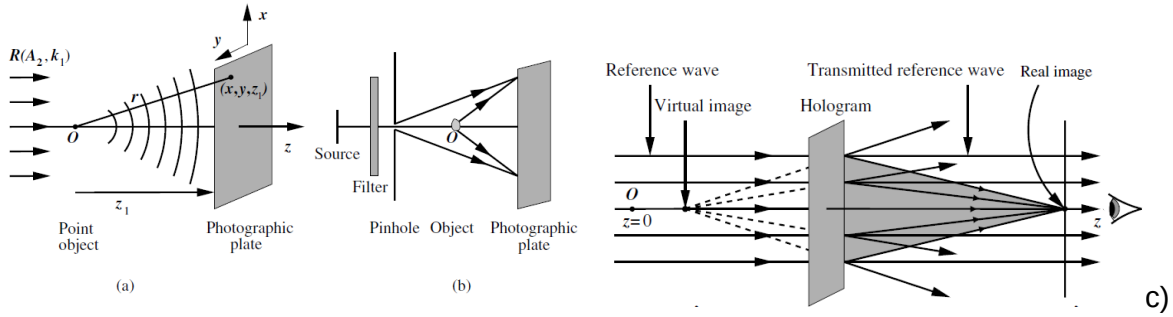
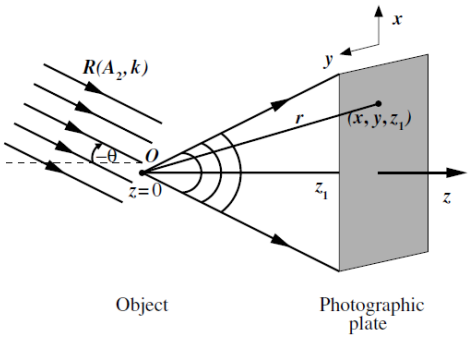


Fig. 2.7: (a) Grabación del holograma en el eje de un objeto puntual con una onda referencia . (b) La disposición de Gabor, O es un objeto semitransparente.c) Imágenes reconstruidas virtuales y reales de un holograma grabado en línea.

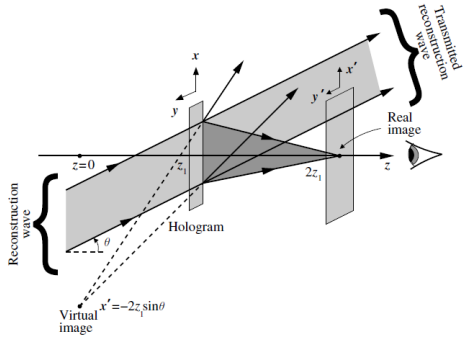
Un holograma puede ser visualizado como una colección de muchas lentes, integradas en una con tantas distancias focales, capaces de producir imágenes nítidas y brillantes de objetos 3D. Dentro de esta interpretación, un holograma puede considerarse como una placa de zona de Fresnel con resultados positivos y negativos acciones de lente y también actuando como un atenuador. De otra manera, permite que el holograma se interprete como una rejilla sinusoidal, que la iluminación coherente genera dos órdenes de difracción además del cero orden. Un holograma actúa como un holograma 2D cuando el espesor de la capa de emulsión en la placa fotográfica es más pequeña que la separación entre consecutivas franjas del patrón de interferencia establecido en la placa fotográfica. Para el presente, se supone que esta condición se cumple.

Holografía fuera del eje

Ahora demostramos que las imágenes reales y virtuales reconstruidas a partir de un holograma grabado con una onda de referencia oblicua que se puede separar. La misma onda de referencia se emplea para la grabación de hologramas y frente de onda de reconstrucción. En la figura 2.8, la onda de referencia forma un ángulo $-\theta$ con el eje de la placa fotográfica.



a)



b)

Fig. 2.8: a) Grabación de hologramas con una onda de referencia fuera del eje. b) Imágenes reconstruidas con una onda de reconstrucción en $+\theta$ a partir de una holograma registrado con una onda de referencia en $-\theta$.

Reconstrucción por a aproximación de Fresnel

Para los valores x e y, así como para los valores ξ y η, que son pequeños en comparación con la distancia d entre el plano de reconstrucción y el CCD, tenemos a una serie de Taylor:

$$\rho = d + \frac{(\xi - x)^2}{2d} + \frac{(\eta - x)^2}{2d} - \frac{1}{8}\frac{\left[(\xi - x)^2 - (\eta - x)^2\right]^2}{d^3} + \cdots \qquad \text{Ec. 2.23}$$

El cuarto término puede ser descuidado, si es pequeño en comparación con la longitud de onda:

$$\frac{1}{8}\frac{\left[(\xi - x)^2 - (\eta - x)^2\right]^2}{d^3} \ll \lambda \qquad \text{Ec. 2.24}$$

O

$$d \gg \sqrt[3]{\frac{1}{8}\frac{\left[(\xi-x)^2-(\eta-x)^2\right]^2}{\lambda}} \qquad \text{Ec. 2.25}$$

Entonces la distancia ρ consiste en términos lineales y cuadráticos:

$$\rho = d + \frac{(\xi-x)^2}{2d} + \frac{(\eta-x)^2}{2d} \qquad \text{Ec. 2.26}$$

Con la aproximación adicional de reemplazar el denominador por d los siguientes resultados de expresión para la reconstrucción de la imagen real:

$$\Gamma(\xi,\eta) = \frac{i}{\lambda d}\, exp\left(-i\frac{2\pi}{\lambda}\, d\right) \ldots \qquad \text{Ec. 2.27}$$

$$\times \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} E_R^*(x,y)h(x,y)exp\left[-i\frac{\pi}{\lambda d}\,((\xi-x)^2+\,(\eta-x)^2\,)\right]\, dxdy$$

Si las multiplicaciones en el argumento de exponencial bajo la integral son llevadas a cabo uno obtiene:

$$\Gamma(\xi,\eta) = \frac{i}{\lambda d}\, exp\left(-i\frac{2\pi}{\lambda}\, d\right) exp\left[-i\frac{\pi}{\lambda d}\,(\xi^2+\eta^2)\right] \ldots \qquad \text{Ec. 2.28}$$

$$\times \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} E_R^*(x,y)h(x,y)exp\left[-i\frac{\pi}{\lambda d}\,(x^2+y^2)\right] exp\left[-i\frac{2\pi}{\lambda d}\,(x\xi+y\eta)\right]\, dxdy$$

Esta ecuación se denomina *Aproximación de Fresnel* o *Transformación de Fresnel* a su similitud matemática con la Transformada de Fourier (ver abajo). Permite la reconstrucción del frente de ondas en un plano detrás del holograma, en este caso en el plano de la imagen real. La intensidad se calcula cuadrando:

$$I(\xi,\eta) = |\Gamma(\xi,\eta)|^2 \qquad \text{Ec. 2.29}$$

La fase es calculada por:

$$\varphi(\xi,\eta) = \arctan\frac{Im[\Gamma(\xi,\eta)]}{Re[\Gamma(\xi,\eta)]} \qquad \text{Ec. 2.30}$$

Re denota la parte real y Im soy la parte imaginaria. La fórmula de reconstrucción para la imagen virtual en la aproximación de Fresnel es:

$$\Gamma(\xi',\eta') = \frac{i}{\lambda d}\, exp\left(-i\frac{2\pi}{\lambda}\, d\right) exp\left[-i\frac{\pi}{\lambda d}\,(\xi'^2+\eta'^2)\right]\, P(\xi',\eta') \ldots \qquad \text{Ec. 2.31}$$

$$\times \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} E_R(x,y)L(x,y)h(x,y)exp\left[-i\frac{\pi}{\lambda d}\,(x^2+y^2)\right] exp\left[i\frac{2\pi}{\lambda d}\,(x\xi'+y\eta')\right]\, dxdy$$

$$= \frac{i}{\lambda d}\, exp\left(-i\frac{2\pi}{\lambda}\, d\right) exp\left[-i\frac{\pi}{\lambda d}\,(\xi'^2+\eta'^2)\right]$$

$$\times \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} E_R(x,y)h(x,y)exp\left[+i\frac{\pi}{\lambda d}\,(x^2+y^2)\right] exp\left[i\frac{2\pi}{\lambda d}\,(x\xi'+y\eta')\right]\, dxdy$$

Para la digitalización de la transformación de Fresnel Eq. (2.28) después de las sustituciones son introducidas [176]:

$$v = \frac{\xi}{\lambda d};\;\; \mu = \frac{\eta}{\lambda d} \qquad \text{Ec. 2.32}$$

Con esto (2.28) se convierte:

$$\Gamma(v,\mu) = \frac{i}{\lambda d}\, exp\left(-i\frac{2\pi}{\lambda}\, d\right) \exp[-i\pi\lambda d(v^2+\mu^2] \qquad \text{Ec. 2.33}$$

$$\times \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} E_R^*(x,y)h(x,y)exp\left[-i\frac{\pi}{\lambda d}\,(x^2+y^2)\right]\, exp[i2\pi(xv+y\mu)]dxdy$$

Una comparación de Eq. (2.33) con la definición de la transformada Fourier bidimensional muestra que la aproximación de Fresnel hasta un factor esférico de fase, es la transformación de Fourier inversa de la función $E_R^*(x,y)h(x,y)exp\left[-i\frac{\pi}{\lambda d}\,(x^2+y^2)\right]$

$$\Gamma(v,\mu) = \frac{i}{\lambda d}\, exp\left(-i\frac{2\pi}{\lambda}\, d\right) \exp[-i\pi\lambda d(v^2+\mu^2] \ldots \qquad \text{Ec. 3.20}$$

$$\times\ \Im^{-1}\left\{E_R^*(x,y)h(x,y)exp\left[-i\frac{\pi}{\lambda d}\ (x^2+y^2)\right]\right\}$$

La función Γ puede digitalizarse si la función de holograma $h\ (x,y)$ se muestrea en una trama rectangular de puntos NxN, con los pasos Δx y Δy a lo largo de las coordenadas. Δx y Δy son las distancias entre los píxeles vecinos en el CCD en la dirección horizontal y vertical. Con estos valores discretos las integrales 3.19 son convertidas en sumas finitas.

$$\Gamma(m,n)=\frac{i}{\lambda d}\ exp\left(-i\frac{2\pi}{\lambda}\ d\right)\exp[-i\pi\lambda d(m^2\Delta v^2+n^2\Delta\mu^2]\ \ldots \qquad \text{Ec. 2.34}$$

$$\times\ \sum_{k=0}^{N-1}\sum_{l=0}^{N-1}\ E_R^*(k,l)h(k,l)exp\left[-i\frac{\pi}{\lambda d}\ (k^2\Delta x^2+l^2\Delta y^2)\right]exp[i2\pi(k\Delta xm\Delta v+l\Delta yn\Delta\mu)]$$

Para m=0, 1,2,…,N-1; n=0, 1,2,…,N-1;

De acuerdo con la teoría de la transformada de Fourier entre Δx, Δy y Δv, $\Delta\mu$ la siguiente relación existe:

$$\Delta v=\frac{1}{N\Delta x}\quad ;\quad \Delta\mu=\frac{1}{N\Delta y} \qquad \text{Ec. 2.35}$$

después de la sustitución:

$$\Delta\xi=\frac{\lambda d}{N\Delta x}\quad ;\quad \Delta\eta=\frac{\lambda d}{N\Delta y} \qquad \text{Ec. 2.36}$$

Usando la ecuación 2.34 convertida a

$$\Gamma(m,n)=\frac{i}{\lambda d}\ exp\left(-i\frac{2\pi}{\lambda}\ d\right)\exp\left[-i\pi\lambda d(\frac{m^2}{\mathrm{N}^2\Delta x^2}+\frac{n^2}{N^2\Delta y^2}\right]\ldots \qquad \text{Ec. 2.37}$$

$$\times\ \sum_{k=0}^{N-1}\sum_{l=0}^{N-1}\ E_R^*(k,l)h(k,l)exp\left[-i\frac{\pi}{\lambda d}\ (k^2\Delta x^2+l^2\Delta y^2)\right]exp\left[i2\pi\left(\frac{km}{\mathrm{N}}+\frac{ln}{\mathrm{N}}\right)\right]$$

Esta es la transformada discreta de Fresnel. La matriz Γ es calculada por la multiplicación $E_R^*(k,l)$ con $h(k,l)$ y $exp\left[-i\frac{\pi}{\lambda d}\ (k^2\Delta x^2+l^2\Delta y^2)\right]$ y aplicando la transformada discreta inversa de Fourier por el producto. El cálculo se hace de la manera más efectiva utilizando el algoritmo de Transformada rápida de Fourier (FFT). El factor frente a la suma solo afecta la fase general y puede descuidarse, si solo la intensidad de acuerdo a Eq. (2.29) es de interés.

Este es también el caso si las diferencias de fase entre hologramas registrados con la misma longitud de onda tienen que calcularse $\Delta\varphi = \varphi_1 + const. - (\varphi_2 + const.) = \varphi_1 - \varphi_2$.

La fórmula discreta correspondiente para la reconstrucción a través de una lente virtual con $f = d / 2$ (Ecuación 2.31) es

$$\Gamma(m,n) = \frac{i}{\lambda d}\, exp\left(-i\frac{2\pi}{\lambda}\, d\right) \exp\left[+i\pi\lambda d\left(\frac{m^2}{\mathrm{N}^2\Delta x^2} + \frac{n^2}{N^2\Delta y^2}\right]\right] \ldots \qquad \text{Ec. 2.38}$$

$$\chi \sum_{k=0}^{N-1}\sum_{l=0}^{N-1} E_R(k,l)h(k,l)exp\left[+i\frac{\pi}{\lambda d}\,(k^2\Delta x^2 + l^2\Delta y^2)\right] exp\left[i2\pi\left(\frac{km}{\mathrm{N}} + \frac{ln}{\mathrm{N}}\right)\right]$$

Un holograma digital típico se muestra en la figura 2.17a. Fue grabado con la geometría de la figura 2.15. El objeto se colocado $d = 1{,}054m$ aparte de la matriz CCD de 1024 × 1024 píxeles con distancias de píxeles $\Delta x = \Delta y = 6.8\mu m$. La longitud de onda es $632.8nm$. La reconstrucción numérica de acuerdo con Eq. (2.28) es demostrado en la figura 2.17b. Una imagen real de los dados utilizados como objeto es notable. El cuadrado brillante en el centro es la onda de reconstrucción no difractada (orden cero) y corresponde al primer término del lado derecho de la ecuación 2.63 referencia [18]. Debido a la geometría fuera de eje, la imagen está espacialmente separada del término de orden cero. La otra imagen (virtual) está fuera de foco en esta reconstrucción. Una propiedad interesante de la holografía es que cada parte de un holograma contiene la información sobre el objeto completo. Esto está ilustrado por los hologramas de figuras 2.18 y 2.19, donde las máscaras negras cubren casi la mitad de las áreas del holograma. Sin embargo todo el cubo es visible en las reconstrucciones sin obstrucciones, figuras 2.18b y 2.19b. Las máscaras son visibles como sombras en los términos de orden cero. La reducción del número de píxeles efectivos conduce a una reducción de la resolución en las imágenes reconstruidas. Esto corresponde al aumento del tamaño de speckle debido a reducción de la apertura en la reconstrucción del holograma óptico. En cuanto a Eq. (2.36) las distancias de píxeles en la imagen reconstruida $\Delta\xi$ y $\Delta\eta$ dependen de la distancia de reconstrucción d elegida para la reconstrucción numérica. Esto es porque Eq. (2.36) corresponde a la resolución limitada de difracción de sistemas ópticos: el holograma es la apertura del sistema óptico con el lado longitud $N\Delta x$. De acuerdo con la teoría de la difracción, se desarrolla un patrón de difracción a una distancia d detrás del holograma $\Delta\xi = \lambda d/N\Delta x$ es por lo tanto el medio diámetro del disco de airy o del diámetro de moteado en el plano de la imagen reconstruida, que limita la resolución. Esto se puede considerar como un

algoritmo de "escalamiento natural", colocando la resolución de la imagen reconstruida por una discreta transformación de Fresnel siempre a el límite físico.

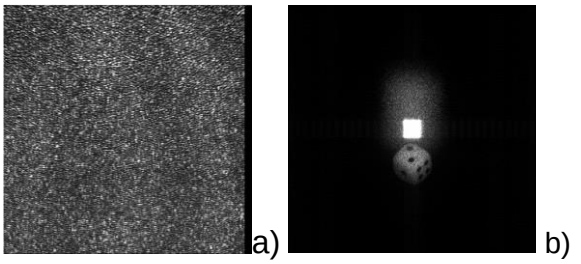


Figura 2.17. a) Holograma, b) Reconstruccion Numerica.

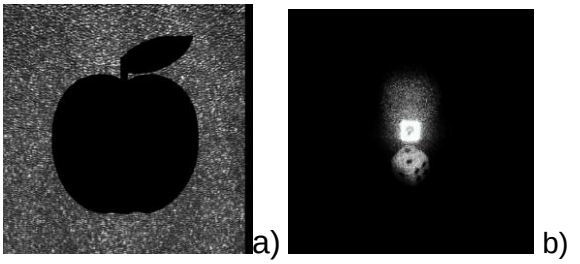


Figura 2.18. a)Holograma digital enmascarado, b)Reconstruccion.

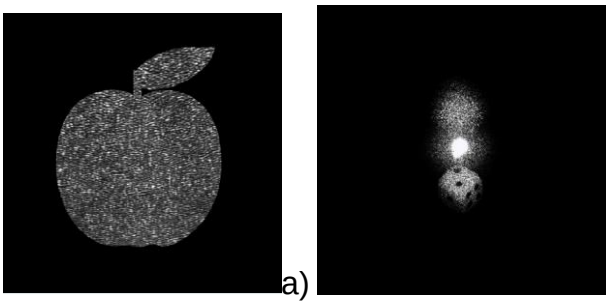


Figura 2.19. Holograma digital enmascarado, b)Reconstruccion.

La integral de Fresnel-Kirchhoff se puede usar para calcular la amplitud compleja en cualquier otro plano. Para calcular una imagen del objeto una lente artificial con $f = d/2$ se introduce en el plano de grabación de acuerdo con. Por medio de la aproximación de Fresnel la ampitud compleja en el plano de la imagen es luego calculado por:

$$E_0(\xi', \eta') = C\ exp\left[+i\frac{\pi}{\lambda d}\ (\xi'^2 + \eta'^2)\right] \ldots \qquad \text{Ec. 2.39}$$

$$\times \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} E_0(x,y)L(x,y)exp\left[-i\frac{\pi}{\lambda d}\ (x^2+y^2)\right]\exp\left[i\frac{2\pi}{\lambda d}(x\xi' + y\eta')\right]dxdy$$

$$= C\ exp\left[+\frac{i\pi}{\lambda d}\ (\xi'^2 + \eta'^2)\right] \ldots$$

$$\times \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} E_0(x,y) exp\left[+i\frac{\pi}{\lambda d}\ (x^2+y^2)\right] \exp\left[i\frac{\pi}{\lambda d}(x\xi' + y\eta')\right] dxdy$$

Donde nuevamente se aplica el sistema de coordenadas de la figura 1.4. Desde la amplitud compleja es conocida en el plano del holograma, también es posible reconstruir el objeto por inversión del proceso de grabación. La grabación del holograma está descrita por

$$E_0(x,y) = \frac{i}{\lambda}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} E_0\ (\xi,\eta)\frac{\exp\left(-i\frac{2\pi}{\lambda}\sqrt{d^2+(\xi-x)^2+(\eta-y)^2}\right)}{\sqrt{d^2+(\xi-x)^2+(\eta-y)^2}} d\xi d\eta \qquad \text{Ec. 2.40}$$

$$= \Im^{-1}\{\Im(E_0(\xi,\eta)) \cdot \Im(g(\xi,\eta,x,y))\}$$

$$g(\xi,\eta,x,y) = \frac{i}{\lambda}\ \frac{\exp\left(-i\frac{2\pi}{\lambda}\sqrt{d^2+(\xi-x)^2+(\eta-y)^2}\right)}{\sqrt{d^2+(\xi-x)^2+(\eta-y)^2}} \qquad \text{Ec. 2.42}$$

$E_0(\xi,\eta)$ es la amplitud compleja de la onda objeto en la superficie emisora, ver la figura 1.4 . Por lo tanto, se puede calcular directamente invirtiendo Eq. (2.40):

$$E_0(\xi,\eta) = \Im^{-1}\left\{\frac{\Im(E_0(x,y)}{\Im(g(\xi,\eta,x,y))}\right\} \qquad \text{Ec. 2.43}$$

La implementación numérica de este método de reconstrucción es crítico debido a la división en Eq. (2.43).

El origen de los FPGA

Para tener una buena idea de la forma en que se desarrollaron los FPGA y las razones por las que aparecieron en escena, es ventajoso considerarlos en el contexto de otras tecnologías relacionadas (Figura 3-1).

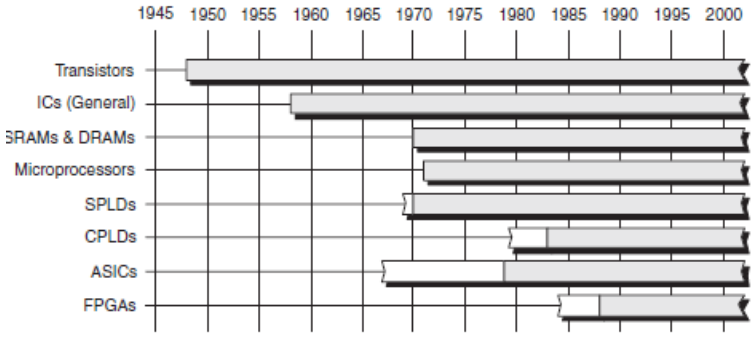


Figure 3-1. Technology timeline (dates are approximate).

Las partes blancas de las barras de la línea de tiempo en esta ilustración indican que, aunque las primeras encarnaciones de estas tecnologías pueden haber estado disponibles, por una u otra razón no fueron recibidas con entusiasmo por los ingenieros que trabajan en las trincheras durante este período. Por ejemplo, aunque Xilinx introdujo el primer FPGA del mundo ya en 1984, los ingenieros de diseño realmente no comenzaron a usar estos pequeños dispositivos con gusto y abandono hasta principios de los 90.

Algo de Historia

El 23 de diciembre de 1947, los físicos William Shockley, Walter Brattain y John Bardeen, trabajando en los Laboratorios Bell en los Estados Unidos, lograron crear el primer transistor: un dispositivo de contacto por puntos formado por germanio.

El año 1950 vio la introducción de un componente más sofisticado llamado transistor de unión bipolar (BJT), que era más fácil y más barato de construir y tenía la ventaja adicional de ser más confiable. A fines de la década de 1950, los transistores se fabricaban de silicio con en lugar de germanio.

Si los BJT se conectan juntos de una cierta manera, las compuertas lógicas digitales resultantes se clasifican como lógica transistor-transistor (TTL).

En 1962, Steven Hofstein y Fredric Heiman en el laboratorio de investigación RCA en Princeton, Nueva Jersey, inventaron una nueva familia de dispositivos llamados transistores de efecto de campo de semiconductores de óxido metálico (MOSFET). A menudo se llaman simplemente FET para abreviar. Hay dos tipos principales de FET llamados NMOS y PMOS. Las compuertas lógicas formadas por transistores NMOS y PMOS conectadas entre sí de manera complementaria se conocen como semiconductores complementarios de óxido de metal (CMOS). 1958 que Jack Kilby, trabajando para Texas Instruments (TI), tuvo éxito en la fabricación de un oscilador de cambio de fase que comprende cinco componentes en una sola pieza de semiconductor. Casi al mismo tiempo que Kilby estaba trabajando en su prototipo, dos de los fundadores del Semiconductor Fairchild - El físico suizo Jean Hoerni y el físico estadounidense Robert Noyce inventaron las técnicas litográficas ópticas subyacentes que ahora se usan para crear transistores, capas aislantes e interconexiones en circuitos integrados modernos(CI). A mediados de la década de 1960, TI introdujo una gran selección de circuitos integrados básicos llamados la serie 54xx ("cincuenta y cuatrocientos") y la serie 74xx ("setenta y cuatrocientos"), que se especificaron para uso militar y comercial, respectivamente. Por ejemplo, un dispositivo 7400 contenía cuatro compuertas NAND de 2 entradas, un 7402 contenía cuatro compuertas NOR de 2 entradas, y un 7404 contenía seis compuertas NOT (inverter). Estos dispositivos de "jalea de frijol", que típicamente tenían alrededor de 3/4 "de largo, 3/8" de ancho, y tenían 14 o 16 pines. A fines de la década de 1960 y comienzos de la década de 1970 proliferaron los nuevos desarrollos en el campo de IC digital. En 1970, por ejemplo, Intel anunció la primera DRAM de 1024 bits (la 1103) y Fairchild introdujo la primera SRAM de 256 bits (la 4100).

Un año más tarde, en 1971, Intel® presentó el primer microprocesador (μP) del mundo, el 4004, que fue concebido y creado por Marciano "Ted" Hoff, Stan Mazor y Federico Faggin.

También conocido como "computadora en un chip", el 4004 contenía solo alrededor de 2.300 transistores y podía ejecutar 60,000 operaciones por segundo.

La razón por la que las tecnologías SRAM y de microprocesadores nos interesan aquí es que la mayoría de los FPGA actuales están basados en SRAM, y algunos de los dispositivos de alta gama actuales incorporan núcleos de microprocesador integrados.

Tecnologías de circuitos integrados

Los tipos de transistores con los que se implementan los circuitos integrados pueden ser transistores bipolares o MOSFET (Metal-Oxide Semiconductor Field-Effect Transistor, transistor de efecto de campo por unión metal-óxido-semiconductor). Una tecnología de circuitos que utiliza MOSFET es la tecnología CMOS (Complementary MOS, MOS complementario). Un tipo de tecnología de CI de función fija que utiliza los transistores bipolares es la TTL (Transistor-Transistor Logic, lógica transistor-transistor). BiCMOS utiliza una combinación de las tecnologías CMOS y TTL.

Todas las puertas y otras funciones se pueden implementar con cualquier tipo de tecnología de circuitos. Generalmente, los circuitos SSI y MSI están disponibles en CMOS y en TTL. LSI, VLSI y ULSI suelen implementarse con tecnología CMOS o NMOS, porque requieren una menor superficie de chip y consumen menos potencia.

Clasificación de los CI de función fija según su complejidad

Los circuitos integrados digitales de función fija se clasifican según su complejidad. A continuación se enumeran de menor a mayor complejidad. La clasificación por complejidad establecida aquí para SSI, MSI, LSI, VLSI y ULSI está generalmente aceptada, aunque las definiciones pueden variar de una fuente de información a otra.

- Integración a pequeña escala (SSI, Small-Scale Integration). Describe los CI de función fija que contienen hasta diez puertas equivalentes en un mismo chip, e incluyen puertas básicas y flip-flops.

- Integración a media escala (MSI, Medium-Scale Integration). Describe los CI que contienen entre 10 y 100 puertas equivalentes en un mismo chip. Incluyen funciones lógicas como codificadores, decodificadores, contadores, registros, multiplexores, circuitos aritméticos, memorias pequeñas y otras.
- Integración a gran escala (LSI, Large-Scale Integration). Es una categoría de los CI que incluyen entre 100 y 10.000 puertas equivalentes por chip, incluyendo memorias.
- Integración de muy gran escala (VLSI, Very Large-Scale Integration). Describe los CI con un número de puertas equivalentes desde 10.000 hasta 100.000 por chip.
- Integración a ultra escala (ULSI, Ultra Large-Scale Integration). Describe memorias de gran capacidad, grandes microprocesadores y computadoras en un solo chip. Esta categoría designa los CI que contienen más de 100.000 puertas equivalentes por chip.
- **Núcleos de procesador incrustados (duro y blando)**
- Casi cualquier parte de un diseño electrónico se puede realizar en hardware (usando compuertas lógicas y registros, etc.) o software (como instrucciones para ser ejecutadas en un microprocesador). Uno de los principales criterios de partición es qué tan rápido desea las diferentes funciones para realizar sus tareas:
- _ Lógica de picosegundos y nanosegundos: Esto tiene que ejecutarse increíblemente rápido, lo que exige que se implemente en hardware (en la estructura de FPGA).
- _ Lógica de microsegundos: esto es bastante rápido y puede implementarse en hardware o software (este tipo de lógica es donde pasa la mayor parte de su tiempo decidiendo adónde ir)....
- Lógica de milisegundos: esta es la lógica utilizada para implementar interfaces tales como la lectura de las posiciones de los interruptores y los diodos emisores de luz (LED) que parpadean. Es un dolor que ralentiza el hardware para implementar este tipo de función (usando enormes contadores para generar retrasos, por ejemplo).

- Por lo tanto, a menudo es mejor implementar estas tareas como un código de microprocesador (porque los procesadores le dan una velocidad pésima, en comparación con el hardware dedicado, pero una complejidad fantástica).
- El hecho es que la mayoría de los diseños hacen uso de microprocesadores de una forma u otra. Hasta hace poco, estos aparecían como dispositivos discretos en la placa de circuito. Últimamente, se han puesto a disposición FPGA de gama alta que contienen uno o más microprocesadores integrados, que generalmente se conocen como núcleos de microprocesador. En este caso, a menudo tiene sentido mover todas las tareas que solía realizar el microprocesador externo en el núcleo interno. Esto proporciona una serie de ventajas, entre ellas, que ahorra el costo de tener dos dispositivos; elimina gran cantidad de pistas, almohadillas y pines en la placa de circuito; y hace que el tablero sea más pequeño y liviano.

La gran mayoría de estos sistemas también hacen uso de un microprocesador de uso general, o µP, para realizar una variedad de aplicaciones de control y procesamiento de datos. Esto a menudo se denomina unidad de procesamiento central (CPU) o unidad de microprocesador (MPU). Hasta hace poco, la CPU y sus periféricos generalmente aparecían en forma de chips discretos en la placa de circuito. Aquí hay un número casi infinito de escenarios posibles, pero los dos principales implican la forma en que la CPU está conectada a su memoria (Figura 13-1).

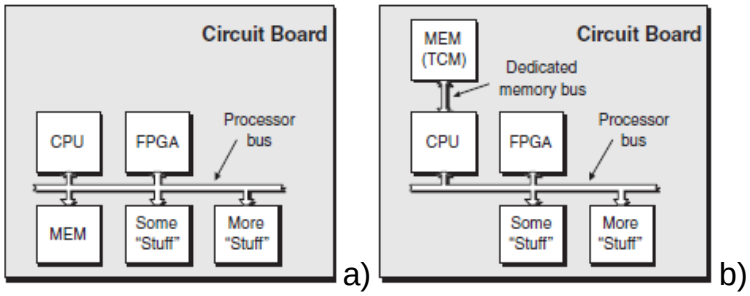


Figura 13-1. Dos escenarios en el nivel de la placa de circuito, (a) Memoria conectada a la CPU a través de bus de procesador de propósito general, (b) Memoria fuertemente acoplada (TCM) conectado a la CPU a través del bus dedicado.

En ambos escenarios, la CPU está conectada a un FPGA y algunas otras cosas a través de un bus de procesador de propósito general. (Por "cosas" nos referimos predominantemente a dispositivos periféricos tales como temporizadores de contador, controladores de interrupción, dispositivos de comunicaciones, etc.) En algunos casos, la memoria principal (MEM) también se conectará a la CPU por medio del bus del procesador principal, como se muestra en la figura 13-1a (en realidad, esta conexión será a través de un periférico especial llamado controlador de memoria, que no se muestra aquí porque estamos tratando de mantener las cosas simples).

Alternativamente, la memoria se puede conectar directamente a la CPU por medio de un bus de memoria dedicado, como se muestra en la Figura 13-1b). El punto es que presentar la CPU y sus diversos dispositivos periféricos en forma de chips dedicados en la placa de circuito. Podría decirse que también afecta la confiabilidad de la placa porque cada junta de soldadura (punto de conexión) es un mecanismo de falla potencial. Una alternativa es incorporar la CPU junto con algunos de sus periféricos en el FPGA mismo (Figura 13-2).

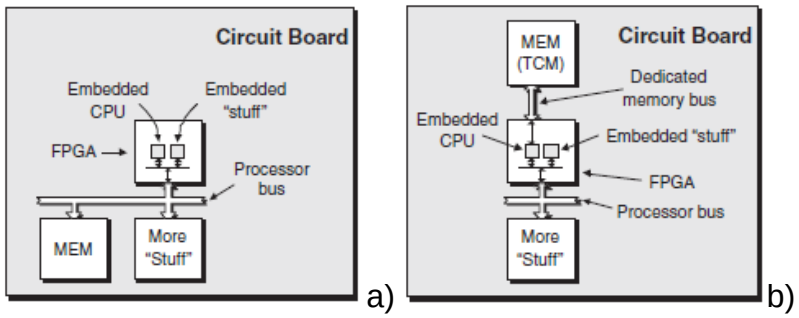


Figura 13-2. Dos escenarios en el nivel de FPGA.a) Memoria conectada a la CPU a través de bus de procesador de propósito general, (b) Memoria fuertemente acoplada (TCM) conectado a la CPU a través del bus dedicado.

Es común que una cantidad relativamente pequeña de memoria utilizada por la CPU se incluya localmente en el FPGA. En el momento de escribir esto, sin embargo, es raro que toda la memoria de la CPU se incluya en el FPGA.

Con respecto al tema de software del diseño, esto podría ser algo tan simple como una máquina de estado utilizada para controlar una interfaz de nivel humano (leyendo el estado de los interruptores y controlando los dispositivos de visualización). Aunque la máquina de estado en sí misma puede ser bastante complicada, este nivel de software ciertamente no es de ciencia espacial. En el otro extremo del espectro, uno podría tener requisitos de software increíblemente complejos, que incluyen:

- Rutinas de inicialización del sistema y una capa de abstracción de hardware
- Un conjunto de pruebas de diagnóstico de hardware
- Un sistema operativo en tiempo real (RTOS)
- Controladores de dispositivo RTOS
- Cualquier código de aplicación incrustado

Este código generalmente se capturará en C / C ++ y luego se compilará en las instrucciones de la máquina que se ejecutarán en el núcleo del procesador (en casos extremos donde se intenta exprimir la última gota del rendimiento del diseño, las rutinas pueden ser hechas a mano en código ensamblador). Al mismo tiempo, los ingenieros de diseño de hardware generalmente capturarán sus partes del diseño en el Nivel de abstracción RTL usando VHDL o Verilog (o SystemVerilog).

El proceso de programación

En muchos aspectos, los primeros días de los PLD fueron el equivalente de los ingenieros de diseño a la Edad Media. La especificación para un nuevo dispositivo típicamente comenzó su vida en la forma de un diagrama esquemático (o máquina de estados). Estos diagramas se crearon usando lápiz y papel porque las herramientas de captura de diseño de electrónico asistida por computadora, en la forma que las conocemos hoy en día, realmente no existían en ese momento.

Una vez que un diseño se había capturado en forma de diagrama, se convertía a mano en un equivalente tabular y luego se escribía en un archivo de texto. Entre otras cosas, este archivo de texto definió qué fusibles iban a explotarse o qué anticongelantes se iban a cultivar. En aquellos días de antaño, el archivo de texto se escribía directamente en una caja especial llamada programador de dispositivos, que luego se usaba para programar el chip. A medida que pasaba el tiempo, sin embargo, se hizo común crear el archivo en una computadora host, que lo descargó y controló según el programador del dispositivo.

En 1980, un comité del Consejo Conjunto de Ingeniería de Dispositivos Electrónicos (JEDEC) -parte de la Asociación de la Industria Electrónica- propuso un formato estándar para los archivos de texto de programación PLD. Por la misma época, John Birkner, el hombre que concibió las primeras PAL y logró su desarrollo, creó PAL Assembler (PALASM). PALASM se refirió tanto a un lenguaje de descripción de hardware (HDL) rudimentario como a una aplicación de software.

Aunque PALASM, ABEL y CUPL son los más conocidos de los primeros HDL, hubo muchos otros, como Automated Map y Zap of Equations (AMAZE) de Signetics. Estos lenguajes simples y herramientas asociadas allanaron el camino para los HDL de nivel superior (como Verilog y VHDL) y las herramientas (como la síntesis lógica) que se utilizan para los diseños ASIC y FPGA actuales.

La lógica programable requiere tanto hardware como software. Los dispositivos lógicos programables pueden programarse para que el fabricante o el usuario pueda llevar a cabo funciones lógicas específicas. Una ventaja de la lógica programable frente a la lógica fija es que los dispositivos utilizan menos espacio de la tarjeta de circuito impreso para una cantidad equivalente de lógica. Otra ventaja es que, con la lógica programable, los diseños se pueden modificar fácilmente sin tener que recablear o reemplazar componentes. Además, generalmente un diseño lógico se puede implementar más rápidamente y con menos coste utilizando circuitos lógicos programables en lugar de los CI de función fija.

Puede pensarse en un SPLD, un CPLD o una FPGA como en un “circuito en blanco” en el que se va a implementar un circuito o sistema específico utilizando un determinado proceso. Este proceso requiere tener instalado un paquete de desarrollo software en una computadora que permita implementar un diseño de circuito en el chip programable. Las computadoras deben poder interaccionar con una tarjeta de desarrollo o con una utilidad de programación que contenga el dispositivo, como se ilustra en la Figura 1.38.

En el proceso de implementar un diseño lógico digital en un dispositivo lógico programable son necesarios varios pasos, lo que se denomina diagrama de flujo del diseño. En la Figura 1.39 se presenta un diagrama de bloques de un proceso típico de programación. Como se indica, el flujo de diseño tiene acceso a una biblioteca de diseño.

COMPILACION DE UN PASO PARA UN KERNEL UN-PASO

Por default AOC compila el kernel OpenCL y crea el archivo de configuración de hardware en un simple paso. El escoger esta opción significa si la aplicación OpenCL requiere mínimas optimizaciones.

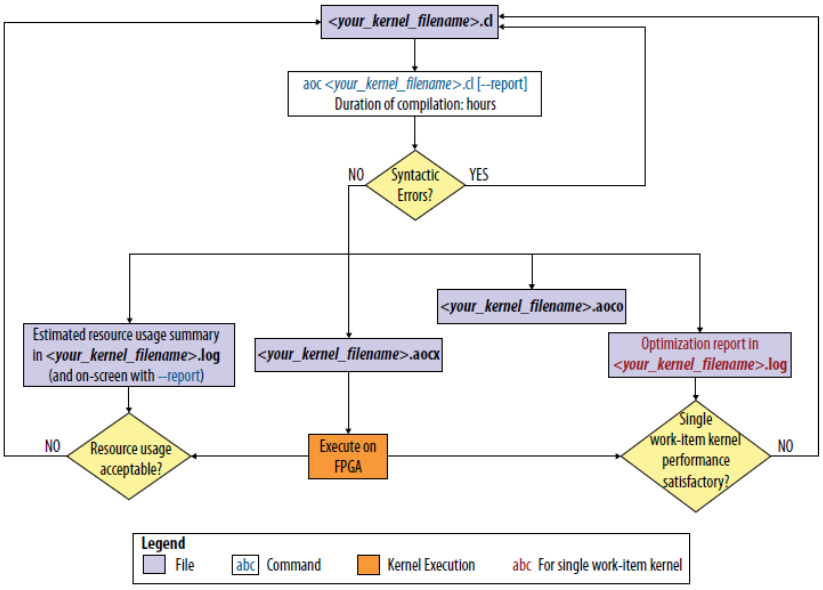


Figura 5.25. Flujo del kernel OpenCL que tiene una compilación de un paso.

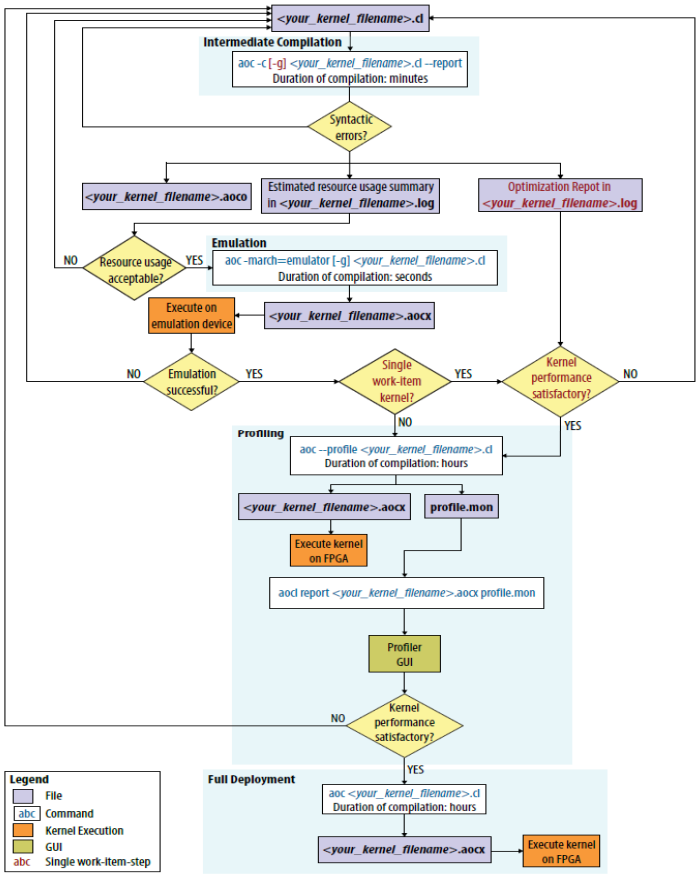
Con una compilación exitosa tenemos generados dos archivos, un *.aoco, un *.aocx, además de un archivo *.log que nos proporciona el estimado de uso de recursos, resumen proporciona una valoración preliminar del área usada.

DISEÑO DE FLUJO ALTERA SDK PARA OPENCL MULTI-PASO

Este diseño de puede elegir si lo que se desea iterar sobre el diseño del kernel OpenCL para implementar optimizaciones de Mejora-Desempeño.

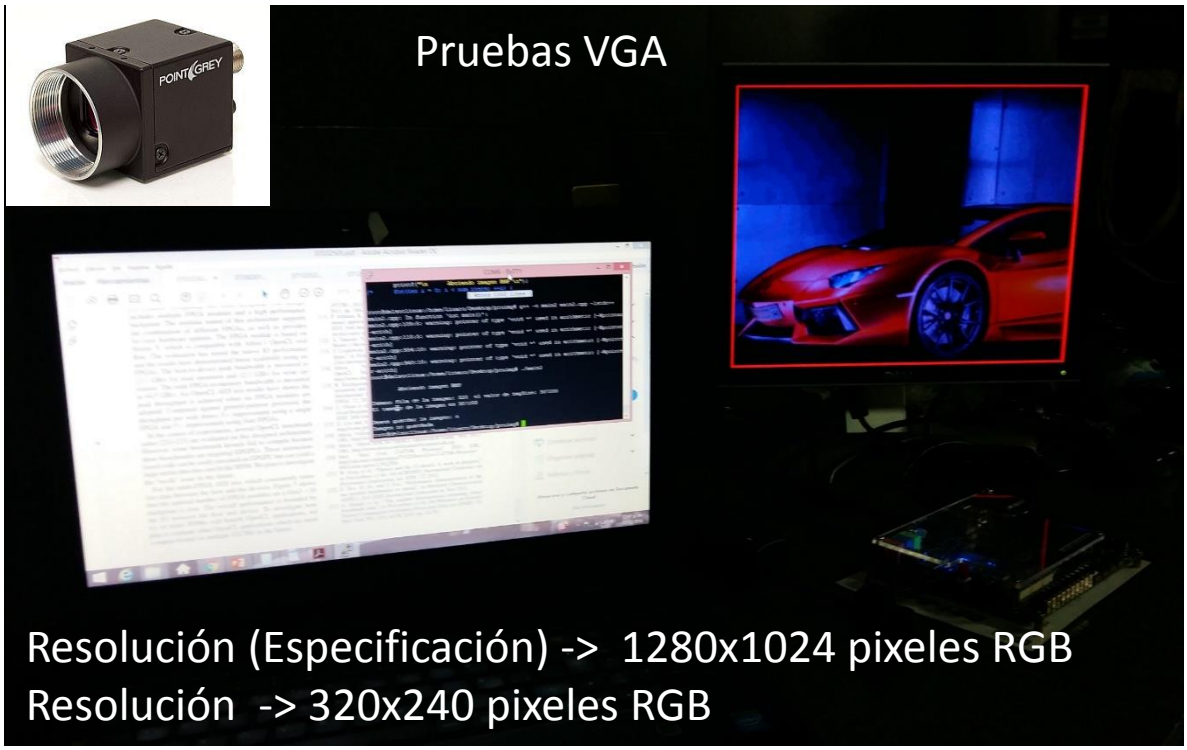
La figura 5.26 muestra los estados en el diseño de flujo AOCL. Los pasos en el diseño de flujo sirven como puntos de referencia para identificar errores funcionales y desempeño de cuellos de botella. Estos pasos permiten modificar el código kernel sin desarrollar una completa compilación después de cada iteración.

Figura 5.26 El diseño de flujo AOCL.



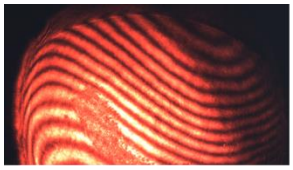
El flujo de diseño de AOCL incluye los siguientes pasos:

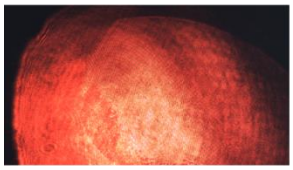
1. Compilación Intermedia: en este paso se verificar errores sintácticos. Por lo tanto genera un archivo *.aoco sin construir un archivo de configuración de hardware. El resumen estimado de recursos usados en un archivo *.log una comprensión dentro del tipo de optimización que se puede realizar al kernel.
2. Emulación: evalúa la funcionalidad de tu kernel OpenCL ejecutándolo en uno o múltiples dispositivos de emulación sobre un Host x86-64.
3. Perfilacion: instruyen al AOC para instrumentar el desempeño de los contadores en código Verilog dentro del archivo *aocx. Durante la ejecución el desempeño de los contadores colectan información del desempeño que se puede revisar en el Perfilador GUI.
4. Despliegue completo: si se está satisfecho con el desempeño del kernel OpenCL a través del diseño de flujo, se puede desarrollar una compilación completa y ejecutar el *aocx en el FPGA.



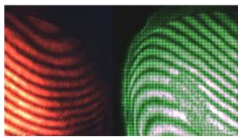
- Adquisición de imágenes a partir de la cámara CCD USB PointGray.

Pruebas





Un error de captura



- Instalación de driver Controlador para la adquisición en FPGA DE1-SoC.
- Se capturaron imágenes por medio de la cámara PointGray con una resolución de 1920x1080 pixeles formato RGB 24bits y tamaño en megas de 6M para cada una, la cámara se conectó directamente al FPGA DE1-SoC.

ANEXO 3. GPU, FPGA y OPENCL.

Los microprocesadores basados en una sola unidad de procesamiento central (CPU), como los de las compañías. Intel® Pentium® familia y la familia AMD® Opteron®, impulsaron un rápido aumento de rendimiento y reducciones de costos en aplicaciones informáticas durante más de dos décadas. Estos microprocesadores trajeron Giga ($1x10^9$) operaciones de punto flotante por segundo (GFLOPS) al escritorio y cientos de GFLOPS a servidores de clúster; por mencionar algunos ejemplos, para 2010 se encontraba en el top de las supercomputadores la Tianjin construido por el Centro Nacional de Supercomputación de China, quien surgió de la oscuridad total para hacerse con la posición de liderazgo entre las supercomputadoras con mejor rendimiento del mundo. Con una velocidad máxima de computación registrada de 2,566 TFLOPS (con trillones de operaciones de coma flotante por segundo, es casi un 50% más rápido que el superordenador Jaguar de Cray). El Tianhe-1A es otro supercomputador con una máxima velocidad de 2566 TFLOPS, con elementos de procesamiento tales como 14,336 Intel Xeon CPUs y 7,168 Nvidia Tesla GPUs, y que está ubicada en el National Supercomputing Center in Tianjin [1]. En 2017 surgió la Sunway TaihuLight con 10, 649,600 núcleos, memoria de 1, 310,720 GB y una velocidad de procesamiento de 93,014.6 TFLOPS; en México también se cuenta con supercomputadoras igual de poderosas entre las que podemos mencionar la ABACUS comparable a las características de un equipo de la NASA, adquirida por el Cinvestav en 2014 y que se encontraba entre las 150 más rápidas de ese año. Con procesadores Intel Xeon E5, el equipo cuenta con 8,904 núcleos además de 100 GPU K40 de Nvidia, lo que junto con 1.2 PB (petabytes) de almacenamiento y 40 TB de memoria RAM lo hace capaz de alcanzar 429 Teraflops (operaciones de coma flotante por segundo) [2]. Lo que es pertinente mencionar aquí como elemento revolucionario de la Tianhe-1ª, por ejemplo, es el uso de GPU para realizar rutinas no-gráficas GPGPU o GPU de propósito general. Antes de 2010, la computación GPGPU se consideraba una novedad en el mundo de la computación de alto rendimiento y no merecía atención seria. Este implacable impulso de la mejora del rendimiento ha permitido que el software de la aplicación

brinde más funcionalidad, tenga mejores interfaces de usuario y genere resultados más útiles. Los usuarios, a su vez, demandan aún más mejoras una vez que se acostumbren a estas mejoras, creando un ciclo positivo para la industria informática.

Prácticamente todos los proveedores de microprocesadores han cambiado a modelos en los que se usan unidades de procesamiento múltiples, denominadas núcleos de procesador en cada chip para aumentar el poder de procesamiento. Un ejemplo actual es el reciente Intel® Microprocesador Core™ I7, que tiene cuatro núcleos de procesador, cada uno de los cuales es un procesador que implementa un conjunto completo de instrucciones x86; el microprocesador admite Hyper-Threading con dos subprocesos de hardware y está diseñado para maximizar la velocidad de ejecución de los programas secuenciales. Este interruptor ha ejercido un tremendo impacto en la comunidad de desarrolladores de software. Actualmente se tiene el Microprocesador Core™ I9-7960X con 16 núcleos, también con Hyper-Threading y Turbo Boost 2.0 entre otras características, aunque con costos algo elevados [3].

Estos desarrollos tecnológicos tanto en GPU como en microprocesadores han marcado una de las principales tendencias en la arquitectura informática: desde estos CPU multinúcleo hasta la computación heterogénea (con el uso de CPU y GPU para representar gráficos 3D en computadoras), el uso de GPU para rutinas no-gráficas se ha denominado informática de GPU de propósito general (o computación GPGPU). Hoy en día, los ingenieros y académicos están llegando a la conclusión de que los sistemas de CPU-GPU representan el futuro de la supercomputación, teniendo en cuenta además que las GPU están diseñadas como motores de computación numérica, y no funcionarán bien en algunas tareas en las que las CPU están diseñadas para funcionar bien; por lo tanto, se debe esperar que la mayoría de las aplicaciones utilicen tanto CPU como GPU, ejecutando las partes secuenciales en la CPU y las partes numéricamente intensivas en las GPU. Por otro lado una buena solución con expectativas de desarrollo, considera además las nuevas arquitecturas SoC que combinan el rendimiento y el software integrado de un procesador de aplicaciones ARM multi-core con la flexibilidad de la estructura

FPGA de Intel, para extender sin problemas las capacidades incorporadas a través del límite CPU-FPGA [4]; entre las ventajas que tiene esta tendencia está que las señales entre el procesador y el FPGA ahora residen en el mismo silicio, de modo que la comunicación entre los dos consume sustancialmente menos energía en comparación con el uso de chips separados. La integración de miles de conexiones internas entre el procesador y el FPGA conducen a un ancho de banda sustancialmente más alto y una latencia más baja en comparación con una solución de dos chips. Anteriormente, la falta de un procesador ARM había sido una barrera para el uso de la tecnología FPGA para la producción completa, pero esta nueva generación de FPGA SoC ofrece procesadores ARM Cortex-A9 completamente funcionales, totalmente compatibles, de alto rendimiento y doble núcleo, hasta 1 GHz con la tecnología de proceso de fabricación actual de 28nm (i.e. dimensión del área del ancho de puerta del transistor).

La necesidad de una informática heterogénea está conduciendo a nuevos lenguajes de programación para explotar el nuevo hardware. Un ejemplo es OpenCL primero desarrollado por Apple, Inc. OpenCL es un estándar que define un conjunto de tipos de datos, estructuras de datos y funciones que aumentan C y C ++; es un marco para escribir programas que se ejecutan a través de plataformas heterogéneas que constan de CPU, GPU, DSP, FPGA y otros tipos de procesadores. OpenCL incluye un lenguaje para desarrollar núcleos ó kernels (i.e. software de funciones que se ejecutan en dispositivos de hardware) así como API que permiten a un programa principal controlar los kernels. OpenCL permite la informática paralela utilizando el paralelismo basado en tareas y datos. Las aplicaciones OpenCL pueden dirigirse a varios dispositivos a la vez, y estos dispositivos no tienen que tener la misma arquitectura o incluso el mismo proveedor, entre sus aplicaciones se encuentra una implementación que propone el desarrollo de un código genérico paralelo de GPU simple para la solución numérica de un sistema de ecuaciones diferenciales ordinarias de primer orden (ODE) basadas en el modelo OpenCL [5].

Los sistemas de hardware Nvidia GPU's programados con OpenCL en los métodos de Lattice Boltzmann (LB), son ampliamente utilizados en la dinámica de fluidos computacional para describir flujos en dos y tres dimensiones, basados en la dinámica sintética de las poblaciones que se sientan en los sitios de una red discreta [6].

Los dispositivos informáticos reconfigurables pueden aumentar el rendimiento de los algoritmos intensivos en cómputo implementando arquitecturas de coprocesador específicas de la aplicación. El costo de energía para esta ganancia de rendimiento es a menudo un orden de magnitud menor que el de las CPU y GPU modernas.

Las compañías de software también han estado ocupadas, desarrollando software que permite que trozos de código de computadora conocidos como hilos (threads) se ejecuten en una verdadera forma paralela. Los hilos pueden ejecutarse por separados núcleos de procesador en lugar del pseudo-paralelismo del pasado en el que los hilos no se ejecutaban en núcleos separados, sino que el sistema operativo los segmentaba temporalmente para parecer que se ejecutaban en paralelo.

A continuación se mencionarán algunas de las aplicaciones e implementaciones con el uso FPGA. Los FPGA son intrínsecamente paralelos, por lo que encajan perfectamente con las capacidades de cómputo paralelo de OpenCL, en el área de comunicaciones a través de la interfaz PCIe entre un FPGA-GPU en comunicación directa, demostrando rendimiento de hasta 2,4 GB/s en una configuración Gen2.1x8 [7]. Recientemente, los principales proveedores de FPGA (Altera y Xilinx) han lanzado herramientas propias con diseño de alto nivel, con potencial para el rápido desarrollo de aceleradores personalizados basados en FPGA, en la ref. [8] se hace una comparación del diseño de hardware FPGA de alto nivel para resolver sistemas lineales tri-diagonales evaluados con el Kit de desarrollo de software OpenCL de Altera y la herramienta de síntesis de alto nivel Vivado de Xilinx. La codificación de imágenes y videos es un área bien estudiada de la informática. Las técnicas de codificación tienen como objetivo reducir la redundancia dentro de los datos de imagen y video para que la cantidad de datos enviados a través de un canal se pueda minimizar.

Una evaluación de la técnica de compresión fractal entre CPUs, GPUs y FPGA ha mostrado una eficiencia 3x en comparación del GPU y 114x más rápido que un CPU [9]. Incrustar la reconstrucción en 3D en tiempo real de una escena a partir de un sensor de profundidad de bajo costo puede mejorar el desarrollo de tecnologías en los dominios de la realidad aumentada, la robótica móvil y más; sin embargo, las implementaciones actuales requieren una computadora con una poderosa GPU, lo que limita sus posibles aplicaciones con requisitos de baja potencia. Para implementar la reconstrucción 3D de baja potencia se han incorporado dos algoritmos de reconstrucción en 3D demostrando que OpenCL puede ser un método viable para desarrollar aplicaciones FPGA mediante la modificación de una versión de código abierto (proyecto Microsoft KinectFusion) para ejecutar parcialmente en un FPGA [10]. En lo que respecta a estudios realizados sobre el rendimiento del particionamiento de datos (i.e. operación central ampliamente utilizada en bases de datos relacionales) y debido a los accesos aleatorios a la memoria, la partición de datos consume mucho tiempo y puede convertirse en un cuello de botella importante para los operadores de bases de datos, como las uniones hash. El enfoque propuesto puede lograr una aceleración aproximada de 10.7x veces respecto a la implementación OpenCL de última generación dentro del estado del arte [11].

Flujo de programación Altera SDK para OpenCL

Con Altera SDK para OpenCL se programa (configura) un FPGA con una aplicación OpenCL en un proceso de dos pasos. Con el Compilador Fuera de Línea (AOC) se compilan los Kernels OpenCL a partir de un archivo *.cl. En el lado Host el compilador C, compila la aplicación Host y entonces se enlaza con los kernels OpenCL.

Un archivo fuente OpenCL (*.cl) contiene nuestro código fuente OpenCL. El AOC agrupa uno o más Kernels en un archivo temporal para compilarlo y entonces generar dos archivos y carpetas:

- Un archivo Altera Offline Compiler Object (.aoco), es un archivo de objeto intermedio que contiene información para etapas posteriores de la compilación.

- Un archivo Altera Offline Compiler Executable (.aocx), es el archivo de configuración del hardware y contiene la información necesaria en el tiempo de ejecución.

Finalmente se genera una carpeta con que contiene los datos necesarios para generar el archivo *.aocx.

El AOC crea el archivo *.aocx, también incorpora información del archivo *.aoc dentro del *.aocx durante la compilación del hardware. El archivo *.aocx contiene datos que la aplicación host usa para crear objetos de programa para la tarjeta FPGA. La aplicación Host abre estos objetos de programa en la memoria. En tiempo de ejecución del Host llama estos objetos de programa de la memoria y programa la tarjeta como se requiere.

El AOC puede crear el archivo de configuración de hardware en un proceso de un solo paso o multipasos. La complejidad del kenel que se compile dicta la opción de compilación AOC de la implementación.