PROYECTO INTEGRADOR DE LA
CARRERA DE INGENIERIA NUCLEAR

Velocimetría PIV en tiempo real basada en lógica programable FPGA

José Miguel Iriarte Muñoz

Fabián J. Bonetto
Co-Director

Damián Dellavale
Co-Director

San Carlos de Bariloche
Junio de 2008

Instituto Balseiro
Universidad Nacional de Cuyo
Comisión Nacional de Energía Atómica
Argentina

Por hacer tan bello este mundo,
a mis padres Miguel y Loren,
a mis hermanos María y Lucas, y a mis abuelos

RESUMEN

La velocimetría por imágenes de partículas (PIV), basada en plano laser, es una potente herramienta de medición en dinámica de fluidos, capaz de medir sin grandes errores, un campo de velocidades distribuido en líquidos, gases y flujo multifase. Los altos requerimientos computacionales de los algoritmos PIV dificultan su empleo en tiempo-real. En este trabajo presentamos el diseño de una plataforma basada en tecnología FPGA para capturar video y procesar en tiempo real el algoritmo de correlación cruzada bidimensional. Mostramos resultados de un primer abordaje de la captura de imágenes y procesamiento de un campo físico de velocidades en tiempo real.

Palabras claves: velocimetría, tiempo real, PIV, plano laser, dinámica de fluidos, FPGA, correlación cruzada bidimensional, procesamiento de imágenes

ABSTRACT

Particle image velocimetry (PIV), based on laser sheet, is a method for image processing and calculation of distributed velocity fields. It is well established as a fluid dynamics measurement tool, being applied to liquid, gases and multiphase flows. Images of particles are processed by means of computationally demanding algorithms, what makes its real-time implementation difficult. The most probable displacements are found applying two dimensional cross-correlation function. In this work, we detail how it is possible to achieve real-time visualization of PIV method by designing an adaptive embedded architecture based on FPGA technology. We show first results of a physical field of velocity calculated by this platform system in a real-time approach.

Key words: velocimetry, real-time, PIV, laser sheet, fluid dynamics, FPGA, two dimensional cross correlation, image processing

Agradecimientos

Quiero reconocer a la persona que más influyó en las decisiones que me condujeron a seguir ingeniería y hoy presentar esta tesis: Miguel Iriarte. Gracias Viejo. Gracias por las visitas al aeropuerto de Mosconi, gracias por el globo terráqueo, gracias por el libro de Asimov, gracias por el viaje a Cabo Cañaveral, gracias por señalarme donde vuelan los cóndores y donde las gaviotas. Gracias por enseñarme que seguir recetas no siempre es mejor, y que este mundo avanza gracias a constructores y loquitos sueltos que no se olvidan de los demás. Este agradecimiento se extiende a toda mi familia por el apoyo de siempre. Gracias Ma por hacerme llegar siempre tu cariño. Gracias a mis hermanos por ser los mejores amigos.

También quiero dar las gracias a mi amigo Sergio Paredi, un gran ingeniero y empresario del norte argentino, por ser un ejemplo en todo mi andar académico. Gracias Sergio por tu guía en las Ferias de Ciencias que atacamos con Darío, esos libros de acústica y comunicaciones digitales, las anécdotas de Helicóperos Marinos y los tantos consejos.

No voy a olvidarme de Vicente Iriarte, quien siempre supo transmitirme con entusiasmo el potencial de la electrónica y las maravillas que esconde el átomo. Gracias tío por el libro de Resnik, el Sadosky Guber, y los tantos sobre televisión.

Sobre el trabajo aquí presentado, quiero destacar la enorme colaboración y asesoramiento de mi co-director Damián Dellavale. Gracias Damián por tus horas dedicadas al proyecto, tu genial tutoría y conocimiento que tanto realimentó al trabajo, y la intachable predisposición. Un amigo sin dudas.

Agradezco a mi co-director Fabián Bonetto por transmitirme mucha de su experiencia y por mostrarme continuamente como un tecnólogo se plantea los problemas y bosqueja soluciones. Una gran usina de ideas.

Agradezco a todos los miembros del Laboratorio de Cavitación y Biotecnología, por ser un magnífico grupo humano y científico, con quienes confirmé que a través de buenas ideas y mucho trabajo se llega con grandes aportes a las primeras planas.

A los buenos profesores de Tartagal, la Universidad Nacional de Tucumán y el Instituto Balseiro, por la calidad y el entusiasmo.

José

Contenido

Capítulo 1

Introducción

La técnica de *velocimetría por imágenes de partículas* (PIV) mediante plano laser ha demostrado ser un método potente para muy diversos problemas de mecánica de fluidos, en líquidos, gases e incluso en flujo multifase. Puede brindar medidas globales de un campo de velocidades con un nivel satisfactorio de precisión. Si bien requiere dominios transparentes o semitransparentes, su aplicación ha encontrado mucha aceptación en los laboratorios de fluidos. Es una herramienta apropiada para el estudio de turbulencias y por ello es aplicable a muy diversos problemas.

Una desventaja del método es la gran demanda computacional de los algoritmos de correlación cruzada que suelen emplearse para calcular los vectores de velocidad, hecho que impidió un surgimiento rápido de los sistemas en tiempo real pese a tener casi treinta años la técnica de PIV por plano laser. Las mediciones en tiempo real del campo de velocidades de un fluido resulta una facilidad experimental de gran interés, ya que abre la posibilidad de actuar sobre los parámetros del experimento fluidodinámico y simultáneamente obtener su respuesta, facilitando así la exploración de regímenes del problema e incluso la implementación de un lazo realimentado de control. Otra importante ventaja de un sistema en tiempo real es la posibilidad del ajuste de los parámetros propios de la técnica PIV como son, cantidad de partículas, magnificación y enfoque de la óptica asociada, etc. previo a la adquisición de los datos para un procesamiento *off-line*.

Nuevas tecnologías permiten implementar plataformas con altas velocidades de procesamiento gracias a la fácil paralelización de los algoritmos de PIV, alcanzándose mayores eficiencias a costos más bajos con respecto a otras soluciones como redes de computadoras (procesamiento paralelo en *clusters*).

En este trabajo analizamos la técnica de velocimetría PIV para conocer las distintas variantes del método y las solicitudes de su implementación. Consideramos las partes que constituyen un experimento de PIV por plano laser, evaluamos diferentes algoritmos de procesamiento, y ciertas tecnologías para su optimización en cuanto a velocidad de procesamiento y la captura de imágenes de video. Comentamos las características y consideraciones del flujo de diseño para la tecnología de los dispositivos de lógica programable, en particular los dispositivos FPGA (*Field Programmable Gate Array*).

Por último describimos el desarrollo de un sistema en tiempo real para PIV basado en tecnología FPGA. Detallamos los principales módulos programados en lenguaje descriptor de hardware (HDL) utilizados en la adquisición de video,

manejo de datos y procesamiento de vectores de velocidad. También mostramos la integración de esos componentes principales en un esquema de arquitectura de bus embebida en el dispositivo FPGA. Presentamos resultados de la medición de un campo físico de velocidades procesado con ésta plataforma.

Capítulo 2

Velocimetría por Imágenes de Partículas

En este capítulo introducimos en forma genérica la técnica de velocimetría por imágenes de partículas (PIV) basada en plano laser, para entender sus principios y como se implementa. La presentamos en contexto con otras técnicas de velocimetría muy difundidas en el ámbito experimental de la mecánica de los fluidos, mostrando sus particularidades que dejan resaltar el potencial de PIV. Señalamos en que sector de la amplia temática en PIV se concentra este trabajo.

Obtener automáticamente campos de velocidad a partir de imágenes es la función de los algoritmos PIV. Existen diversas estrategias de procesamiento que presentamos en este capítulo. Para poder estimar errores de una medición PIV cuantitativa, investigar simplificaciones de los algoritmos que faciliten su implementación y establecer limitaciones de su aplicación, describimos detalles de los mismos.

2.1 Velocimetría por Imágenes de Partículas

La *velocimetría por imágenes de partículas* (*PIV – Particle Image Velocimetry*) puede entenderse como la *medición cuantitativa de un campo de velocidades en un gran número de puntos* [5]. Los albores de PIV están en la mecánica del sólido, cuando se median campos de velocidades por la técnica de *laser speckle* (*LSV*) que en 1977 fue adoptada y evaluada en la mecánica de fluidos demostrando mediciones exitosas del perfil parabólico del flujo laminar en un tubo. Básicamente se median cambios en los patrones de interferencia de la luz que atravesaba una región de interés y se deducía un campo de velocidades que originaba. En 1984 se incorpora la idea de "sembrar" el fluido con partículas para ser iluminadas, bautizando la técnica como PIV. Originalmente se consideraba el patrón de interferencia generado con la luz dispersada en las partículas, pero pronto tomó relevancia trabajar directamente con las imágenes de las partículas producidas en tiempos muy cortos de iluminación. Midiendo el desplazamiento de las imágenes entre dos capturas de iluminaciones permite estimar los movimientos. La ideal herramienta compañera empezó a ser el laser con pulso doble y tiempo controlado entre pares de pulsos, por ser una fuente de luz direccional con emisiones muy breves pero intensas.

En corto tiempo muchos grupos de investigación en fluidodinámica adoptaron, mejoraron y diversificaron la técnica; en general motivados por resultar este tipo de velocimetría una herramienta apropiada para los problemas de turbulencia. En una medición por PIV se aporta información del campo de velocidades en varias escalas físicas, y en todas las direcciones, para un instante dado. Y el estudio de turbulencias presta mucha

atención a la fenomenología local y global en el dominio de estudio, considerando obviamente la aleatoriedad en la dirección de las velocidades.

Estas cualidades de PIV extienden su aplicabilidad a otros campos de investigación distintos del estudio de turbulencias. La literatura muestra experiencias exitosas en gases y líquidos, con números de Reynolds de todos los órdenes. En el Laboratorio Cavitación y Biotecnología (CABLCB-IB- CONICET), donde se desarrolló el presente trabajo, existe una línea activa de velocimetría PIV en problemas de hemodinámica y refrigeración de sistemas nucleares.

2.2 Clasificación de los sistemas de velocimetría y tipos de PIV

Algunos de los primeros métodos experimentales cuantitativos para medir velocidades en campos de fluidos se basan en *tubos Pitot*. Significó un avance la aparición de los *anemómetros de filamento por hilo caliente* en 1920, especialmente por su mejor respuesta en frecuencia, miniaturización del transductor, y la posibilidad de medir varias componentes de velocidad. Sin embargo ambas técnicas implicaban fuertes perturbaciones del medio, no despreciables en muchos problemas. El advenimiento del *laser* en los '60 permitió desarrollar la técnica del *anemómetro laser de efecto Doppler*. Si bien permitió medir velocidades con alta precisión y despreciable perturbación del medio, no dejaban de ser puntuales. La posibilidad de realizar mediciones globales de velocidad condujo rápidamente a las técnicas basadas en imágenes de partículas a ser muy ponderadas en el ámbito de la mecánica de fluidos.

La técnica PIV puede proveer medidas instantáneas de velocidad en dominios 2D o 3D con mediana precisión. Frente a las técnicas de visualización directa del flujo, que fueron aplicadas por décadas, puede considerarse a PIV una evolución hacia mayores capacidades cuantitativas y de automatización.

De acuerdo a una clasificación propuesta por Hinsh [3] un sistema de medición puede clasificarse según la terna *(k, l, m)*, donde k= 1, 2, 3 indica el número de componentes de velocidad que se miden, l =0, 1, 2, 3 señala la cantidad de dimensiones espaciales del dominio de medición, y m = 0, 1 indica si el registro temporal de la medición es instantáneo o continuo. Una cuarta componente podría indicar si diferencia entre gas, líquido y sólido respectivamente (generalizando a Hinsh). Según esto un sistema de medición puntual puede llegar a la categoría (3, 0, 1) en el mejor de los casos. Por otra parte el sistema PIV más difundido alcanza la clasificación (2, 2, 0) por brindar medidas de dos componentes de velocidad en un plano aunque en tiempos discretos, que también se define como PIV 2D-2C. La mayoría de los sistemas PIV actuales pertenecen a esta categoría [1].

La técnica PIV por holografía cinemática pertenece al caso (3, 3, 1). Más difundida, aunque también muy difícil y costosa, es la PIV holográfica simple que se cataloga por (3, 3, 0). Otros sistemas son el PIV estereoscópico [4] (3, 2, 1) y PIV traslativo [3] que provee información de velocidad 3D en dominios planos y se explicará más adelante.

2.3 Componentes y consideraciones experimentales en PIV

Hacer PIV demanda básicamente cuatro componentes (Figura 2.1): 1) Una región ópticamente transparente que contenga al fluido con las partículas a evaluar; 2) Una fuente de luz (*laser*) para iluminar la zona de interés (un plano o un volumen); 3) Sistema de

adquisición, transmisión y/o almacenamiento basado en una cámara CCD/CMOS, o filmadora, o placas holográficas; 4) Sistema de procesamiento de datos para extraer la información de velocidades.

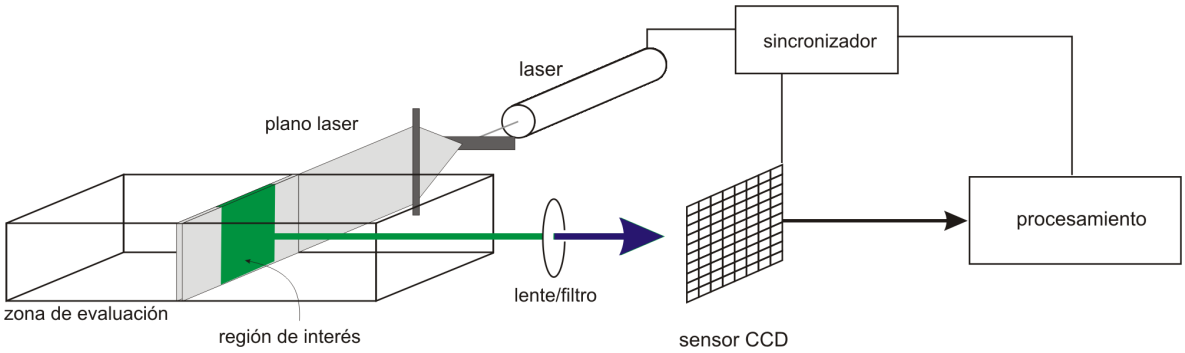


Figura 2.1 Componentes de un sistema PIV

Las variantes de cómo implementar PIV se analizan evaluando estos ítems, según necesidades o posibilidades. Por ejemplo, el laser puede ser pulsado o continuo, las plataformas de cómputo pueden ser muy distintas tanto como el algoritmo que en ellas se ejecuta. En este trabajo en particular se desarrolló un sistema para efectuar PIV cuantitativo en tiempo real. La descripción del sistema se realiza en el Capítulo 4.

2.3.1 Óptica

En muchos contextos experimentales puede ser importante conocer algún detalle de la formación de imágenes de las partículas en el sensor (plano de imagen). Una relación geométrica fundamental de óptica se da entre la distancia del objeto (x_0) y la distancia de la imagen (x_i) al plano de la lente mostrada en las Figura 2.1 y Figura 2.2:

$$\begin{aligned} x_0 &= (1+M^{-1})f, \\ x_i &= (1+M)f, \\ M &= x_i / x_o, \end{aligned} \tag{2.1}$$

donde M es el aumento de la imagen y f es la distancia focal de la lente.

Dos efectos deben considerarse para estimar el diámetro de la imagen de las partículas en el plano de imagen: el geométrico y el de difracción. Si d_p es el diámetro real de las partículas, Md_p debería ser el diámetro en la imagen (d_i) considerando el efecto geométrico solamente. Sin embargo la difracción es muy importante en muchos casos prácticos de captura de imágenes. La imagen de un punto en el plano de objetos se corresponde a una función de Airy en el plano de imagen, y puede considerarse al diámetro d_a de dicho disco de Airy la medida mínima que puede tener una partícula en la imagen. La función que calcula este tamaño viene dada por:

$$d_a = 2.44(1+M)\frac{f}{D}\lambda, \tag{2.2}$$

siendo λ la longitud de onda del laser y D la apertura de la lente. Combinando este valor con el tamaño geométrico se obtiene un el diámetro efectivo de la imagen de una partícula de la siguiente manera:

$$d_{ef} = \sqrt{(M^2 d_p{}^2 + d_a{}^2)} \quad . \tag{2.3}$$

El espesor δz de la región donde las partículas se captan con un enfoque aceptable se llama *profundidad de campo*. La relación óptica de la ec.(2.4) la tiene en cuenta. En general se espera que la profundidad de campo sea mayor que el espesor del plano laser, evitando tener imágenes de partículas fuera de foco. En [6] se presenta la descripción de la técnica de volumen láser, en la cual se pretende que ocurra lo contrario; a través del desenfoque de las imágenes de las partículas es posible medir un desplazamiento 3D.

$$\delta z = 4(1 + M^{-1})^2 (f / D)^2 \lambda \quad . \tag{2.4}$$

El plano láser puede generarse con una simple combinación de lentes (Figura 2.3). Con una lente cilíndrica se consigue expandir el haz laser en una sola dirección y con una lente esférica se reorientan los rayos más colimados. En particular, a una distancia focal de la lente esférica se espera una mayor concentración o enfoque del plano laser, en la llamada *cintura del plano* que define su espesor (e_p). Si se logra una buena colimación (altura $H \approx cte$) con la lente esférica, la distribución de la intensidad de luz en el plano será uniforme a distintas distancias x. La intensidad local de la luz depende directamente de la potencia del laser (Figura 2.4) e inversamente de dicha la altura H y del espesor e_p. El *perfil gaussiano* del haz laser original se conserva en las direcciones de la altura y del espesor del plano.

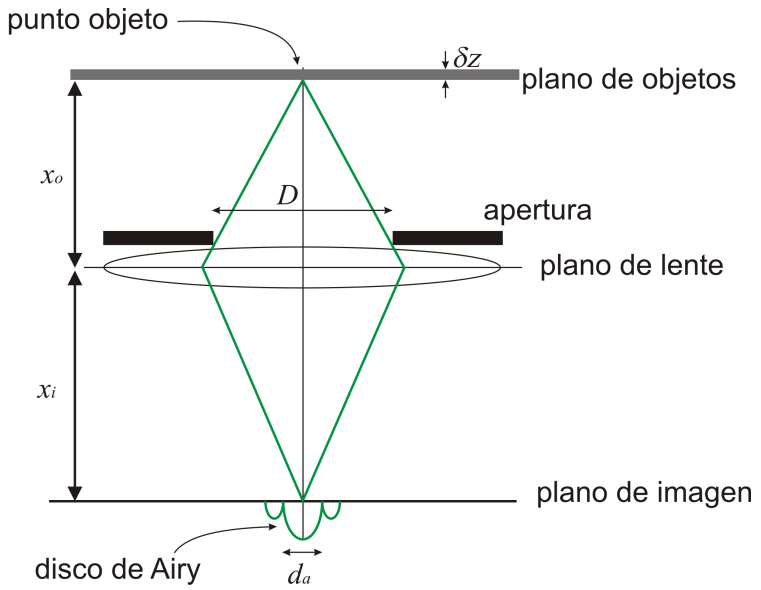


Figura 2.2 Efecto de difracción en la formación de imágenes de las partículas

A modo de síntesis de todo lo anterior cito el ejemplo de [1]. En cuanto a la captura de imágenes, si se considera una lente con M = 0.2 y f/D = 8, con un laser de λ = 532 nm

se obtiene d_a = 12.5 μm. Para partículas típicas de d_p = 8 μm se obtiene una imagen con d_{ef} = 12.6 μm y predominancia del efecto de difracción (Md_p = 1.6 μm). Con este sistema la profundidad de campo es 4.9 mm. En la generación del plano laser, para una región de interés en el fluido de 10 cm x 10 cm puede emplearse una lente cilíndrica de distancia focal -20 mm con una lente esférica de 1000 mm de distancia focal resultando un espesor e_p = 1 mm.

2.3.2 Partículas

Las partículas de seguimiento para PIV tienen que satisfacer dos requerimientos: 1) Deben poder seguir las líneas de corriente del flujo sin excesivo resbalamiento (*slip*), y 2) deben ser buenas dispersoras de la luz laser seleccionada o fluorescer. Esta segunda condición impacta mucho en el sistema de iluminación a elegir y la forma de capturar las imágenes [1]. Partículas con poca capacidad de dispersión demandarán más potencia del laser o cámaras más sensibles o mayores concentraciones que pueden modificar la dinámica del problema; en consecuencia mayores costos y requisitos de seguridad.

Para evaluar el primer requerimiento de bajo resbalamiento un modo muy simple de hacerlo es determinando la velocidad terminal u_∞ de la partícula bajo gravedad. Suponiendo que el proceso esta gobernado por el *arrastre de Stokes*, esa velocidad u_∞ viene dada por:

$$u_\infty = \frac{g\ d_p(\rho_p - \rho_f)}{18\mu}, \tag{2.5}$$

donde d_p y ρ_p son el diámetro y densidad de la partícula respectivamente, y μ junto a ρ_f son respectivamente la viscosidad y densidad del fluido. Las partículas son apropiadas cuando u_∞ es despreciable comparado a las velocidades del flujo real.

Para bajas velocidades es preferible usar partículas con densidad próxima a la del fluido. En el caso del agua son buenos materiales candidatos el poliestireno u otro plástico. Siempre debe hacerse valer que d_p sea lo suficientemente pequeña para no alterar el flujo. Desafortunadamente, la mayoría de los materiales con gravedad específica $\approx$ 1 también poseen un índice de refracción similar al del agua y por ende su capacidad de difractar la luz del laser es escasa. Pueden conseguirse partículas plateadas o recubiertas con material fluorescente como rodamina.

Para flujos gaseosos es común encontrar el uso de gotas de aceite [1]. La baja densidad de los gases obliga a concluir de la ec. (2.5) que las partículas deben ser muy pequeñas (típicamente $<$ 1 μm). Sin embargo su centrifugación en vórtices es una conocida dificultad en el método. Una posible manera de generar gotas tan pequeñas consiste en burbujear aceite con aire y filtrar con un manojo de pequeños tubos las partículas desprendidas al explotar las burbujas (se conoce como *tobera de Laskin*).

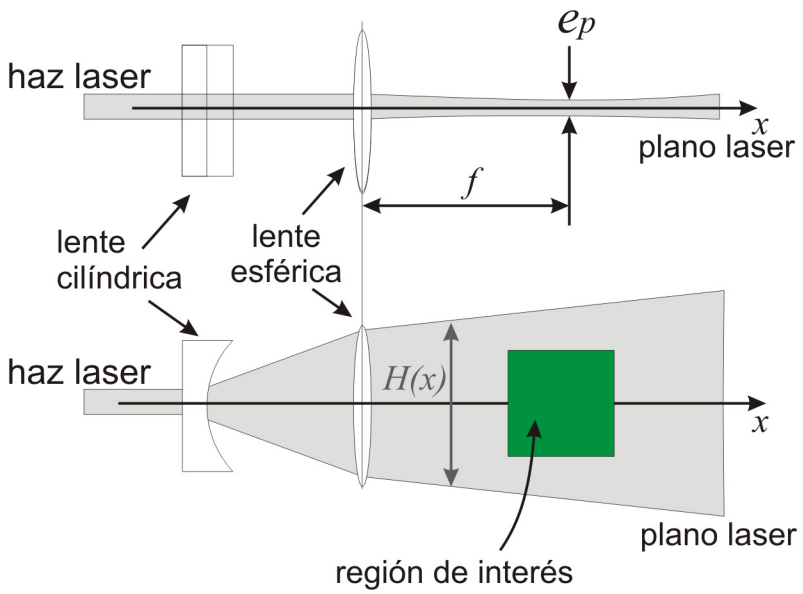


Figura 2.3 Generación del plano laser

2.3.3 Iluminación por laser

Láseres continuos y pulsador han sido aplicados en PIV, siendo estos últimos los más utilizados. Entre los láseres continuos es común encontrar los de Helio-Neón y en mayor medida los de *iones de argón* (por ser más potentes principalmente) con potencias de algunos vatios. Entre los láseres pulsados son muy utilizados los de Nd+: YAG, que pueden producir pulsos de unos 100 mJ (con potencias de megavatios en cada pulso) con repeticiones de a decenas de Hz.

A partir de los láseres continuos pueden crearse pulsos de luz haciendo cortes al haz (*chopped beam*) o empleando un arreglo de espejos que deriven la luz al rotar (produciendo a la vez un plano laser por barrido).

Sin embargo, el gran atractivo está en los láseres pulsados por su corta duración de luz (δt de algunos nano-segundos). En términos prácticos una partícula incluso de muy altas velocidades aparecerá "congelada" en la imagen, con niveles altos de intensidad. Los láseres continuos imponen restricciones a la velocidad máxima del fluido. Para conseguir dos pulsos muy cercanos en tiempo (Δt del orden de microsegundos a milisegundos) se necesitan dos láseres, que en general permiten separar temporalmente dichos pulsos de manera arbitraria (Figura 2.4). Una cuestión a resaltar es la alineación de los dos planos laser, ya que existe un nivel de solapamiento espacial ($\approx 50\%$) por debajo del cual los resultados son pobres.

Los láseres pueden sincronizarse con una cámara de video y obtener en un cuadro la imagen del primer pulso y en el siguiente la del segundo pulso con lo que la cámara no necesita ser rápida. La información de ambos cuadros se correlaciona para obtener los desplazamientos.

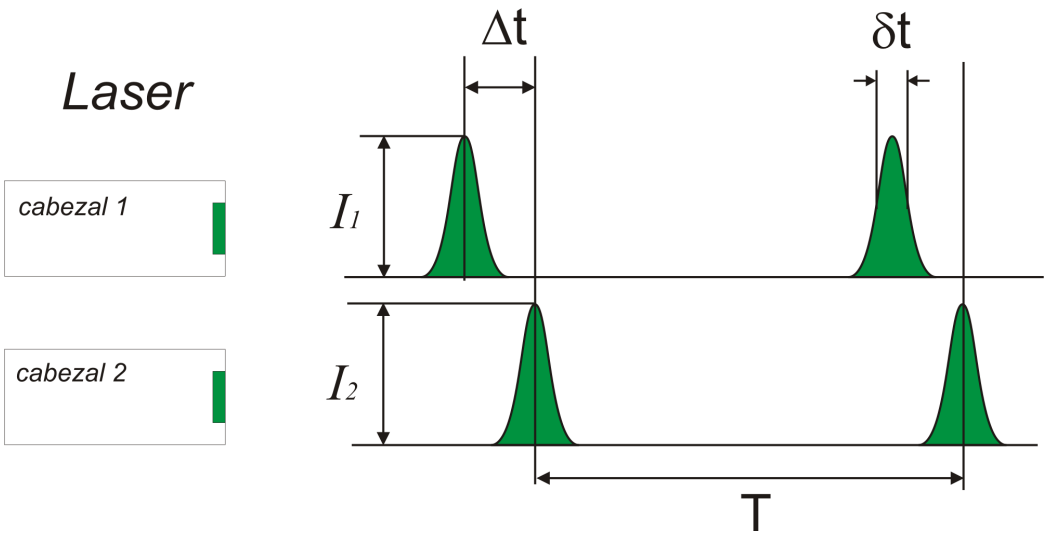


Figura 2.4 Pulsos de laser. Valores típicos T = 0.02 a 0.1 s, Δt = micro-segundos a milisegundos, δt = varios nano-segundos.

2.3.4 Electrónica de captura, almacenamiento y procesamiento

Las imágenes de PIV pueden capturarse en una filmación o con un sensor CCD (*charged-coupled device*) o CMOS (*Complementary Metal Oxide Semiconductor*), mientras que otras aplicaciones en general usan placas o películas fotográficas de alta resolución. El volumen de datos de cada método define en gran medida la estrategia de procesamiento. Una cámara CCD produce en general cuadros de 1 Mbyte, y las películas fotográficas entre 80 Mbytes hasta 1 Gbytes. Aprovechar toda la información de éstas últimas a veces resulta engorroso.

La captura digital usando CCD presenta enormes facilidades experimentales frente a los otros procesos donde la recuperación de las imágenes posee muchos pasos. La captura directa de la información acerca muchas opciones para el procesamiento, como tener una PC que almacene y luego procese, o tener un conjunto de computadoras en red que realicen procesos en paralelo, o implementar electrónica con la función específica de acelerar los cálculos (los procesamientos para velocimetría de plano laser son fácilmente paralelizables). Este trabajo trata acerca de un diseño para realizar procesamiento PIV en tiempo real o incluso adaptable para procesar información guardada en un dispositivo de almacenamiento.

Algunos diseños de cámaras permiten capturar varias imágenes completas y sucesivas en tiempo muy cortos (microsegundos), realizando almacenamientos intermedios (*buffering*) y posterior transferencia al sistema final de almacenamiento y procesamiento. Esto resulta ideal para hacer PIV doble-cuadro de alta velocidad, donde la captura se realiza para los dos pulsos de luz cercanos distante Δt y la transmisión durante el periodo T entre pulsos de cada laser.

La elección más popular para PIV es conectar la cámara a una computadora personal mediante placas de captura (*frame grabbers*). Pero incluso los sistemas más modernos presentan dificultades para implementar en tiempo real la visualización del campo vectorial resuelto sin sacrificar resolución. El manejo eficiente de memoria, de grandes caudales de información y poseer capacidad de cálculo y una robusta arquitectura gráfica, son las solicitudes principales [8].

La próxima sección introduce los algoritmos de procesamiento donde queda en evidencia la importante labor computacional necesaria en PIV. El Capítulo 3 se concentra en información sobre las tecnologías tenidas en cuenta para este trabajo, en su mayoría aplicadas a la versión desarrollada basada en tecnología de lógica programable FPGA. En este trabajo de velocimetría en tiempo real se pensó como una plataforma versátil de procesamiento desde su concepción, por lo tanto la recopilación hecha en dichos capítulos sirve como punto de partida para futuras adaptaciones y expansiones de esta primera versión que captura y procesa directamente la información registrada por la cámara.

2.4 Algoritmos PIV

2.4.1 PIV por seguimiento de partículas

El primer sistema desarrollado para el procesamiento automático de PIV [1] consistía en iluminar dos veces las partículas y capturar dentro de un mismo cuadro (o fotograma) las dos imágenes generadas, y sobre ese patrón *asociar una posición inicial y final a cada partícula*. De allí el nombre *velocimetría por seguimiento de partículas* (*particle tracking velocimetry* o PTV). La distancia entre las imágenes (*desplazamiento lagrangiano*) se toma para estimar la *velocidad euleriana* local de la partícula en el plano. El lector se imaginará la dificultad lógica de este algoritmo. En primer lugar deben asociarse coordenadas a cada imagen de partícula, separarlas del ruido, y posteriormente asociar de a pares lo puntos que corresponden a la misma partícula. Dicho esquema requiere un código bastante sofisticado [10].

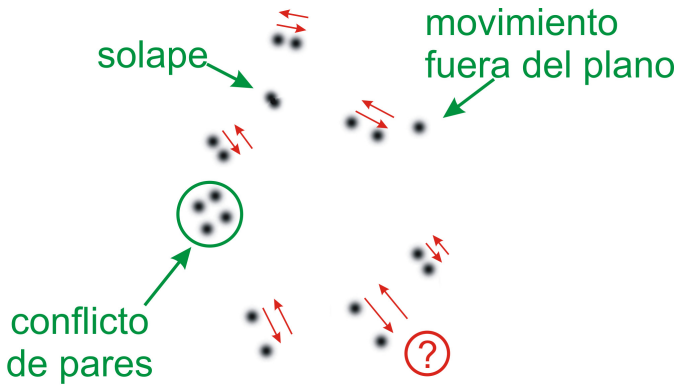


Figura 2.5 Particularidades del algoritmo de seguimiento de partículas (PTV)

En PTV en general se eliminan los valores de señal por debajo de cierto umbral (*thresholding*) para rechazar buena parte del ruido, un procesamiento para mejorar la relación señal-ruido (*SNR*). Posteriormente se buscan los centros de los puntos brillantes.

En PTV muchas dificultades aparecen por solapamiento o fragmentación de las imágenes, y también por pérdida de uno de los puntos asociados al par (Figura 2.5). Esto último es de esperar en los casos que la partícula este fuera del plano laser en una de las dos capturas, efecto denominado *movimiento fuera del plano*. En general PTV funciona mejor cuando las concentraciones de partículas son bajas, i.e. cuando la distancia entre partículas es mucho mayor a la distancia del desplazamiento entre capturas. Pero esto puede llevar a obtener bajas resoluciones espaciales y temporales.

Los resultados en PTV siempre se presentan en los lugares donde se encontraron pares de imágenes procesables, o sea de modo no-uniforme en el plano de medición. Esto es una dificultad adicional, ya que se necesita aplicar interpolaciones para conseguir medidas de velocidad en una grilla que facilite otras operaciones posteriores.

Pese a caer en desuso, PTV fue muy útil en los comienzos de la velocimetría PIV. Aunque tiene grandes dificultades, su gran complejidad de procesamiento es lógica y no computacional. En 1985, como bien describe Adrians a su computadora DEC PDP 11/23 en [1], el poder de almacenamiento y cómputo era muy bajo respecto al actual en la mayoría de los laboratorios de fluidos y PTV era la solución más práctica entonces.

2.4.2 Algoritmos basados en correlación

Usando una estrategia de correlación se reduce la complejidad lógica del algoritmo PIV respecto al seguimiento individual de partículas, en particular por no existir la necesidad de asociar entre si pares de imágenes de partículas ni buscar coordenadas para cada una. En lugar de hacer un cómputo individual, la técnica de correlación brinda una medida promedio, o estadística, del movimiento de un conjunto de partículas dentro de pequeñas regiones conocidas como *ventanas de interrogación*. En el diagrama del Anexo A se muestra resumidamente las definiciones más importantes que continuamente usaremos al hablar de PIV basado en correlación.

Esencialmente, todo el cuadro (o fotograma) es dividido en ventanas de interrogación y para cada una de ellas se asociará al final del proceso un vector velocidad. De esa manera se genera una grilla uniforme de resultados. El proceso de promediar movimientos le confiere al método de correlación mayor tolerancia al ruido en las imágenes y mejor robustez que el método de seguimiento PTV.

El cómputo del desplazamiento medio puede realizarse utilizando una *auto-correlación* (una imagen con dos exposiciones) o preferentemente una *correlación-cruzada* (2 imágenes con una exposición cada una). Estas operaciones constituyen temas importantes en el campo del procesamiento de señales [13]. Comúnmente es asociada la auto-correlación de una señal a la extracción de información sobre la estructura de si misma y su comportamiento en el dominio que la contiene (tiempo, espacio, etc.); es útil para encontrar periodicidades ocultas en una señal. La función de correlación-cruzada brinda una medida de las similitudes o propiedades compartidas entre dos señales; se utiliza en análisis espectrales, detección y reestablecimiento de señales inmersas en ruido y suele aplicarse en la detección de señales de retorno de un radar, el acople entre patrones, en la medición de retardos temporales o en el problema que nos interesa: corrimientos en un dominio espacial de dos dimensiones.

Más adelante se tratan resumidamente los errores que el promediado de correlación introduce cuando existen gradientes elevados de velocidad, y otras consideraciones físicas como la posible alteración de la dinámica por grandes concentraciones de partículas.

2.4.3 Auto-correlación

La auto-correlación se aplica cuando las dos imágenes consecutivas de las partículas se graban en un mismo fotograma, como el caso usado para describir el método por seguimiento PTV. Esto puede generarse dejando el obturador de la cámara abierto durante dos pulsos laser consecutivos, que es un modo de grabación llamado *simple-*

fotograma/doble-pulso [1]. La función de auto-correlación *C(x,y)* de un patrón de intensidades *I(i,j)* (el fotograma) se define en la ventana de interrogación (*vi*) como

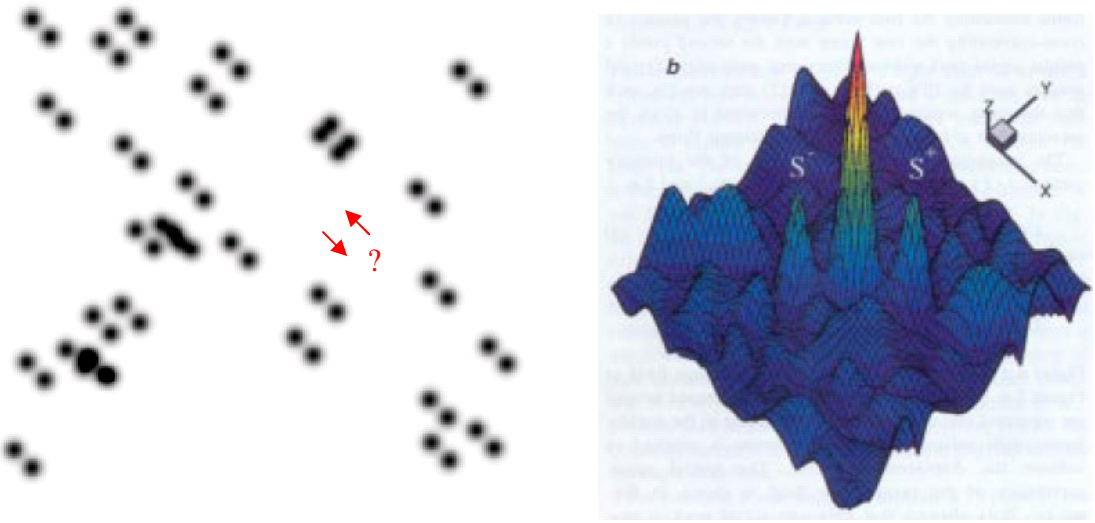
$$C(x,y) = \iint_{vi} I(i,j)I(i+x, j+y)didj\,. \tag{2.6}$$

La integral de área se convierte en una doble sumatoria discreta cuando la definición se traslada al dominio digital. Esta función tiene la posibilidad de calcularse para la imagen digital mediante una *Transformada Rápida de Fourier bidimensional* (*FFT-2D*) si resulta de las dimensiones apropiadas.
Se requiere que sea cuadrada y sus lados una potencia de 2. Denotamos la operación de correlación mediante el símbolo $\otimes$ y la operación de convolución por $*$, que por definición [3] se relacionan de la siguiente manera:

$$C(x,y) = I(i,j) \otimes I(i,j) = I(i,j) * I(-i,-j)\,. \tag{2.7}$$

Denotando *F[ ]* a la Transformada de Fourier, por el teorema de convolución:

$$\begin{aligned} &F\left[C(x,y)\right] = F\left[I(i,j)\right]\ \ F^{*}\left[I(i,j)\right] = \left|F\left[I(i,j)\right]\right|^{2}, \\ &\Rightarrow C(x,y) = F^{-1}\left[\left|F\left[I(i,j)\right]\right|^{2}\right]. \end{aligned} \tag{2.8}$$



a) b)

Figura 2.6 a) Imágenes de partículas simuladas para una captura simple-cuadro/doble pulso. El desplazamiento es uniforme, y permite visualizar características de los resultados esperables usando el algoritmo de auto-correlación. b) Resultado típico de un mapa de auto-correlación

En un mapa de auto-correlación típicamente aparecen tres picos destacados. Uno en la posición central llamado pico de auto-correlación y corresponde claramente a correlacionar un desplazamiento nulo entre los patrones de ese único cuadro. En sus

vecindades suelen aparecer dos picos menores de alturas muy parecidas cuyas posiciones equidistan del pico central y reflejan información del real desplazamiento de las partículas (Figura 2.6). La generación de estos dos picos es evidente por la simetría del problema de auto-correlación, y son fuente de una indefinición en el sentido del movimiento, aunque si queda de manifiesto la magnitud y dirección. Esto es una desventaja que puede salvarse si se conoce el sentido preferencial del flujo en el problema o aplicándose estrategias experimentales como describe Prasad en [10].

Adicional a este problema de indefinición en la auto-correlación hay que destacar la imposibilidad de medir desplazamientos nulos o muy pequeños por la presencia del pico de auto-correlación que en general tiene el diámetro de la imagen de una partícula.

En general se evita hacer Plano Laser por auto-correlación en preferencia de la técnica de correlación cruzada.

2.4.4 Correlación cruzada

A partir de una captura del tipo doble cuadro y simple pulso por cuadro es posible utilizar un algoritmo de correlación cruzada. La idea consiste en capturar de la cámara dos cuadros con imágenes de un pulso asociada a cada uno y cada vez en coordinación con un pulso de laser. Así cada cuadro retiene la imagen iluminada por un solo pulso laser. La operación requiere sincronismos entre el laser y la cámara.

La identidad temporal ocasionada por haber dos cuadros secuenciales, con un pulso laser asociado a cada uno, permite que al correlacionar cruzadamente el primero con el segundo se obtenga un único pico máximo sin ambigüedades en la dirección del movimiento de las partículas como ocurre en la auto-correlación.

La correlación cruzada puede realizarse de un modo directo o de un modo normalizado, y siempre se toma la ubicación del valor máximo de correlación entre desplazamientos candidatos como medida del movimiento real. El caso normalizado posee mayor complejidad computacional, sin embargo es más robusto ante situaciones donde la distribución espacial de las partículas no es homogénea, las partículas no son monodispersas, y la intensidad de los pulsos laser sucesivos es muy diferente. A continuación se describen matemáticamente ambos casos.

Correlación Cruzada Directa

La función de *correlación cruzada directa* entre dos patrones de intensidad *A(i,j)* y *B(i,j)* definidos en la misma ventana de interrogación (pero originados en distintos instantes de tiempo separados por Δt) se define como

$$R_{AB}(x,y) = \iint_{vi} A(i,j)B(i+x,j+y)didj\,. \tag{2.9}$$

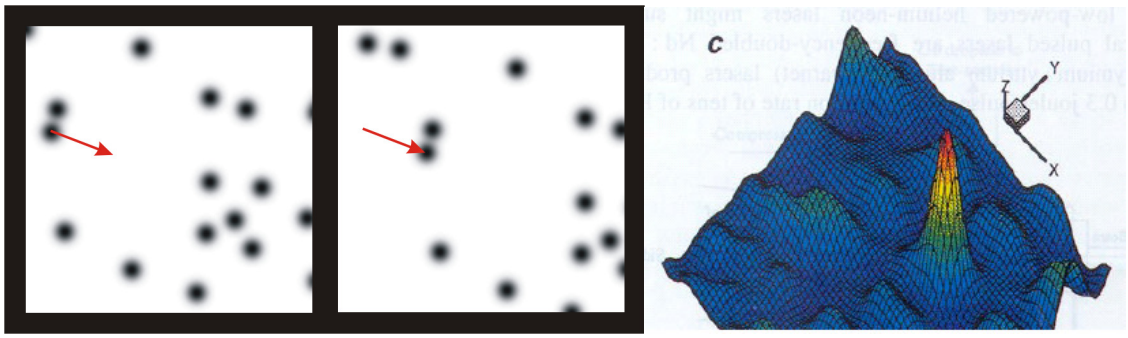
Y en forma discreta como:

$$R_{AB}(x,y) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} A(i,j)\;\; B(i+x,j+y)\,. \tag{2.10}$$

Usar la Transformada de Fourier requiere que los patrones sean cuadrados. Se muestra que en esos casos vale:

$$R_{AB}(x,y) = F^{-1}\left[F\left[A(i,j)\right]\ F^{*}\left[B(i,j)\right]\right]. \tag{2.11}$$

Este proceso no deja ambigüedades en la dirección del movimiento y permite detectar incluso desplazamientos nulos por no estar presente el pico de auto-correlación (Figura 2.7). Un tercer beneficio frente a la auto-correlación es la mayor tolerancia a ruidos en la imagen. Por estas razones es preferente hacer PIV con el método de correlación cruzada y casi nadie hace auto-correlación.



a) b)

Figura 2.7 a) Imágenes de partículas simuladas mostrando el vector de desplazamiento. b) Mapa de correlación típico al resolver la correlación cruzada. La presencia de un solo pico no da lugar a ambigüedades y permite medir corrimientos nulos.

Introducimos un factor de mérito f_O a la dificultad de implementación que posee la versión espacial (*cc-d-espacial*) frente a la versión basada en la Transformada de Fourier (*cc-d-Fourier*) del algoritmo anterior. Ese factor de mérito, en una implementación de *hardware,* puede definirse por el número total de multiplicaciones necesarias para calcular una parte representativa del algoritmo. En nuestro caso adoptamos las correlaciones necesarias para evaluar una ventana de interrogación. En el modo espacial por cada par *(x,y)* se realizan n^2 multiplicaciones. Además, dentro de una ventana se evalúa un dominio de z posibles corrimientos de muestreo (cada uno definido por x, y). En el caso frecuencial uno considera las dos transformadas rápidas de Fourier (una por patrón), una multiplicación de las matrices resultantes y finalmente una transformada rápida inversa de Fourier. Las siguientes relaciones permiten calcular el factor de mérito, siendo k la potencia entera de dos igual o inmediatamente mayor a n [4]. Se tiene en cuenta que por cada multiplicación compleja se necesitan cuatro multiplicaciones reales.

$$\begin{aligned} f_O(cc-d-espacial) &= f_e = z\ n^2, \\ f_O(cc-d-Fourier) &= f_F = 4\left(3\left(2k^2\ \log_2 k\right)+k^2\right). \end{aligned} \tag{2.12}$$

En el Capítulo 4 los patrones de intensidad A y B se definen de manera más exhaustiva introduciéndose claramente las definiciones de *áreas de interrogación* y

corrimientos de muestreo (ver Anexo A). De esa manera, para z = 81, n = 32 y k = 64, resulta

$$\frac{f_F}{f_e} \approx 4$$

de donde se obtiene 82944 multiplicaciones para el caso espacial que resulta favorecido y es el implementado en el diseño del presente trabajo.

Correlación Cruzada Normalizada

Se comentó que la ec. (2.9) resulta sensible a los cambios en la intensidad entre los patrones *A* y *B*, entre otras cosas, producidos por no ser de idéntica potencia ambos pulsos laser (Figura 2.4). Este problema puede resolverse usando la correlación cruzada normalizada:

$$R_{AB}(x,y) = \frac{\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\left[A(i,j)-\bar{A}\right]\left[B(i+x,j+y)-\bar{B}(x,y)\right]}{\sqrt{\left[\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\left[A(i,j)-\bar{A}\right]^2\right]\left[\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\left[B(i+x,j+y)-\bar{B}(x,y)\right]^2\right]}}, \qquad (2.13)$$

donde $\bar{A}$ y $\bar{B}$ son los valores medios de *A(i,j)* y *B(i,j)* [5]. El manejo de esta expresión mediante la transformada de Fourier resulta muy elaborado, y por ello suele usarse en el dominio espacial únicamente.

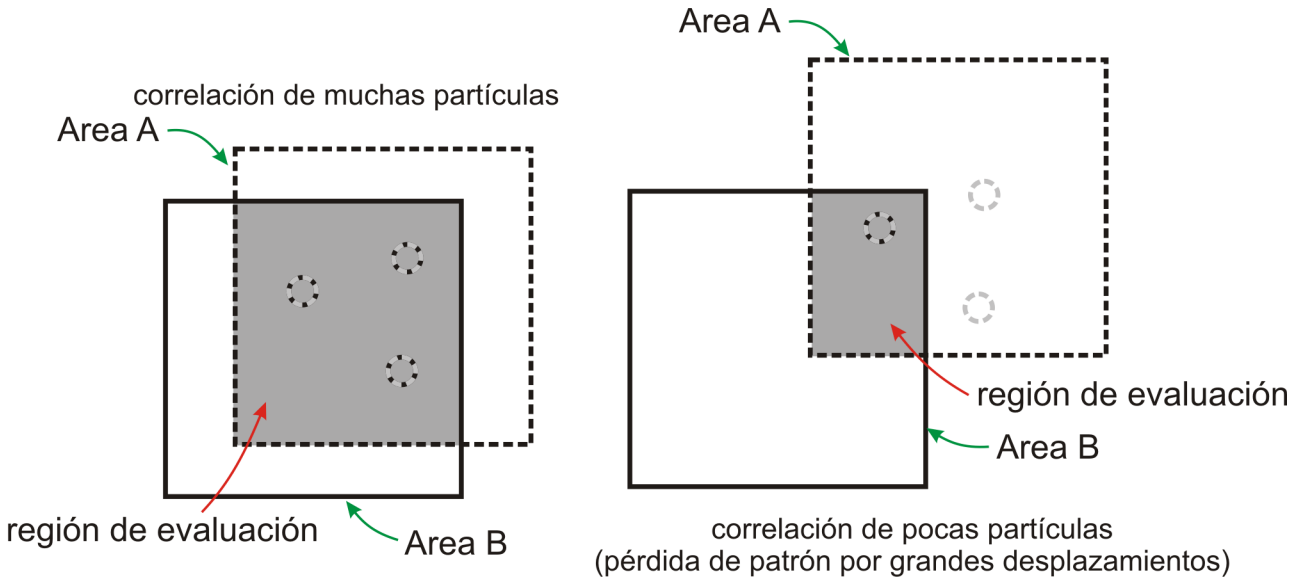


Figura 2.8 Efecto de pérdida de patrón. Se genera por grandes desplazamientos de partículas. Esto no permite obtener picos destacados entre el ruido de correlación

2.4.5 Efecto de bordes y de pérdida de patrón

Un problema común en la técnica PIV es la pérdida de patrón. Si los dos patrones de intensidad que buscamos correlacionar son de igual tamaño, al evaluar desplazamientos cada vez mayores se corre el riesgo de encontrar valores pequeños de correlación por ser alto el número de partículas que salieron de la región evaluada. A ese efecto se lo denomina pérdida de patrón. La Figura 2.8 esquematiza tal problema.

Por tener las Areas de correlación tamaños acotados, se presenta un efecto de borde debido al cálculo numérico finito; y en general solo tiene sentido evaluar corrimientos pequeños. Una solución consiste en realizar las correlaciones con patrones de intensidad *A* y *B* de diferentes tamaño (Figura 2.9) en una dada ventana de interrogación (ver Anexo A), algo denominado *correspondencia de patrón de imágenes* (*Particle Image Pattern Matching - PIPM*) [5]. Una apropiada elección de los tamaños debe respetar el desplazamiento máximo en cada dirección que se espera de las imágenes (*D* [píxel]) según

$$n - m > D,$$

donde suponemos cuadradas las *subAreaA* y Area*B*, de ancho *n* [píxeles] y *m* [píxeles] respectivamente.

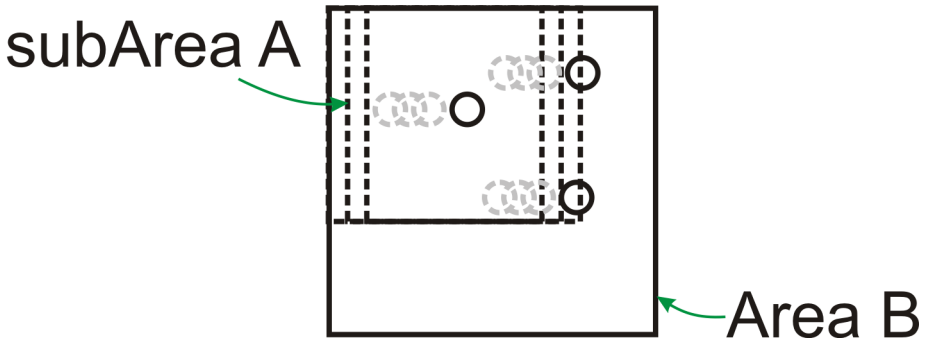


Figura 2.9 Correspondencia de patrón. Se busca el mejor ajuste de un área móvil (se define en Anexo A como subArea A) dentro de un área fija mayor (definida como Area B). Permite obtener resultados equivalentes al caso de anterior pero simplificando el cálculo por no evaluarse los casos con posible pérdida de patrón.

En cualquier caso es importante estimar la componente de velocidad real máxima de las partículas en el plano laser admitida por el sistema a diseñar, usando la magnificación del sistema de captura de video, tiempo Δt entre capturas de imágenes y el tamaño máximo de desplazamiento a evaluar la ventana de interrogación (supongamos *D*). Entonces, la componente de velocidad real máxima que puede medirse en todas direcciones es:

$$v_{\max} = \frac{D}{M}\Delta t^{-1}, \quad [D] = píxeles,\ [M] = píxeles/m. \qquad (2.14)$$

En el caso de la dirección diagonal en la ventana de interrogación dicha velocidad máxima es:

$$v_{diag-\max} = \sqrt{2}\ v_{\max}\ .$$

2.5 Simplificación de Algoritmos

Para la implementación en tiempo real de la técnica PIV, una de las decisiones más relevantes es la elección del algoritmo para el procesamiento eficiente de las imágenes [7]. En general, mejorar la eficiencia implica reducir el costo computacional, la utilización de memoria y eventualmente el consumo de energía. La gran cantidad de datos que se involucran en un procesamiento de imágenes hace que no sea inmediata la categorización de la eficiencia de un determinado algoritmo. la definición de que algoritmo puede considerarse eficiente. Si bien muchas optimizaciones a nivel hardware y software pueden realizarse para objetivos de tiempo real, se reconoce que en general las simplificaciones al nivel del algoritmo consiguen los mayores beneficios [8].

Repasando la literatura sobre procesamiento en tiempo-real de imágenes y video tres grandes conceptos suelen destacarse en la simplificación de algoritmos:

- reducción del número de operaciones;
- reducción de la cantidad de datos a procesar; y
- utilización de algoritmos simples o aproximados.

En el problema de PIV, se dijo que un algoritmo muy robusto, incluso aplicable a problemas de velocimetría o seguimiento de objetos más genéricos que PIV, es el de correlación cruzada normalizada, ec. (2.13). Tomamos este caso por lo enriquecedor y potente que finalmente resulta. Se busca tener en cuenta las particularidades de nuestro problema PIV, de la implementación de funciones matemáticas en hardware (ver Capítulo 6) y algún criterio de cálculo aplicado al procesamiento de señales.

La primera consideración es una aproximación de la ec.(2.13), que fue aplicada con éxito en un seguidor infrarrojo de objetos [9], y consiste en aproximar el denominador como se muestra en la ec.(2.15). La idea parte del desarrollo multidimensional en series de Taylor, de donde se obtiene:

$$f(\vec{x}) = \sqrt{\sum_i x_i^{\,2}} \approx \sum_i \left| x_i \right| .$$

$$R_{AB}(x,y) \approx \frac{\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\left[A(i,j)-\overline{A}\right]\left[B(i+x,j+y)-\overline{B}(x,y)\right]}{\left(\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\left|A(i,j)-\overline{A}\right|\right)\left(\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\left|B(i+x,j+y)-\overline{B}(x,y)\right|\right)}, \qquad (2.15)$$

Otra consideración consiste en despreciar el efecto de restar los valores medios de los patrones en cada término de (2.15). Esto quita robustez ante relaciones bajas de señal a ruido (*SNR*), pero es algo admisible si se tiene control sobre las condiciones. Por último

señalamos la posibilidad de simplificar aún más el denominador en (2.15). Si tenemos en cuenta que la búsqueda del pico de correlación máxima consiste en un algoritmo de comparación relativa entre los resultados de todas las correlaciones dentro de una ventana, es razonable extraer el factor integral del patrón *A* que se repite como un coeficiente de proporcionalidad en todos los cálculos de correlación R_{AB}. Esta idea se aplica satisfactoriamente en [9]. Finalmente, resumimos el análisis anterior en la ec. (2.16).

$$R_{AB}(x,y) \tilde{\propto} \frac{\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} A(i,j)\ B(i+x,j+y)}{\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} B(i+x,j+y)}, \tag{2.16}$$

2.6 Interpolación sub-píxel

En PIV, la ubicación del pico de correlación esta directamente asociada con el desplazamiento de las partículas. La naturaleza discreta de la captura de la imagen y su procesamiento digital introducen un error de ±0.5 píxeles. En las correlaciones expresadas por las ecs. (2.10) y (2.13) un máximo solo puede encontrarse en posiciones discretas. Para incrementar la precisión del método, o determinar la posición de los máximos con una precisión menor a un píxel (*precisión sub-píxel*), existen diversos métodos [6] como *centro de masas*, *ajuste parabólico* y *ajuste gaussiano*. La idea es que la ubicación del pico de correlación sea una función de la forma del pico de correlación. En general el ajuste gaussiano (2.18) es el más empleado en PIV digital, aunque el ajuste parabólico (2.17) resulta más atractivo para implementar en hardware. En las ecs. siguientes se calcula la posición corregida del pico de correlación a partir de la posición entera del pico (x,y) y valores de las correlaciones en las cercanías.

$$\begin{aligned} p_x &= x + \frac{R_{AB}(x-1,y) - R_{AB}(x+1,y)}{2R_{AB}(x-1,y) - 4R_{AB}(x,y) + 2R_{AB}(x+1,y)} \\ p_y &= y + \frac{R_{AB}(x,y-1) - R_{AB}(x,y+1)}{2R_{AB}(x,y-1) - 4R_{AB}(x,y) + 2R_{AB}(x,y+1)} \end{aligned} \quad . \tag{2.17}$$

$$\begin{aligned} p_x &= x + \frac{\ln R_{AB}(x-1,y) - \ln R_{AB}(x+1,y)}{2\ln R_{AB}(x-1,y) - 4\ln R_{AB}(x,y) + 2\ln R_{AB}(x+1,y)} \\ p_y &= y + \frac{\ln R_{AB}(x,y-1) - \ln R_{AB}(x,y+1)}{2\ln R_{AB}(x,y-1) - 4\ln R_{AB}(x,y) + 2\ln R_{AB}(x,y+1)} \end{aligned} \quad . \tag{2.18}$$

En [5] se presenta un análisis mediante imágenes simuladas del error en PIV donde se suponen los picos con forma gaussiana.

2.7 Estrategia para velocimetría 3D: PIV traslativo

Lo descrito anteriormente es la base del cálculo por PIV de las componentes de un campo de velocidades proyectadas en el plano de visualización, que siempre se supuso igual al plano laser. Si bien la información que aporta esta modalidad 2D-2C de PIV (un dominio bidimensional, el plano de visualización, y medición de dos componentes de velocidad) es muy útil, la medición de la tercera componente de velocidad no esta muy lejos de ser alcanzada si se domina el herramental experimental y matemático ya mencionado. PIV traslativo es una estrategia que permite hacer una medición de calificación 3D-2D, o en la clasificación de Hinsh como (3,2,0).

Cuando se posee información de las componentes de velocidad proyectadas sobre dos planos paralelos muy cercanos (Figura 2.10), una formulación diferencial apropiada del principio de conservación de masa permite obtener información de la tercera componente de velocidad. En el caso de un flujo no compresible la ecuación de continuidad (2.19) es la correspondiente. Definiendo al vector genérico de velocidad según

$$\vec{V} = u\ \hat{x} + v\ \hat{y} + w\ \hat{z}\,,$$

Entonces

$$\frac{\partial w}{\partial z} = -\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right). \tag{2.19}$$

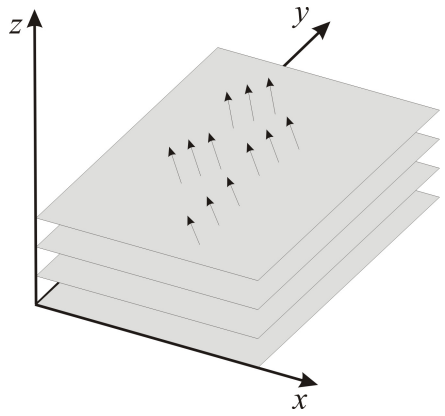


Figura 2.10 Planos de resolución para PIV traslativo

Una medida absoluta de esa tercera componente puede conseguirse si se aporta al problema una condición de contorno, típicamente la condición de no deslizamiento en las paredes del dominio que se representa con la función *f(x,y)* constante de integración en *z* en la ec.(2.20). Suponiendo constante los valores de las derivadas de *u* y *v*, se computa la componente *w* según:

$$w = -\int_{z_1}^{z_2}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)dz + f(x,y) \approx -\left(\frac{\Delta u}{\Delta x}+\frac{\Delta v}{\Delta y}\right)\Delta z + f(x,y)\,. \tag{2.20}$$

2.8 Errores en PIV

2.8.1 Errores físicos

Pueden distinguirse tres fuentes principales de error de origen físico en la técnica PIV:

- error por gradientes
- error de seguimiento
- error por aceleración

Se indicó con anterioridad que el método estadístico de usar correlación de imágenes en PIV produce una valor medio del movimiento de todas las partículas en una ventana de interrogación. Si se produce una fuerte deformación del patrón de partículas, por ejemplo una rotación, las correlaciones resultarán bajas y en ocasiones un pico máximo indistinguible o erróneo.

Si las partículas no siguen con buena fidelidad al fluido pierde mucho sentido toda etapa posterior del método. Sin embargo, como ya se discutió, sobre este problema (error de seguimiento) se tiene cierto control experimental variando tamaños y materiales de las partículas.

La hipótesis más fuerte de PIV es aproximar la velocidad *euleriana* local midiendo el desplazamiento *lagrangiano* de las partículas. Las mediciones discretas en tiempo no permiten obtener información de las trayectorias durante esos intervalos Δt. Por esta razón se desea reducir Δt, sin embargo juega en contra al existir errores aleatorios que quitan resolución al método. En [2] se deriva una separación entre pulsos de iluminación óptima:

$$\Delta t = \sqrt{\frac{2\sigma_{rand}}{Ma}}\,,$$

donde *a* es la aceleración local de la partícula, *M* la magnificación dada por el sistema, y el error aleatorio σ_{rand} suele estimarse con la ec. (2.21) de más adelante.

2.8.2 Errores de captura y procesamiento

En la captura y procesamiento de información de video para realizar PIV por algún método de correlación, el ruido inmerso en las señales produce una componente de error aleatorio en los resultados. También genera incertezas aleatorias el efecto de corrimiento del pico máximo de correlación por presencia de otros picos menores en el plano de correlación por mal ajuste de algunas pares de imágenes. Para el caso de correlación cruzada, en [1, 3] muestran una relación proporcional entre este error aleatorio y el diámetro efectivo de las imágenes de las partículas, según la cual:

$$\sigma_{rand} = \alpha\, d_{ef}\,, \qquad \alpha \in (\ 0.05,\ 0.10)\,. \tag{2.21}$$

También se reconoce [3] un error de corrimiento en el cálculo con precisión sub-píxel del pico de correlación. El mismo aparece con cualquiera de los mecanismos comentados anteriormente (ajuste de curvas, centro de masas). Suele llamarse al efecto *píxel-locked* por verificarse el corrimiento hacia la medida en píxel entera mas cercana cuando el desplazamiento real no coincide con un numero entero de píxel o dista de serlo en 0.5 píxeles. En general es posible obtener una función corrección del efecto y hacer una evaluación estadística con histogramas de desplazamientos.

Se reconoce [4] que el efecto de corrimiento es importante cuando se reduce el tamaño de partícula ($\approx$ 1píxel), y se encuentra una relación de compromiso con el error σ_{rand} en $d_{ef} \approx 2$ píxeles.

En trabajos numéricos como [3] se estudian los distintos algoritmos y errores con un modelo de imagen de partícula que sigue la ec. (2.22) y la "pixelización" del campo de partículas con la ec. (2.23). Esta última aproxima una manera de representar el efecto del sensor que captura las imágenes con una función de respuesta lineal.

$$I_{particula}(x, y) = I_0 \exp\left(- \frac{(x - x_0)^2 + (y - y_0)^2}{2{d_{ef}}^2} \right). \tag{2.22}$$

$$I_{i,j} \tilde{\propto} \iint_{pixel\ (i,j)} I_{sensor}(x, y)\ dx\ dy\ . \tag{2.23}$$

Finalmente, en el procesamiento de los vectores es común que un porcentaje de los mismos sean falsos y su inexactitud no se deba a los errores arriba comentados sino a malas correlaciones. En general pueden descartarse mediciones con bajos valores de correlación frente al nivel medio del plano de correlación. Otro modo es realizando comparaciones de cada vector con sus vecinos y fijando criterios de rechazo.

2.9 Aplicaciones modernas en mecánica de fluidos

Comentamos algunos trabajamos de diversas áreas donde se aplica velocimetría PIV.

En [4] se describen varias versiones de PIV aplicadas en un túnel de viento para el diseño de aeronaves, principalmente para el estudio de los campos de velocidades en los costados del fuselaje, extremo y flaps de las alas. Los resultados obtenidos se resumen en 6000 mapas de vectores 2D de tamaño 47 x 29, y muestran una distribución de velocidad y contornos de vorticidad.

El trabajo en [10] muestra estudios experimentales con PIV que buscan identificar y documentar, entre otras cosas, el flujo turbulento inducido en un perfil hidráulico (especialmente en los puntos de fuga) en altos número de Reynolds.

La descripción de [11] indica el interés dentro de la medicina por el estudio de los flujos en cavidades ya que se interpreta un vínculo cercano entre las lesiones en las paredes y las características del flujo. Allí proponen un método para visualizar los cambios del esfuerzo de corte en las paredes, y lo consideran un desarrollo alrededor de PIV.

2.10 Resumen

La velocimetría PIV mediante plano laser es una herramienta moderna de creciente aplicación a muy diversos problemas experimentales de mecánica de fluidos, ya sea en gases, líquidos o flujo multifase. Su potencial se sustenta en la posibilidad de medir varias componentes de un campo distribuido de velocidades sin importantes perturbaciones del medio. Su principal desventaja es que requiere de interfases transparentes o semitransparentes. Para cada proyecto de Plano Laser es necesario resolver problemas en óptica, láseres, electrónica, computación y principalmente definir y saber que se necesita medir, con que precisión y para que del problema fluidodinámico en si mismo.

Una revisión de los algoritmos más difundidos de velocimetría PIV basada en plano laser deja ver las ventajas de la correlación cruzada frente a los métodos de auto-correlación y seguimientos de partículas: mayor tolerancia al ruido, definición de la dirección y el sentido de movimiento, además de la capacidad de medir desplazamientos muy pequeños.

Además de su alto paralelismo, los algoritmos PIV de correlación pueden simplificarse al grado de facilitar su implementación y ejecución con alta velocidad.

Referencias

[1] A. Prasad, *Particle Image Velocimetry*, Review Article, Current Science, Vol. 79, No.1, July 2000

[2] E. Ifeachor, B. Jervis, *Digital Signal Processing – A Practical Approach*, Addison-Wesley, 1993

[3] K.Hinsch, Meas. Sci. Technol., *Three-Dimensional Particle Velocimetry* 1995, 6, 742-753

[4] C. Willert, M. Raffel, J. Kompenhans, *Recent Applications of Particle Image Velocimetry in Large-Scale Industrial Wind Tunnels*, 1997, 258-266.

[5] R. Adrian, *Twenty Years of Particle Image Velocimetry*, 12th International Symposium on Applications of Laser Techniques to Fluid Mechanics, Lisbon, 2004

[6] F. Bonetto, *Dinámica de Fluidos con Laser*. Capítulo 15 del Libro "Introducción al análisis científico del aparato cardiovascular en bioingeniería", Editores: E. Cabrera Fischer y M.R. Risk, editado por la UTN(FRBA) y el Comité de Ciencias Básicas de la Federación Argentina de Cardiología, 2004

[7] Y. Pu, L. Cao, H. Meng, *Fundamental Issues and Latest Developments in Holographic Particle Image Velocimetry*, Proceedings of IMECE2002, November 2002

[8] N. Kehtarnavaz, M. Gamadia, *Real-Time Image and Video Processing: From Research to Reality, Morgan & Claypool*, 2006

[9] S. Cancelos, *Jet bifásico vertical ascendente,* Trabajo especial de Ingeniería Nuclear, Instituto Balseiro, 2000

[10] D. Bourgoyne, C. Judge, S. Ceccio, *Lifting Surface Flow, Pressure, and vibration at High Reynolds-Numbers*, ASME-IMECE, 2001

[11] U. Kertzscher, P. Debaene, K. Affelg, *New method to visualize and to measure the wall shear rate in blood pumps*, 4th Int. Symp. on PIV, Göttingen, 2001

Capítulo 3

Elementos para el Diseño en Lógica Programable

Un amplio conjunto de tecnologías y técnicas en el campo de la electrónica digital y el manejo masivo de datos se presentan en muchas soluciones modernas y diversos campos de la ingeniería. En este capítulo describimos aspectos básicos del diseño de lógica digital y su utilización para ejecutar algoritmos de procesamiento de datos y cumplir con solicitudes de velocidad. Presentamos herramientas de diseño como los códigos HDL (VHDL / Verilog) para la descripción de hardware y una introducción a los circuitos integrados FPGA, donde puede implementarse lógica sincrónica de manera flexible. Existen elementos lógicos y componentes sincrónicos presentes en la mayoría de los diseños que resulta útil conocer: multiplexores, registros de datos, registros de corrimiento, sumadores, multiplicadores, *buffers* de triple estado.

Se comenta la transferencia de datos mediante interfases serie de diferentes prestaciones como Ethernet, RS-232, RS-422, USB, y paralelas como PCI. Describimos brevemente las señales diferenciales tipo LVDS empleadas en comunicación de video digital con salidas Camera Link. Por su ubicuidad, presentamos una breve referencia de los protocolos de comunicaciones en capa, con mención especial de IP / UDP por ser preferente en redes con pocos nodos. En la conexión directa entre dos circuitos integrados son comunes los estándares I^2C y SPI, con los cuales desde un chip FPGA se puede tener control de sistemas con propósitos específicos como conversores ADC / DAC, digitalizadores de video, etc.

Para diseñar sistemas de captura de video y administración de sus cuadros describimos algunos estándares, en especial a YCbCr y NTSC. La tecnología de memorias también conforma una parte crítica en el manejo de grandes cantidades de datos.

Cuando un diseño embebido en un dispositivo de lógica programable supera cierta cantidad de partes funcionales suele necesitar de una estrategia que eleve el nivel de abstracción del diseñador. El tamaño de muchas FPGAs modernas permite implementar arquitecturas de bus con un microprocesador como master principal de esos canales de datos y control. Se describe la estrategia de acceso directo a memoria (DMA) para transferencias eficientes de datos entre memorias de un sistema y las interrupciones como mecanismo para la atención de eventos en un microprocesador.

3.1 Diseño de Lógica Digital

3.1.1 Conceptos fundamentales

Primeramente debemos tener en cuenta que todo *sistema digital* se sustenta en el *diseño lógico*, cuyas bases matemáticas están en el *álgebra de Bool*. En la práctica se habla de la *lógica sincrónica*, dentro de cada máquina que ejecuta un algoritmo o proceso computacional, ya que es el método determinista que permite a la misma avanzar de a pasos en sus funciones.

El planteo de un proyecto digital significa representar digitalmente las tareas asociadas, que siempre se componen de operaciones y ecuaciones lógicas que admiten cierta manipulación booleana. En esa acción de diseño son herramientas comunes los *mapas de Karnaugh* y la representación de los valores booleanos en *números binarios* y *hexadecimales*. Es el punto de partida para realizar funciones prácticas con los elementos lógicos.

En esa resolución siempre participan operaciones lógicas de alto nivel como sumas, restas, multiplicaciones y divisiones que implican cierto grado de complejidad. Una revisión de esa teoría permite ver cuestiones asociadas como la representación de números negativos, los conceptos de complemento a uno y a dos, bits de acarreo, los intensos pasos lógicos de una multiplicación y las dificultades que involucran los procesos de división.

Como dijimos, el complemento del álgebra booleana es el diseño lógico sincrónico cuya materialización se lleva a cabo con dos elementos tecnológicos muy importantes: las *señales de reloj* distribuidas y los elementos que permiten mantener en el tiempo estados de progreso llamados *flip-flops*. La explicación reside en la capacidad de almacenamiento y manejo de datos que los flip-flops brindan, permitiendo administrar un avance temporal en la resolución de un problema digital. En síntesis, como lógica sincrónica se denomina a una colección de *compuertas lógicas* y flip-flops controlados por un reloj común. En un circuito sincrónico la transición y establecimiento de todos sus flip-flops ocurren al mismo tiempo. Si bien es una idealización, gran parte de los diseñadores digitales lo aplican por simplificar el análisis de los diseños. En la práctica se contempla en alguna medida con componentes que permiten una distribución suficientemente buena de la señal de reloj (*clock distribution*). Entran en juego las características de los *osciladores*, *buffers* de reloj, *reguladores de fase* de reloj (*Phase Locked Loop - PLL*) y *reguladores de retardo* de reloj (*Delay Locked Loop - DLL*), etc.

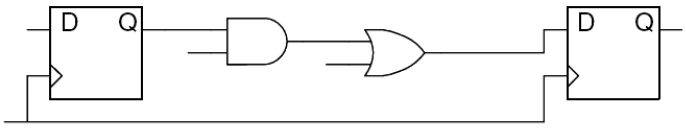


Figura 3.1 Ejemplo de un circuito con lógica sincrónca. Se aprecian el flip-flop de salida, el flip-flop de llegada y lógica combinatoria en el camino. La señal de reloj posee una misma fuente.

Llamamos lógica combinatoria a como se realimentan los nuevos *estados de un sistema*. Consiste en varios elementos lógicos sin estado asociado cuyos niveles de salida son capturados por los flip-flops.

Tratar con lógica sincrónica involucra incorporar al tiempo en un diseño. Se establecen pautas para definir a que velocidad puede trabajar correctamente un circuito. Entran en juego la atenuación y el retardo temporal de los componentes electrónicos que conforman los elementos lógicos. Cada elemento tiene asociado un *retardo de propagación* entre que se presenta a su entrada una señal y se genera la salida correspondiente. La conexión entre si de varias compuertas hace que el *retardo global* de esa propagación entre extremos de una red con lógica combinatoria aumente.

El *análisis de tiempos* es importante ya que permite determinar como impactan esos retardos en la velocidad de operación del sistema sincrónico. Como el tiempo se discretiza según un reloj, siempre comparamos los retardos con el periodo de dicha señal de reloj que actualiza el estado de los flip-flops.

Un proceso en cascada, o *pipelining*, es un modo de dividir una *ecuación booleana* en varias ecuaciones menores y calcular los resultados parciales durante ciclos de reloj sucesivos. Se requieren menos compuertas lógicas cuando las ecuaciones son menores, por lo tanto el retardo de propagación total de una señal es menor relativo a la ecuación completa y permite a la lógica sincrónica funcionar a mayor frecuencia (pero la ecuación completa no se procesa necesariamente más rápido).

Términos necesarios para entender buena parte del diseño sincrónico son los de *desplazamiento de señal de reloj* (*clock skew*) y *variaciones de la señal de reloj* (*clock jitter*). El primero es una medida de cuan distinto en tiempo pueden llegar los flancos de reloj a diferentes flip-flops dentro de un mismo dominio sincrónico. Por *clock jitter* se entiende al comportamiento no ideal del generador de señales de reloj, ya que se producen pequeñas variaciones en el tiempo entre dos flancos de dicha señal periódica.

Las compuertas básicas y los flip-flops pueden combinarse para formar estructuras más complejas, muchas con funciones comunes. Mencionamos algunos a continuación.

Los *multiplexores* (*mux*), también llamados *selectores*, sirven como interruptores lógicos de múltiples posiciones. Un *demultiplexor* (*demux*) realiza la operación inversa.

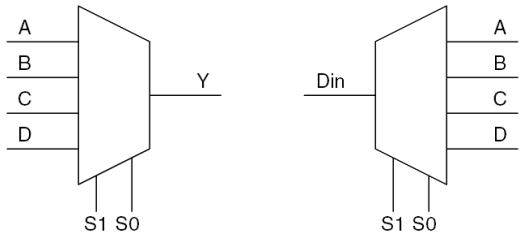


Figura 3.2 Muliplexor 4 a 1 y Demultiplexor 1 a 4

Los *buffers triple-estado* son elementos combinatorios que pueden activar tres formas de estado a su salida en lugar de solamente los estados '0' y '1'. El tercer estado es el de *alta impedancia*, Z, o desconectado. Esto permite a varios dispositivos compartir un cable o canal de salida.

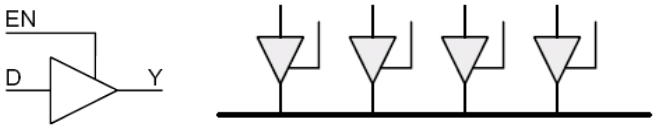


Figura 3.3 Símbolo de un buffer triple-estado y su empleo en varias conexión a un canal común

Los *registros* son colecciones de varios flip-flops agrupados en una función conjunta. En general se agrupan según el tamaño de una unidad de datos, por ejemplo 8 bits (*b*) para un byte (*B*). Los registros poseen una entrada de reloj común (*clk*) y una de habilitación de reloj (*ce*) para ejercer control en la actualización del registro. Los *registros de desplazamiento* (*shift registers*) tienen la función adicional de manipular los bits, y con ellos es posible hacer conversiones entre datos en serie y paralelos.

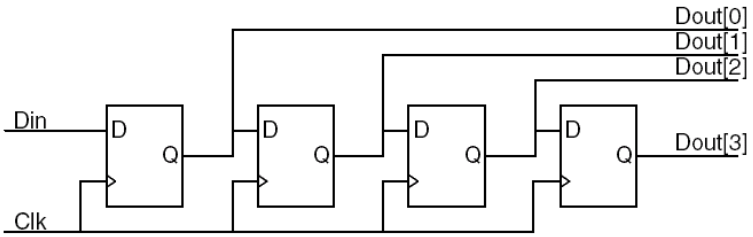


Figura 3.4 Registro de desplazamiento con entrada serie y salida paralela

Los *contadores* permiten realizar conteos ascendentes o descendentes según como se implementen. Sus señales de salidas pueden servir para activar funciones cuando el contador alcanza algún valor, realizar selecciones o para identificar un orden en datos que se transmiten como serie de valores, entre otros.

3.1.2 Diseño Lógico y la implementación de algoritmos genéricos en HW

Entender el flujo de diseño de un sistema digital moderno cobra utilidad en la implementación de algoritmos de aplicación específica con algún nivel complejidad y solicitudes de velocidad. Muchas partes de un algoritmo pueden ejecutarse en *software* (*SW*) por su flexibilidad y fácil programación de funciones secuenciales. Pero ciertas funcionalidades solo pueden implementarse en *hardware* (*HW*) dedicado, como la captura y almacenamiento de datos. Algunos sistemas no pueden ejecutar todas sus tareas en *SW*, sobre todo por la velocidad insuficiente del *microprocesador* asociado.

Existen tres definiciones fundamentales de velocidad en el ámbito del HW: *rendimiento* (*throughput*), *latencia* (*latency*) y *retardo crítico* (*timing*). Como *throughput* entendemos a la cantidad de datos que se procesa por *ciclo de reloj* (bits/s o bps). El tiempo entre el ingreso de los datos y la salida de los resultados de un proceso se denomina latencia. El término *timing* se refiere a los retardos lógicos entre elementos conectados en serie. No cumplir el *timing* significa que el periodo de reloj es menor a los retardos del camino crítico en el circuito. Esas medidas de tiempo se componen de retardos combinatorios, tiempo de establecimiento de salida – *clock-to-out* - y de llegada – *setup* -, *clock skew*, etc.

Las *máquinas de estados finitos* (*FSM*) se utilizan para implementar algoritmos en hardware. De igual modo que un programa es ejecutado en pasos secuenciales por un microprocesador, las máquinas de estado se diseñan para avanzar cuando se cumplen ciertas condiciones; y en ese progreso pueden activar otras funciones de igual manera que un microprocesador solicita acciones a sus periféricos.

Cuando se ejecutan algoritmos complejos el tamaño de una máquina de estados puede crecer en tamaño (por ejemplo en número de estados) y presentar problemas de tiempos. Los conflictos se deben a la existencia de muchos niveles lógicos necesarios en un cambio de estado o definición de las salidas. Por esta razón el diseño óptimo de máquinas de estados complejas tiene en cuenta desde un principio detalles de cada parte de su arquitectura global, por ejemplo más paralelismo dividiendo una tarea en partes menores. Las estrategias de optimización en el diseño de una máquina de estado tienen en cuenta su partición, método de representación del *vector de estado* (*one-hot*, *binario*, etc.) y uso de *pipelining*.

Las máquinas de estado se clasifican en dos tipos: *Mealy* y *Moore*. Las de Mealy determinan sus valores de salida basadas tanto en el estado presente como en el valor de las entradas, mientras que la de Moore solo en su estado.

La evolución en las técnicas de diseño también se da en el ambiente del hardware. Al igual que la programación de SW para microprocesadores actualmente se realiza en general a un alto nivel (poca programación en Assemby y mucha en lenguajes tipo C/C++), existen *herramientas de diseño lógico* automatizadas como son los *lenguajes descriptores de hardware* (*HDL*), por ejemplo *VHDL* y *Verilog*. Actualmente, un diseñador de HW pocas veces necesita atender la interconexión de compuertas o la simplificación lógica, para enfocar la mayor parte de su trabajo en la funcionalidad del algoritmo. HDL se adapta muy bien para proyectos digitales de diversas complejidades, escalas y especificaciones.

El producto final de toda descripción de HW digital puede encontrar una implementación robusta en *circuitos integrados* (*ICs*) muy a medida de los requerimientos. Si bien las posibles tecnologías de implementación difieren según costo, velocidad, propiedades físicas (térmicas, mecánicas, de resistencia a la radiación), existen denominadores comunes: obtener agrupación de funciones lógicas arbitrarias y flip-flops dentro del mismo ICs. Los chips *ASICs* (*circuitos integrados de aplicación específica*) son un caso importante; se caracterizan por tener lógica muy específica y una configuración fija. También los *PLDs* (*dispositivos de lógica programable*), cuya configuración es programable y se utilizan en muchas aplicaciones arbitrarias y de prototipado. En esta última categoría se encuentran las FPGA descritas más adelante.

En los últimos años dejó de ser uso exclusivo en pocas empresas el diseñar con HDL. Su popularidad sigue en aumento, y en la Argentina ya es herramienta frecuente en varias universidades [2, 3] y empresas de intensivos desarrollos electrónicos como INVAP SE, la CONAE y el INTI.

3.1.3 Implementando funciones matemáticas

Los operadores aritméticos de *punto fijo* comúnmente implementados en lógica digital son los sumadores y multiplicadores. Las restas en general se realizan mediante una transformación del número y una suma. Cuando un sistema necesita realizar sucesivas sumas de multiplicaciones suele implementarse un *acumulador de multiplicaciones* (*MAC*) que opera secuencialmente con los resultados de un multiplicador, o una *suma de operaciones* (*SOP*) que implementa una arquitectura de sumas en cascadas.

Las divisiones siempre representaron una dificultad para el diseño digital [3]. No existe una operación lógica simple que resuelva un cociente, y por otro lado sus resultados no son finitos ni predecibles en una representación de punto fijo.

Usando multiplicaciones y desplazamientos de bits (*shift*) es posible realizar divisiones. La idea es calcular la inversa del denominador y multiplicarla por el numerador, teniendo presente que un corrimiento de los bits hacia el menos significativo es equivalente a una división por 2. Por razones explicadas en [6] suele emplearse solamente para denominadores fijos o cuando la precisión requerida es baja. Más genérica es la técnica de la *división iterativa* aunque presenta altas latencias. El *método de Goldschmidt* permite resoluciones en cascada con altos rendimientos. Por último mencionamos a las tablas de búsqueda (LUTs o *Lookup Tables*) como mecanismo de acceso a una memoria que guarda representaciones en punto fijo del valor de la inversa del denominador.

El cálculo de funciones matemáticas genéricas puede implementarse mediante *aproximaciones en series de Taylor*. Una función compleja queda reducida a un polinomio que solo requiere multiplicadores y sumadores en el hardware. En el caso de las funciones trigonométricas los algoritmos *CORDIC* (*Coordinate Rotational Digital Computer*) son una de las frecuentes opciones para el cómputo de funciones trigonometría [6, 7].

3.1.4 Cambios de dominio de reloj

Cuando existen múltiples dominios de reloj en un diseño, no hay garantías de la relación entre los flancos de reloj entre esos dominio. Las diferencias pueden ser por fase o desplazamientos (*skew*). El análisis de tiempos sincrónicos señala que los tiempos de *establecimiento* (*setup)* y tiempos de *mantenimiento* (*hold*) en un flip-flop deben satisfacerse para la captura correcta de un estado lógico. Pero esto es imposible de hacer si no se conoce la relación entre el reloj del flip-flop de salida con el reloj del flip-flop de llegada.

Trasmitiendo directamente una señal de un dominio de reloj a otro, puede generar que la señal se capture, se pierda o provoque un efecto llamado *metaestabilidad*. La metaestabilidad es un comportamiento pseudos-aleatorio del flip-flop que en general se evita en los diseños sincrónicos; si bien las señales alcanzan su estado final el problema es la indefinición de cuándo lo hacen. Existen técnicas a tener en cuenta para conectar entre dominios de reloj distintos, tanto señales de control como buses de datos.

La sincronización de señales de control puede realizarse con varios flip-flops en el camino de la señal y activados por el reloj del dominio de llegada. Luego la señal puede utilizarse como sincrónica. Tratar con la metaestabilidad es tratar con probabilidades; y empleando dos flip-flops se considera muy baja la probabilidad de que se transfiera la metaestabilidad [6].

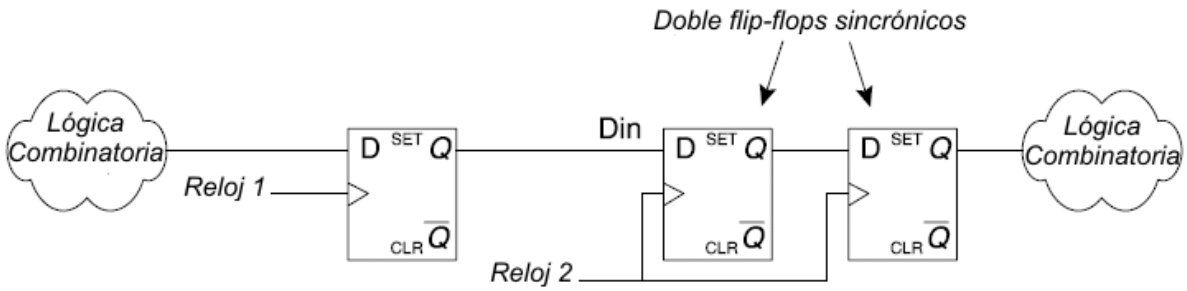


Figura 3.5 Pasaje de una señal de control entre dominios de reloj mediante doble flip-flops

Cuando se tiene control de al menos uno de los dominios de reloj, y uno de los relojes tiene un periodo múltiplo del otro, pueden ajustarse las fases de los mismos con elementos PLL o DLL. De esa manera no se violan las solicitudes de tiempos.

Las estructuras de *memoria tipo cola* (*first-in first-out* o *FIFO*) pueden usarse cuando se pasan múltiples señales o buses de datos. El caso más simple es cuando no se trabaja con banderas de control de estado lleno o vacío de la memoria. En cierto tipo de transferencias puede usarse una *códificación Gray*, en la cual cada número solo tiene un dígito binario diferente de su predecesor, y se utiliza para transmitir datos, por ejemplo, de un contador de varios bits usando pocos recursos.

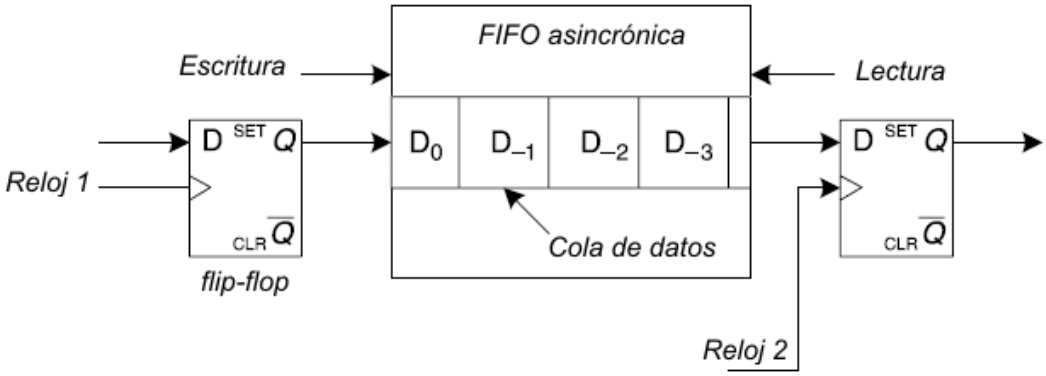


Figura 3.6 Memoria de cola (FIFO) asincrónica

Las prácticas de diseño digital [1] sugieren hacer bloques sincrónicos separados por *bloques de sincronización* para simplificar el análisis de tiempos de un diseño.

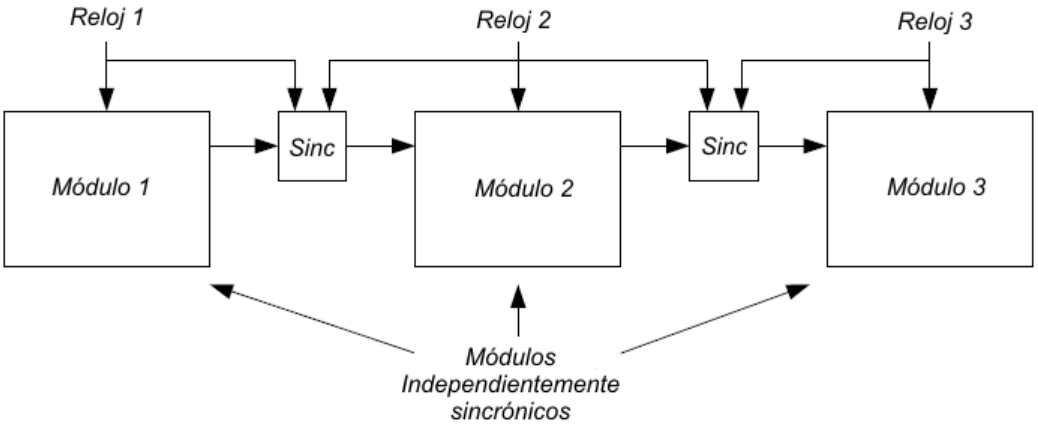


Figura 3.7 Separación de diseños sincrónicos por módulos de sincronización

3.2 Lógica Programable y Sistemas Embebidos

Los *dispositivos de lógica programable* son el medio de implementación de lógica a medida usada por muchos grupos de ingeniería. La mayoría de la lógica programable se implementa con algún lenguaje HDL que libera al ingeniero de deducir y minimizar expresiones booleanas. Las ventajas de la lógica programable son la rapidez con que permite disponer de dichos circuitos a medida, a veces muy complejos, y con costos relativamente bajos. Esto permite a individuos, compañías y laboratorios pequeños disponer de circuitos integrados con lógica muy personalizada.

Existe un amplio rango en la oferta de dispositivos de lógica programable, y su selección en general requiere una investigación preliminar que atiende a las necesidades del usuario [4].

3.2.1 Generalidades de Dispositivos de Lógica Programable *PLDs*

Los dispositivos de lógica programable (*PLD*) se constituyen de recursos lógicos de propósito general que pueden conectarse en muchas combinaciones de acuerdo a la lógica diseñada. Esta conectividad programable viene dada por otro tipo de lógica incorporada que permite hacer esas conexiones dentro del chip. El gran beneficio es la rapidez con que el diseñador puede descargar al chip su diseño, sin que intervenga proceso alguno de manufactura. Muchos PLD son reprogramables, y tienen la ventaja de permitir al ingeniero corregir errores y evaluar los cambios con rapidez. Otros solo admiten ser programados una sola vez.

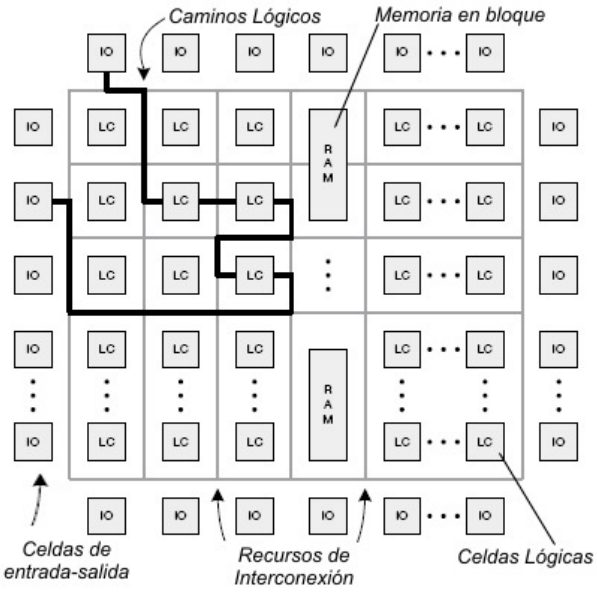


Figura 3.8 Componentes típicos en una FPGA

La desventaja de los PLD esta en la lógica oculta que permite las interconexiones programables entre compuertas lógicas. Generan mayores costos unitarios, menores velocidades, y aumento en el consumo de potencia. Los retardos de propagación son

inherentes a toda estructura de silicio, y cuantos más elementos hay en un camino, más lento será el funcionamiento. Sin embargo estas oposiciones son salvables en muchos casos, a veces superadas por los beneficios comentados.

Existe un tipo de PLD llamado FPGA (*Field Programmable Gate Array*) que se caracteriza por tener cientos o miles de flip-flops en su interior, además de permitir programar lógica combinatoria. Son importantes por servir en aplicaciones que examinan y manipulan el contenido de flujos de datos, ya que se requieren para dichas tareas grandes cantidades de registros en cascada, almacenamiento temporario de datos, contadores y largos indicadores de como se encuentra una máquina de estados (vectores de máquina de estado). También poseen bloques de memoria interna (Block RAM o BRAM) para almacenamientos intermedios (*buffering*) de datos o para la implementación de colas de memoria (FIFOs).

La característica más referenciada de una FPGA es su fina arquitectura compuesta de un arreglo de pequeñas *celdas lógicas*. Cada celda lógica posee componentes de aplicación genérica como algunos flip-flops, pequeñas *tablas lógicas* (*lookup tables* o *LUTs*), uno o varios multiplexores, mecanismos de acarreo para sumadores y contadores. Las expresiones booleanas se evaluan en las LUTs (construidas con tecnología SRAM), que implementan una funciones donde para cada combinación de entrada la regla lógica definirá el valor lógico de salida. Se habla de *área* para generalizar la cantidad de estos recursos que demanda un diseño.

Las celdas se agrupan junto a recursos de interconexión que pueden realizar los enlaces arbitrarios entre las celdas. Según la FPGA, recursos especiales (o dedicados a cierta función) se ubican entre el arreglo de celdas lógicas. Por ejemplo, elementos de distribución de señales de reloj (*Digital Clock Manager, Clock Buffers, etc*), o *bloques de memoria RAM* (pueden ser configurables en *simple* o *doble puerto*, según *ancho de palabra*, etc.), *multiplicadores*, etc.

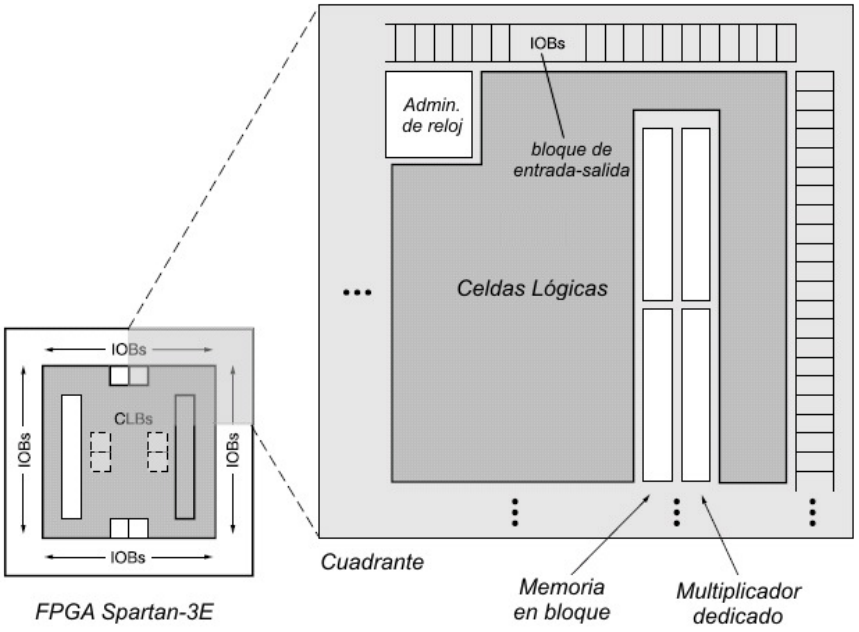


Figura 3.9 Arquitectura de la familia de FPGAs Spartan-3E de Xilinx Inc.

En la periferia del chip FPGA se encuentran las *celdas de entrada / salida* (*I/O cells*). En general hay flip-flops en ellas para capturar la señal de entrada y minimizar las distancias de transito hacia el interior del chip. Es posible optar por varias modalidades eléctricas para la conexión externa de dichos puertos. En general la tecnología de las FPGA es SRAM, por lo que sus configuraciones son volátiles y deben reprogramarse cada vez que se energiza el chip.

Existen características de alto nivel que diferencian entre si a las FPGAs más allá de los recursos lógicos y cantidad de pines de entrada-salida (I/O). Estos recursos son los elementos de distribución de reloj, memoria embebida, módulos de funcionalidad dedicada y embebidos, y celdas I/O con múltiples funciones.

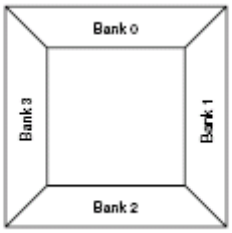


Figura 3.10 Separación en bancos de entrada-salida en la FPGA Spartan-3E

Algunas FPGAS soportan varias señales de reloj, pero poseen restricciones al número que puede funcionar en una dada porción (por ejemplo una división en cuadrantes) del chip. Diseños grandes suelen requerir varios relojes, principalmente cuando participan muchos periféricos y existen varias interfases externas. Las FPGAs proveen recursos para la distribución de señales de reloj con bajos corrimientos (*low-skew*) en forma global, como pistas distintas (*clock distribution network*) a las usadas para transferir señales de lógica. Muchas FPGAs proveen componentes DLL y PLL para corregir (*deskewing*), dividir, y multiplicar señales de reloj generadas externamente.

Los bloques de memoria embebida (*Block RAM* o *BRAM*) se emplean en aplicaciones de procesamiento de datos como pequeñas memorias intermedias (*buffering*), memorias de cola (FIFO), etc. Sin ellas, muchas ventajas en la velocidad que poseen las FPGA no se aprovecharían. Son configurables y flexibles para adaptarse en aplicaciones diversas, se clasifican según su densidad de bits y la capacidad de trabajar como simple o doble puerto. Su capacidad de memoria puede usarse en varias configuraciones de ancho / número (*width / depth*) de palabras. La escritura y lectura simultánea es posible usando el modo doble puerto. Pueden tener interfases sincrónicas o asincrónicas y puede soportar dos relojes en el modo sincrónico, algo empleado en las memorias FIFO para transferir datos entre dos dominios de reloj.

Las FPGAs permiten emplear sus tablas LUTs como memorias (*memorias distribuidas*), aunque es más eficiente hacerlo con las BRAMs cuando las estructuras de datos son grandes. Para pequeños volúmenes es preferible usar memoria distribuida.

En la arquitectura de las celdas I/O se consideran las funcionalidades sincrónicas y los niveles de corriente-tensión. Muchas pueden configurarse para operaciones solamente de entrada, de salida o bidireccionales, con facilidades de buffer de triple-estado. Para requerimientos de tiempo estas celdas incluyen flip-flops para todos los modos de operación.

3.2.2 Diseño en FPGA

El proceso de diseño con lógica programable implica pasos que van desde la definición del circuito (HDLs, esquemáticos, etc.) hasta su funcionamiento. Intervienen varias herramientas de software y al ciclo se lo llama *flujo de diseño* (*design flow*).

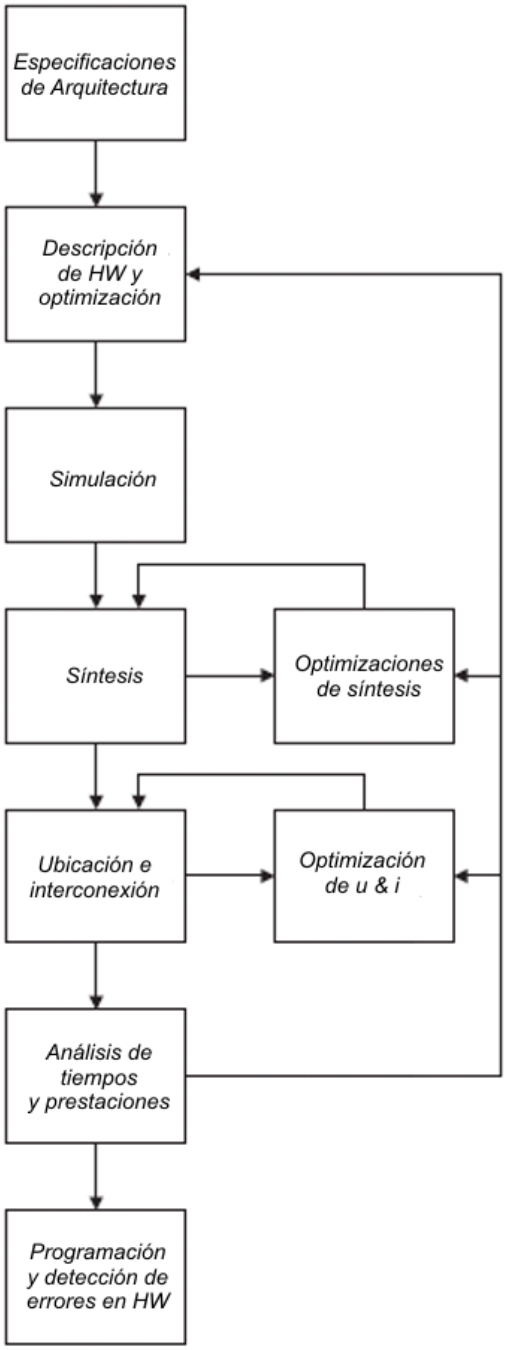


Figura 3.11 Flujo de diseño en FPGA

La descripción del diseño es el punto de partida. Consiste en escribir código HDL, hacer diagramas esquemáticos, usar herramientas gráficas, etc. La etapa de generación (*Generate*, *Translate*) convierte todo el diseño en una descripción HDL.

La traducción entre el código HDL y la descripción en componentes lógicos del dispositivo programable se llama *síntesis* (*Synthesis*), y su resultado es una formulación a más bajo nivel del diseño que se llama *netlist*. La etapa de *ubicación e interconexión* (*Place and Route*) significa decidir de cuales componentes lógicos dentro del chip se implementarán, y como se conectarán entre si y con los pines de entrada-salida. Este paso emplea información complementaria de *condiciones* físicas (*Constraints*) requeridas por el diseñador. Un ejemplo es la asignación de pines y sus estándares eléctricos. Diseños con altos requerimientos emplean algoritmos iterativos para la Ubicación e Interconexión.

Todo el sistema finalmente se resume en un *archivo binario* (*Bitstream*) usado en la *programación del PLD*.

En forma intermedia a lo descrito están los pasos de *simulación y verificación*. Para simular se *compila* (*Compile*) el código HDL a un formato más rápido para los simuladores. La simulación podemos realizarla con el código HDL original (verificamos el comportamiento lógico), luego de sintetizarse (pueden incorporarse mejores modelos de retardo en el análisis), o luego de la ubicación e interconexión (*Place and Route*) que es la más exacta por incluir información de la arquitectura final.

Dependiendo de la complejidad del diseño, generar el esquema de conexiones puede llevar de minutos a horas en una computadora convencional usando programas de diseño provisto por los mismos fabricantes de la FPGA (Actel, Altera, Laticce, Xilinx, etc) o por terceras empresas (Mentor, Symplify, etc).

Como puede haber fallas internas en los integrados, se realiza una verificación del diseño sobre el dispositivo una vez configurado.

El estilo de programación HDL induce a las herramientas, de las etapas de flujo diseño, a implementar las cosas de ciertas maneras dentro del chip. Por ejemplo, las máquinas de estado estilo Moore y la representación estilo one-hot en general son más adecuadas para implementar en FPGAs, y algunas herramientas de síntesis pueden no tenerlo en cuenta.

Diseños muy complejos con FPGA pueden necesitar de la intervención humana en la forma de ubicar y conectar algunos bloques lógicos del diseño dentro del arreglo de celdas lógicas. Esta tarea se llama *Floorplanning* y produce nuevas condiciones para la herramienta de ubicación e interconexión. Esto reduce las permutaciones que evalúa el software, se acelera el proceso y permite diseñar con más estrategias (por ejemplo cumplir con solicitudes de emisión electromagnética o incrementar la confiabilidad del sistema al daño por radiación distribuyendo lógica redundante).

3.2.3 Sistemas Embebidos: Arquitectura de Bus / Interrupciones - DMA

Pueden resolverse problemas en el diseño de sistemas digitales usando combinaciones de hardware (HW) y software (SW) en una *arquitectura de computadora*. Un microprocesador es la plataforma del SW y puede servir flexiblemente como controlador central de un hardware con funciones específicas. En ese microprocesador puede correr el algoritmo principal de un sistema y el HW periférico servir como asistente de tareas.

En los elementos de una computadora deben distinguirse el microprocesador, la *memoria* y los *dispositivos de entrada-salida*. En el microprocesador existe un fundamento de HW en la ejecución de instrucciones y en la interacción con otros sistemas.

Un microprocesador se conecta con los dispositivos de memoria y de entrada-salida mediante canales de *datos* y *direcciones* llamados *buses*, o en conjunto se los denomina bus del microprocessor. El bus de dirección tiene capacidad de seleccionar todo el *espacio de direcciones*, que es la máxima cantidad de memoria y puertos I/O que un microprocesador puede directamente acceder. Cada dispositivo periférico se debe mapear en una región de ese espacio de direcciones.

El bus de datos posee uno o varios bytes de ancho, y marca un límite a la capacidad de manejo que posee el microprocesador. Un microprocesador de 32 bits puede operar hasta 4 bytes por vez y puede tener un bus de datos de hasta 32 bits.

Existe una situación donde el microprocesador no sigue la normal operación de una *secuencia de instrucciones*, y es cuando debe *manejar eventos* que ocurren aisladamente y dan aviso mediante *interrupciones*. Esas intervenciones pueden ser periódicas, y en cada oportunidad la administración de cada tipo de evento se describe con una particular *rutina de servicio de interrupción* (*ISR*). Una vez ejecutadas esas instrucciones, el microprocesador vuelve a operar normalmente sobre el punto donde fue interrumpido.

La *transferencia de datos* entre una región de memoria y otra es una tarea común dentro de una computadora. Por ejemplo, un controlador de comunicaciones series puede introducir datos para guardarse en una memoria. Una transferencia puede darse en una de varias combinaciones entre puertos individuales y áreas de memoria del sistema.

La velocidad con que una memoria puede transferirse normalmente depende de los tiempos que necesita un microprocesador para realizar sucesivas operaciones de lectura y escritura. En lugar de mover un *flujo continuo de bytes*, cada byte transferido por un microprocesador tiene un costo temporal (overhead) por requerir varias operaciones de actualización y control del estado de la transacción.

Con la técnica de *acceso directo a memoria* (*DMA*) una transferencia de grandes volúmenes de memoria tiene mayores rendimientos. Al requerimiento de un microprocesador de mover cierta cantidad de datos desde un origen a un destino, un *controlador DMA* toma transitoriamente el dominio del bus (función *master*) y realiza eficientemente la transferencia.

El tamaño de nuevas FPGAs permite implementar en su lógica microprocesadores (*soft processors*) y arquitecturas de bus. Se usa el término de *Sistema Embebido* o sistema en chip (*SoC*) a la integración compacta de una arquitectura de computadora con funciones específicas.

Los microprocesadores pueden dividirse en dos categorías. Existen los que soportan un conjunto de instrucciones reducido (*RISC - Reduced Instruction Set Computing*) o un conjunto complejo de instrucciones (*CISC - Complex Instruction Set Computing*). En el primer caso el sistema lógico que procesa las instrucciones es más simple, pero necesita más ciclos de reloj para ejecutar cada instrucción.

3.3 Memorias

Las memorias se necesitan en toda arquitectura de cómputo. La capacidad de

almacenamiento de una memoria por realizar transacciones en cantidades de datos y en tiempos acotados, lo cual define gran parte de la capacidad de un sistema digital.

Entre las cosas a saber de una memoria se encuentran los tiempos de interfase de sus puertos con otros elementos, su *densidad*, y los *protocolos* de comunicación. Deben distinguirse las memorias *volátiles* de las *no-volátiles*.

De las memorias de tecnología de estado sólido existe una categoría de solo lectura. Entre ellas mencionamos *EPROM*, *flash*, y *EEPROM*.

Las memorias de acceso aleatorio (*RAM*) tienen sus bases en la tecnología *SRAM* y *DRAM* asincrónicas. Hoy en día predominan las memorias *RAM* sincrónicas, como *SDRAM* y *SSRAM*. Existen tipos de memorias con una parte lógica y una parte RAM llamadas *CAM*.

Como definiciones funcionales mencionamos la posibilidad de algunas memorias de trabajar con multi-puertos y como memorias FIFO. Sus aplicaciones están principalmente en las comunicaciones. En muchas situaciones de sistemas de redes y comunicaciones las memorias tienen un rol dominante, como cuando un bloque de datos se recibe (por ejemplo línea de un cuadro de video) y debe conservarse transitoriamente hasta que una lógica defina su destino. Las tablas lógicas o de búsqueda (Lookup tables) son otro uso común de las memorias. En una tabla pueden almacenarse términos precalculados de un cómputo complejo de modo tal que un resultado pueda ser determinado rápidamente cuando se necesite.

3.4 Comunicación de datos

3.4.1 Partes de un sistema de comunicación en serie

Son frecuentes las interfases de comunicación serie como enlace de datos entre sistemas digitales. Son económicas por llevar pocos cables incluso en largas distancias, y pueden transmitir hasta Gbps. Los buses son necesarios para obtener altos rendimientos en un diseño digital, pero deben volcar su información en una serie de bits al momento de transferirlos en largas distancias.

Cada tipo de enlace utiliza técnicas específicas, pero existen conceptos de aplicación general como el modo de delinear el principio y fin de paquetes (*framing*) y la detección de errores. Para facilitar la conexión entre distintos equipos existen estándares para cada modo comunicación: RS-232, RS-422, RS-485, Ethernet, PCI, USB, I^2C, SPI, etc.

Cuando una conexión se piensa para varios nodos los esquemas de comunicaciones deben ser más avanzados. Esto incorpora conceptos complementarios como las topologías de una red y los formatos de paquetes en capas donde puede participar un software en diseños de alto nivel. Si el canal admite una transmisión bidireccional y simultánea se denomina *full-duplex*, caso contrario se llama *half-duplex*. En los canales compartidos se implementan esquemas de detección de colisiones para contener los casos de escritura simultánea por diferentes dispositivos.

Un canal de comunicaciones se compone de un medio físico de transmisión, transductores que adaptan las señales del medio a señales eléctricas dentro de los dispositivos (y viceversa), el sistema receptor / transmisor que adapta los datos para la comunicación, y el sistema de procesamiento propiamente dicho que utiliza y genera esos datos.

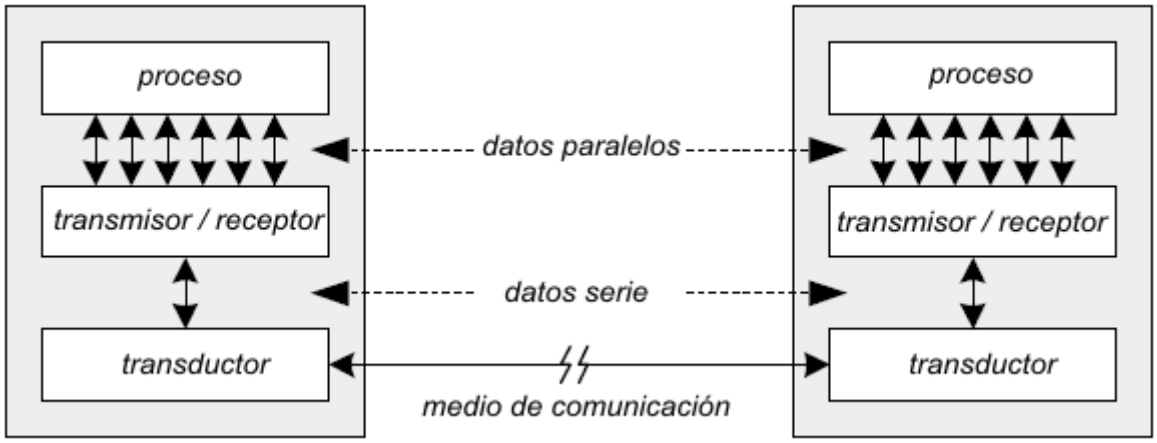


Figura 3.12 Partes de un sistema de comunicación en serie

UART

El UART (*universal asynchronous receiver/transmitter*) es un elemento receptor/transmisor básico que ordena en forma de serie datos paralelos para transmitir, y de serie a paralelo en el momento de recibir datos. Además de ésta conversión, el UART también administra la comunicación y sus funciones de sincronización. Debe encargarse del delineamiento de los datos en paquetes que llegan y se envían, que usa conceptos como bit de inicio, bits de parada, bit de paridad, etc. Otra función del UART es el control de flujo (*handshaking*) mediante el cual los extremos del enlace confirman su estado de disponibilidad para una transmisión.

Señales Diferenciales: alta velocidad e inmunidad al ruido

Las tecnologías de señales diferenciales son empleadas en aplicaciones de transmisión en alta velocidad, largas distancias o inmersas en ambientes con ruido. Se diferencian entre si por la máxima tasa de transmisión, tensiones de trabajo, potencia consumida, y se caracterizan por su alta inmunidad al ruido electromagnético. Los cableados pueden diseñarse para reducir las emisiones electromagnéticas.

Las transmisiones diferenciales se denominan balanceadas, y las que emplean un cable por señal y una tierra de retorno común se llaman desbalanceadas (*single-ended*). En las transmisiones de alta velocidad se evitan los rebotes de una señal adaptando en impedancia las líneas y sus terminaciones. Los cables coaxiales y pares trenzados en general usan 50, 75, o 100 Ω.

3.4.2 Comunicación entre circuitos integrados

Un mecanismo económico para conectar a circuitos integrados (ICs) entre si es mediante redes de muy pequeña escala. En general sus solicitudes de ancho de banda son bajas y el acceso al medio de comunicación es poco frecuente, y esto se aprovecha para dedicar pocos pines de en un circuito integrado en los enlaces.

Existen dos estándares muy usados: el I^2C (*inter-IC bus*) creado por Philips y *SPI* (*Serial Peripheral Interface*) de Motorola. Admiten algunos cientos de kbps y las redes

creadas con estas interfases típicamente tienen un dispositivo master y uno o varios esclavos.

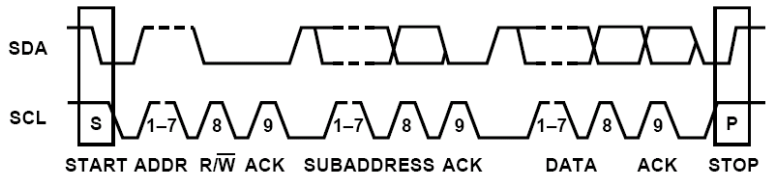


Figura 3.13 Ejemplo de transferencia en I^2C. Destacamos la relación entre los flancos de reloj (SCL) y los estados del canal de datos (SDA). Apreciar las partes de control de flujo (ACK)

I^2C soporta varios master y requiere solo dos cables, mientras que SPI necesita más cables. I^2C consiste en una señal de reloj (SCL) y una señal de datos (SDA). Ambas señales tienen la propiedad de ser *colector-abierto*, por lo que los dispositivos solo pueden fijar niveles de estado lógico bajos. Es similar a un buffer triple-estado pero solo con estados de alta y baja impedancia. Con cada señal debe existir una resistencia de *pull-up* para establecer el estado lógico '1' cuando ningún puerto esté fijando activamente el '0' lógico. Esta modalidad permite que varios ICs compartan un mismo cable sin conflictos eléctricos.

3.4.3 Comunicación de equipos: media y baja velocidad

Existen dos estándares muy usados para conexiones punto a punto: RS-232 y RS-422.

El RS-232 se convirtió en un estándar confiable y RS-422 sigue es un caso similar en aplicaciones que requieren señales diferenciales. Los estándares de representación de datos, como ASCII (*American Standard Code for Information Interchange*), también simplifican la conectividad entre equipos muy diversos.

Se utilizan en funciones de control y transmisión con bajos o medios caudales de datos (desde los kps hasta el orden de los Mbps). Con RS-422 puede transmitirse a más de 1Km a 9.6 kbps.

3.4.4 Comunicación de equipos: alta velocidad

Programación de redes Ethernet / IP-UDP

Las redes de computadoras permiten transferir datos en altas velocidades y distribuir información a varios nodos desde un mismo origen. El manejo de datos a través de redes se basa en un tratamiento de paquetes a nivel de SW y de HW. Ethernet y varios protocolos montados en él son ubicuos hoy en día.

Las capas de un protocolo de red permiten entender las distintas funciones lógicas en cada etapa necesaria para transmitir un paquete de datos. Se habla de un *protocol stack* que se descodifica cuando una aplicación desea llegar a los datos de interés dentro de cada paquete que le fue transferido. A la hora de enviar datos (o *carga útil*) estos deben complementarse de las partes de dicho protocolo que incorpora información del sistema transmisor y del receptor obviamente. Las definiciones estándares de cómo se componen esos protocolos son parte de un modelo llamado *OSI* (*Open System Interconection*).

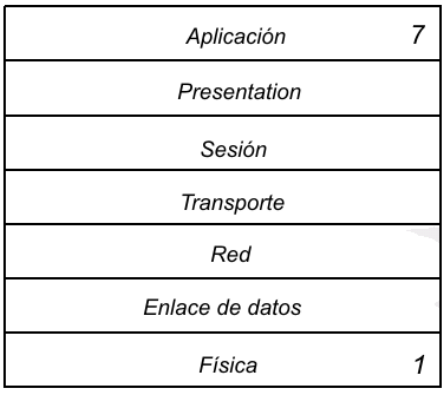


Figura 3.14 Capas del modelo de comunicaciones OSI

La capa 1 o *física* se encarga de la conectividad a través de un *medio electromecánico*. La capa 2 se encarga de la lógica de control y detección del principio y fin de los mensajes (*framing*). Esta capa se maneja por el *controlador de acceso al medio* (MAC), que es un dispositivo de hardware que realiza esas funciones para el acceso a la red. A través de direcciones asociadas a esos MAC se facilitan las transmisiones de paquetes a través de componentes que enlazan varios nodos llamados *switches*.

Usar una capa superior es generalizar el alcance de las comunicaciones de los paquetes, para que no dependa de la topología de la red, y otorgando mucha flexibilidad. La modalidad es que esta tercera capa se encapsula como una carga útil de la capa 2, siempre respetando una estructura donde predomina un encabezado para cada capa. El nombre de IP (*Internet Protocol*) aparece ahora como el ejemplo más popular de esta tercera capa, donde las *direcciones IP* participan como un modo más eficiente de indicar una computadora de destino y origen.

Por último mencionamos la capa de transporte, responsable de comunicar datos directamente entre programas corriendo internamente en las computadoras a través de puertos (pueden haber varios programas ejecutándose a la vez). Según la aplicación, existen protocolos para la capa de transporte de distinta complejidad. TCP (*transmision control protocol*) es el más conocido por garantizar la entrega de los datos a través de redes extensas. Sin embargo, en muchas aplicaciones puede ser suficiente trabajar con UDP (*user datagram protocol*) que es más simple de implementar.

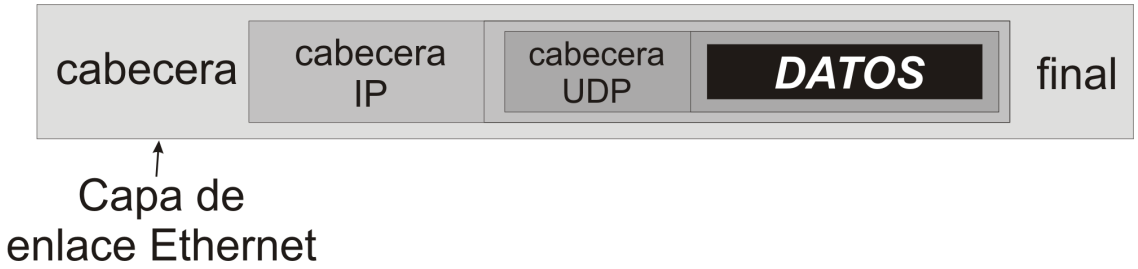


Figura 3.15 Aplicación OSI en las transmisiones Ethernet / IP / UDP

Las conexiones ethernet tipo full-duplex aceptan transferencias efectivas de 10 y 100 Mbps en ambos sentidos. Más recientes y cada vez más accesibles son las conexiones a 1 Gigabps.

PCI

El estándar PCI (*Peripheral Component Interconnect*) define un bus paralelo de comunicaciones del cual pueden conectarse dos partes en una comunicación simple, o varios periféricos como ocurre en el bus de las computadoras modernas de propósitos generales. Los periféricos pueden tener conexiones del tipo plana (soldados directamente al bus) o tipo sócalo (tarjetas de expasión). El estándar PCI fue creado por Intel en 1993, su bus es de 32 o 64 bits y su ancho de banda puede llegar a 133 MBps con un reloj de 33.33 MHz. En 2004 se presentó una versión moderna llamada PCI Express.

USB

USB (*Universal Serial Bus*) es un estándar de bus serie creado en 1996 para comunicar dispositivos. Tiene capacidad de controlar hasta 127 elementos por controlador. Su ancho de banda puede ser 1.5 MBps ó 60 MBps.

Facilita las capacidades de conexión por tener zócalos estandarizados y nuevas conexiones no requieren reiniciar una computadora.

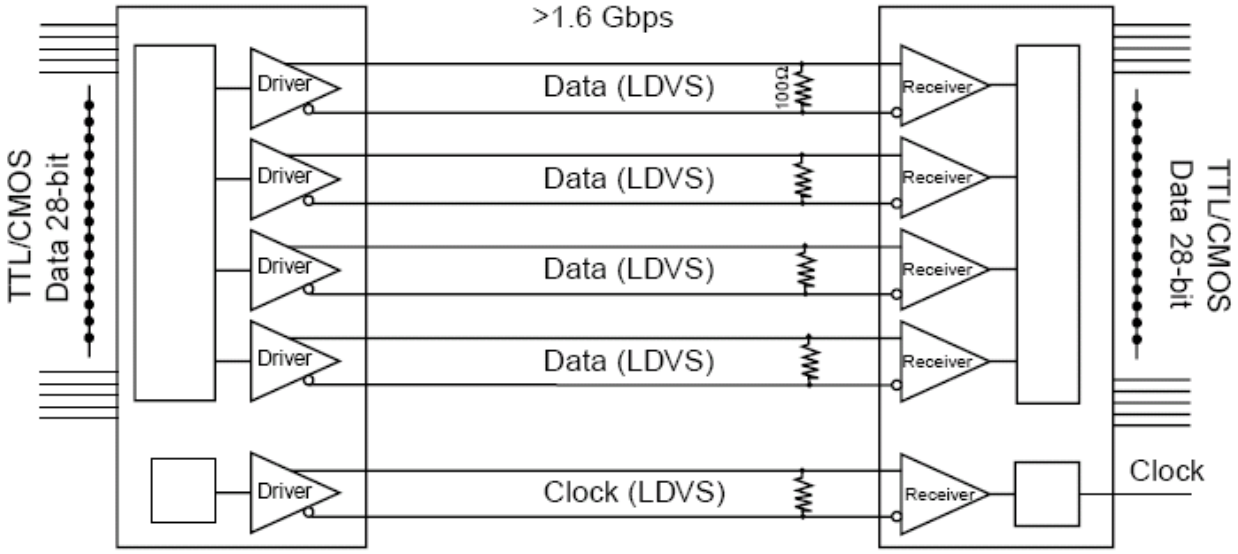


Figura 3.15 Bus de transferencia por LVDS

Camera Link - LVDS

Un grupo de estándares importante pertenecen a las llamadas señales diferenciales de bajo voltaje (LVDS), que incluyen a los casos TIA/EIA-644 y TIA/EIA-899. El primero esta presente en la transmisión de video de muchas cámaras con salida digital. Permite transferir por un canal (dos cables) hasta 1.923 Gbps, con bajos consumos de potencia y cableado simple. Las líneas de transmisión diferencial tienen terminaciones de 100 Ω y la alternancia nominal de la señal es de 350 mV diferenciales.

Camera Link es una interfase para aplicaciones de visión. El mercado de video digital industrial y científico encontró en éste estándar unificación y simplificaciones como

ser en la producción de conectores y cables. Gran parte de las conexiones con altos flujos entre cámaras digitales y receptores de cuadros (*frame grabbers*) se diseñan a partir de Camera Link. Su transmisión se realiza con un bus paralelo de estándar eléctrico LVDS, por el cual viajan datos, una señal de reloj y señales de control.

3.4.5 Transmisión de video

Desde la invención de la televisión muchos estándares de video se desarrollaron, y con la llegada del video digital otros tantos formatos y estándares. Muchas cámaras modernas de video, con sensores CCD o CMOS, poseen salida de video digital y/o analógica. Las digitales presentan beneficios por ser robustas al ruido que suele incorporarse en una transmisión eléctrica. Sin embargo, ambos tipos de cámaras son útiles para PIV. Es común encontrar un mecanismo de control por RS-232 para la configuración remota de las cámaras, y en el caso de Camera Link a través de cables en el bus LVDS.

Son frecuentes los estándares NTSC, PAL y SECAM en las salidas analógicas, en trasmisiones tipo S-Video, Compuesta o de componentes RGB; y el Camera Link como salida digital. Existe un estándar de video digital especificado como ITU-R BT.601/605 usado por muchos circuitos integrados digitalizadores de video (por ejemplo varios productos de Analog Devices) y el estándar tiene capacidad para formatos populares como NTSC, PAL, etc.

Los sistemas de conversión analógica-digital para señales video en general tienen salidas paralelas bajo un estándar eléctrico típico (TTL, CMOS33, etc). Su configuración puede hacerse con interfases I^2C o SPI, entre otras.

Señalamos algunas características eléctricas de las interfases digitales porque en su mayoría son compatibles con los bancos de estrada/salida de las FPGA.

Señal de Video

Una señal de video se compone de cuadros transferidos a una frecuencia que depende del estándar o formato principal del video. En el caso de NTSC son 30 cuadros por segundo y en PAL 25 cuadros por segundo. En estos estándares se divide cada cuadro, dentro de la cámara, en dos campos entrelazados (impar y par) para su transmisión. En el punto de llegada los cuadros se componen nuevamente. Al final de cada cuadro o campo se introduce en la señal una indicación de sincronismo vertical (*vertical sync*).

Un cuadro de video esta compuesto por líneas, 525 en NTSC y 625 en PAL. Al final de cada línea, se genera una señal de sincronismo horizontal (*horizontal sync*). De esa manera el receptor puede saber que parte de un cuadro esta llegando por el canal de comunicación.

En las transmisiones de video el tiempo puede dividirse en tiempos vacíos (blanking) o momentos donde la señal no da información de algún píxel, y tiempos activos.

En una transmisión digital cada línea, correspondiente a un tiempo vacío o activo, posee además indicaciones de comienzo de línea (*SAV*) y de fin de línea (*EAV*). El tiempo entre el pulso EAV y SAV no posee información de píxeles. Los sistemas de captura de video llaman señal de referencia temporal (*TRS*) a un pulso digital que sintetizan cada vez que se confirma la llegada de un SAV o un EAV. Los píxeles activos que se transmiten por línea en NTSC son 720, y el número de líneas por cuadro 525.

Existen varios modos de representación de colores e intensidad de luz en un cuadro. La Figura 3.16 compara el caso RGB (rojo verde azul) con un formato de intensidad-color (YCbCr). El caso YCbCr 4:2:2 (descrito en el estándar ITU-R BT.601) se creó para sistemas de poco ancho de banda y teniendo en cuanta que la percepción visual humana es más sensible a los cambios de intensidad de luz que a las variaciones de color. Para transmitir los datos como una serie de valores, se crean pares de intensidad (Y) – color (C), intercalando la información de rojo (Cr) y azul (Cb) como describe el estándar ITU-R BT.605 y mostramos en la Figura 3.17. En el Anexo indicamos como se compone todo un cuadro por éste método.

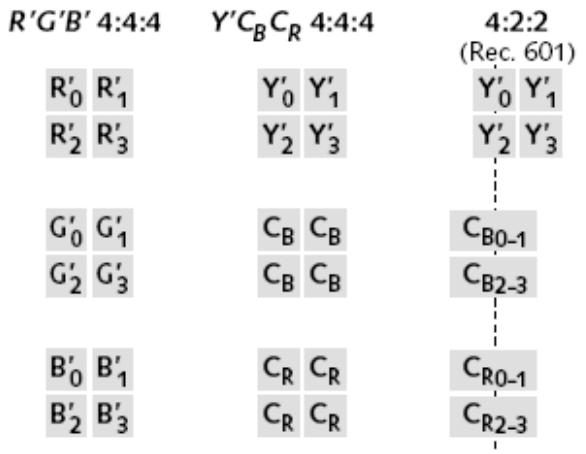


Figura 3.16 Entandar RGB y YCbCb representados en una matriz de 2 x 2 píxeles. El caso de la derecha es de muestreo reducido de color.

i-1: C_B Y C_R Y C_B Y C_R Y C_B Y C_R Y ...
i: C_B Y C_R Y C_B Y C_R Y C_B Y C_R Y ...
i+1: C_B Y C_R Y C_B Y C_R Y C_B Y C_R Y ...

Figura 3.17 Ordenamiento de la información en la transmisión de tres lineas sucesivas de un cuadro con formato YCbCr 4:2:2

3.5 Resumen

El diseño digital permite incorporar en una solución de ingeniería herramientas diversas para el tratamiento de grandes volúmenes de información, tanto en su comunicación como en su procesamiento.

La lógica sincrónica es la base del diseño digital y hoy puede implementarse de manera flexible en dispositivos de lógica programable, siendo las FPGAs un tipo de circuito integrado apropiado para el prototipado de desarrollos.

Un conjunto de estándares y medios de comunicación son comunes en muchas implementaciones, como las usadas en la comunicación entre circuitos integrados, equipos y dispositivos de laboratorio o industriales.

Referencias

[1] Mark Balch, *Complete Digital Design – A Comprehensive Guide to Digital Electronics and Computer System Architecture*, McGraw Hill, 2003

[2] G. Güinchal, *Diseño Digital Utilizando Lógicas Programables*, UTN, Fac. Reg. Bahía Blanca, 2005

[3] J. Deschamps, G. Bioul, G. Sutter, *Synthesis of Arithmetic Circuits – FPGA, ASIC and Embedded Systems*, Wiley, 2006

[4] J. Siman, G. Jaquenod, H. Mascialino, *FPGA-Based Transmite/Receive Distributed Controller for the TR Modules of an L Band Antenna (SAR)*, 2008 IV Southern Conference on Programmable Logic, Bariloche, March 2008

[5] National Semiconductors Corp., *LVDS Owner's Manual* , fourth edition, 2008.

[6] S. Kilts, *Advanced FPGA Design*, Wiley, 2007

[7] U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, Springer, 2001

[8] Recommendation ITU-R BT.470-7, *Conventional Analog Television Systems*, 1998.

[9] Xilinx, Inc, http://www.xilinx.com

[10] Lattice Semiconductor, Corp., http://www.laticesemi.com

[11] Digilent, Inc., Digilent Video Decoder Board (VDEC1) Reference Manual, 2005

[12] D. Dellavale, M. Sonaillon, F. Bonneto, *FPGA Based Multi-Harmonic Control System for Single Bubble Sonoluminescence*, 2008 IV Southern Conference on Programmable Logic, Bariloche, 2008

[13] Camera Link, *Specifications of the Camera Link Interface Standard for Digital Cameras and Frame Grabbers*, 2000

[14] N. Kehtarnavaz, M. Gamadia, *Real-Time Image and Video Processing: From Research to Reality*, Morgan & Claypool, 2006

[15] A.Bovik, *Handbook of Image & Video Processing*, Academic Press, 2000

[16] Xilinx Inc., White Paper 324, *New High Speed Broadcast Video Connectivity Solution (3G) with Low-cost FPGA*, 2007

[17] Xilinx Inc., App. Note 172, *The Design of a Video Capture Board Using the Spartan Series*, 1999

[18] Xilinx Inc., App. Note 132, *Using the Virtex Delay-Locked Loop*, 2006

Capítulo 4

Desarrollo de un sistema de tiempo real para PIV

A continuación exponemos algunas decisiones de diseño basadas en el análisis del problema de velocimetría PIV realizado en el Capítulo 2 del trabajo, y la investigación resumida en el Capítulo 3 sobre potenciales tecnologías de captura, almacenamiento y procesamiento de información de video.

En este capítulo planteamos la estrategia utilizada para la aceleración de un algoritmo PIV orientado a implementarse en tecnología FPGA, haciendo hincapié en una variedad de compromisos que guardan relación con el consumo de área o cantidad de recursos lógicos utilizados y la velocidad de procesamiento.

Con un diseño lógico eficiente pueden conseguirse notables mejoras en la velocidad o capacidad de cálculo del sistema (*throughput*). Explicamos como un manejo óptimo de los datos de una ventana de interrogación permite un máximo aprovechamiento de los multiplicadores y la memoria vinculados al proceso. Sumado a esto, el manejo realizado totalmente por *hardware* de los corrimientos de muestreo en cada ventana de interrogación hace del sistema algo modular y lo convierte en un potente periférico para sistemas embebidos de diverso tamaño. La idea es llevar la paralelización no solo a la aritmética sino también a la administración de la información.

El objetivo es diseñar una arquitectura que permita el funcionamiento sincronizado de los distintos módulos, maximizando la velocidad de procesamiento.

4.1 Algoritmo de PIV 2D-2C (Plano Laser) y su implementación en Lógica Programable

4.1.1 El algoritmo

De los algoritmos de PIV 2D-2C descritos en el Capitulo 3 elegimos implementar el de *correlación cruzada directa* versión *espacial* (*cc-d-espacial*), con un manejo de datos del tipo *correspondencia de patrón de imágenes* (PIPM). El criterio responde en primer lugar a las numerosas ventajas comentadas anteriormente de la correlación cruzada frente a los otros algoritmos (PTV, auto-correlación); también tiene en cuenta las dificultades de aplicar divisiones en la lógica programable (en el Capítulo 3 hacemos consideraciones de diseño en los sistemas digitales empleados) y el importante incremento del número de operaciones

que requieren otros algoritmos como el de correlación cruzada normalizada. Por razones similares preferimos no implementar alguna de las versiones de los algoritmos basadas en la transformada de Fourier, basándonos en el factor de mérito f_O presentado en el Capítulo 2.

Siguiendo el diagrama del Anexo A, en primer lugar requerimos dos imágenes de partículas A y B (cuadros secuenciales de video) de igual tamaño obtenidas de la cámara. Para procesar estas imágenes debemos dividirlas en ventanas de interrogación de tamaño menor. El tamaño de cada ventana depende de la velocidad del fluido y el intervalo temporal Δt entre la iluminación de las imágenes A y B. Con cada ventana asociamos las AreaA y AreaB, que son porciones de las respectivas imágenes A y B. Para simplificar la siguiente explicación suponemos imágenes y ventanas cuadradas, aunque no siempre es así y el sistema comentado puede adaptare. A continuación, cuando indiquemos dimensiones de las imágenes estas se referirán a unidades de *píxeles* (puntos o elementos básicos de una imagen digital) salvo aclaración. En general, cuando se indiquen anchos de palabras en un registro, lugar de memoria o bus de datos, corresponden a unidades de bits.

Los tamaños de A y B: $N \times N$
(en general 1024 x 1024 para cámaras digitales, y 720 x 487 para cámaras analógicas)
El tamaño de subAreaA: $n \times n$
El tamaño de AreaB: $m \times m$
Donde respetamos que

$$N \geq m > n\,.$$

Para hallar el mejor ajuste entre subAreaA y AreaB reescribimos la formulación de *correlación cruzada* que nos interesa, y viene dada por

$$R_{AB}(x, y) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} A(i, j)\ B(i + x, j + y)\,. \tag{4.1}$$

Llamamos a (x, y) *corrimiento del muestreo*. Para cada vector (x, y) puede calcularse un valor $R_{AB}(x, y)$ donde el *rango de corrimiento* es

$$\left(-\frac{m-n}{2} \leq x \leq \frac{m-n}{2},\ -\frac{m-n}{2} \leq y \leq \frac{m-n}{2} \right).$$

Esto genera un *mapa de correlación* de tamaño $(I_x = m - n + 1) \times (I_y = m - n + 1)$, y la ubicación de la máxima correlación la utilizamos como medida del desplazamiento local de las partículas, y este a su vez para el cálculo del movimiento del fluido usando Δt. La cantidad de corrimientos de muestreo definida en el Capítulo 2 sería $z = I_x \times I_y$. Llamamos subAreaB(x, y) a cada conjunto de *píxeles* del AreaB que participa en el cómputo de la ec.(4.1) para el par (x, y). Suele aplicarse un post-procesamiento sobre el plano de correlación para conseguir una resolución sub-píxel y calcular algún factor de mérito sobre la calidad del resultado [4].

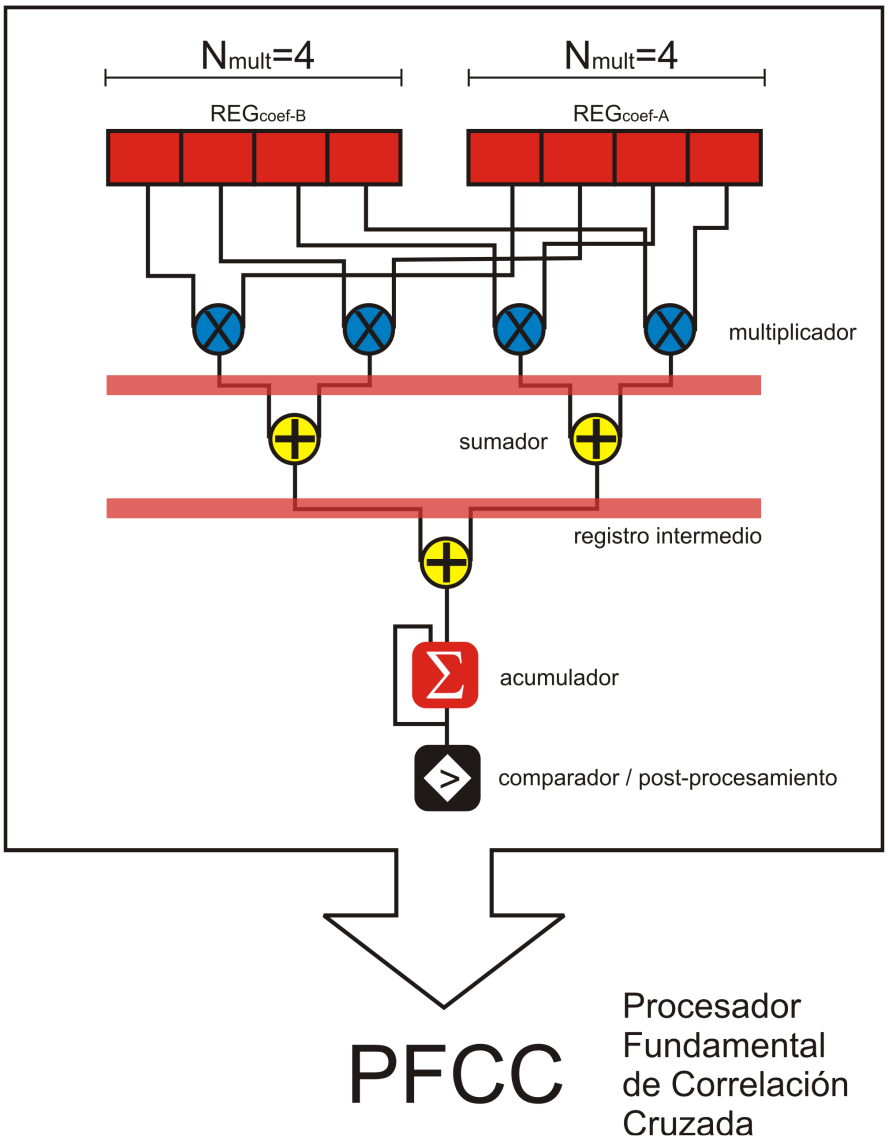


Figura 4.1. Esquema del algebra distribuida que implementamos en el procesador fundamental de correlación cruzada (PFCC). En este caso N_{mult}=4.

4.1.2 Implementación en hardware reprogramable

Destacamos tres cuestiones a considerar en la implementación del algoritmo PIV mediante hardware reprogramable tipo FPGA (Field Programmable Gate Array):

a) Administración de los datos de las imágenes y las ventanas de interrogación. Esto incluye el uso de las memorias externas e internas del chip FPGA.

b) Paralelización de las ecuaciones del algoritmo.

c) Implementación concurrente y/o secuencial de la parte lógica del algoritmo.

Los compromisos en el diseño -que más adelante detallamos- consideran estos tres aspectos y la optimización de los recursos del chip. Si bien los programas encargados de sintetizar e implementar un diseño de éste tipo realizan optimizaciones muy específicas del tipo de arquitectura FPGA con la que trabajemos (comentadas en el Capítulo 3), el aporte del diseñador es lo que más pesa [9].

Una estrategia muy empleada en aplicaciones de suma de productos (*SOP*) se describe en [1, 8]. Esta consiste en agrupar cierto número de multiplicaciones (N_{mult}) de la ec. (4.1) y ejecutarlas en paralelo. En forma concurrente a estos multiplicadores implementamos sumadores y registros para guardar resultados intermedios –y reducir los retardos en los caminos con lógica combinatoria- (Figura 4.1), teniendo cada etapa el ancho de palabra apropiado para la correspondiente operación binaria: sumar dos valores de igual ancho produce un resultado con un bit extra; multiplicar dos valores de igual ancho produce un resultado con el doble de ancho de palabra a lo sumo. Estos sistemas en cascadas son parte del concepto digital de *pipelining*, que acorta las demoras por propagación de las señales y permite a la lógica en su conjunto ejecutarse a mayor velocidad.

El bloque con el símbolo Σ representa la acción de acumular secuencialmente los valores que resultan de la cascada. De ésta manera la ec. (4.1) podemos reescribirla para representar la implementación parte paralelizada (cómputo de *X* más abajo) y parte secuencial (sumatoria) de siguiente modo:

$$R_{AB}(x,y) = \sum_{\xi=0}^{\frac{n^2}{N_{mult}}-1} X(\xi,x,y)\,.$$

A primer orden el tiempo T_R de cálculo de $R_{AB}(x,y)$ sigue una relación

$$T_R \propto \frac{1}{N_{mult}}\,.$$

Por otro lado, la administración interna de los datos que alimentan a los registros de coeficientes REG_{coef} establece el tiempo T_{REG} para cada carga del par $REG_{coef\text{-}A}$ y $REG_{coef\text{-}B}$. El tiempo mínimo es un ciclo de reloj, o sea $T_{REG} \geq T_{reloj}$.

El número de veces que deben alimentarse los registros del procesador PFCC (figura 4.1) para calcular una correlación R_{AB} depende del tamaño del subArea de interrogación (subAreaA y cada subAreaB poseen 1024 píxeles cada una) y del *ancho de píxeles* W_A (*palabra)* que pueden almacenarse en un lugar de la memoria interna del chip (BRAM por ejemplo) que alberga a dichos patrones de intensidad. Combinando adecuadamente estos conceptos resulta una cota inferior para el tiempo del cálculo T_R:

$$T_{R_{\min}} = \frac{n^2}{N_{mult} W_A} T_{reloj}\,. \qquad (4.2)$$

El tiempo T_{vi} necesario para procesar todos los corrimientos de muestreo para una *ventana de interrogación* (o sea recorrer con subAreaA todas las posibles subAreaBs dentro de AreaB) es al menos $T_{vi} = T_R \; I_x \; I_y$.

Finalmente la expresión para el tiempo de cómputo T_{AB} de un par completo de cuadros *A*, *B*:

$$T_{AB} = \left(\frac{n^2 I_x I_y}{N_{mult} W_A} + N_{TRANS} \right) \frac{T_{reloj} N_{vi}}{N_{PFCC}}, \tag{4.3}$$

donde N_{vi} es el número de ventanas de interrogación por par de cuadros de video, N_{PFCC} es el número de procesadores PFCC trabajando en paralelo y N_{TRANS} el número de ciclos efectivos de reloj asociado a las pausas en el proceso por las transferencias de datos hacia la memoria interna, entre la memoria interna y los PFCCs, y detenimientos utilizados en sincronizaciones. A estas sincronizaciones asociamos el acumulador y comparador de cada PFCC y los cambios en la máquina de estado que coordina la carga de nuevos corrimientos de muestreo en un PFCC. N_{TRANS} no es una función sencilla y esta vinculada a la arquitectura del diseño. En general

$$N_{vi} = (N / m)^2 f,$$

donde *f* es el factor de solapamiento de las ventanas de interrogación dentro de un cuadro. Según [10] la mayoría de los problemas en PIV se resuelven con *f* = 4, o sea 50% de solapamiento.

Un importante objetivo es reducir el tiempo T_{AB} tanto como sea posible, para que el sistema tenga una tasa de procesamiento de imágenes (*throughput*) más alta, lo que involucra menos requerimientos de memoria externa y la convierte en una muy útil herramienta experimental como acelerador del algoritmo de PIV. Además pueden bajarse los retardos al punto de servir el instrumento para mediciones en tiempo real y asistir a sistemas de control realimentados [1, 2].

La ec. (4.3) explica porque el tiempo T_{AB} depende de la arquitectura del HW reconfigurable utilizada (los recursos del chip FPGA), la calidad de la lógica implementada (mejor gestión de los datos y reducción de T_{reloj}) y, obviamente, del tipo de problema físico (que fija el tamaño y cantidad de las ventanas de interrogación).

4.1.3 Módulo de procesamiento PIV: *"PCC Core"*

Para hacer uso eficiente del procesador PFCC requerimos un conductor lógico para los datos. Consideramos necesaria la descripción en hardware de un algoritmo que realice todas las funciones de comunicación y control (*handshaking*) con una interfase genérica e internamente gestione un manejo eficiente de los datos de las ventanas de correlación.

Introducir jerarquías en el diseño es uno de los primeros y más importantes pasos en los desarrollos basados en FPGA [3]. El diagrama en bloques de la Figura 4.2 muestra las partes funcionales para procesar una ventana de interrogación y obtener un vector de movimiento a partir de ella.

Como mostramos, la esencia del bloque PFCC es establecer una aritmética distribuida. Implementar multiplicadores, sumadores, registros intermedios, acumuladores y cierto post-procesamiento en forma óptima.

El bloque *Controlador de Proceso* (CdP) es la parte más crítica del módulo PCC. Contiene la lógica que asiste en la lectura simultánea de las memorias AMem y BMem (las cuales almacenan respectivamente la subAreaA y la AreaB respectivamente), genera señales de sincronización con PFCC, realiza almacenamientos intermedios (*buffering*) y multiplexados para componer adecuadamente los datos del subAreaB a correlacionar (recordar que los corrimientos de muestro generan subAreas a partir del AreaB para correlacionarlas punto a punto con la subAreaA). CdP es responsable maximizar el aprovechamiento de los ciclos de reloj en la provisión de datos a los registros $REG_{coef\text{-}A}$ y $REG_{coef\text{-}B}$ y mantener con un alto factor de carga al procesador PFCC (que repercute directamente en el *throughput*).

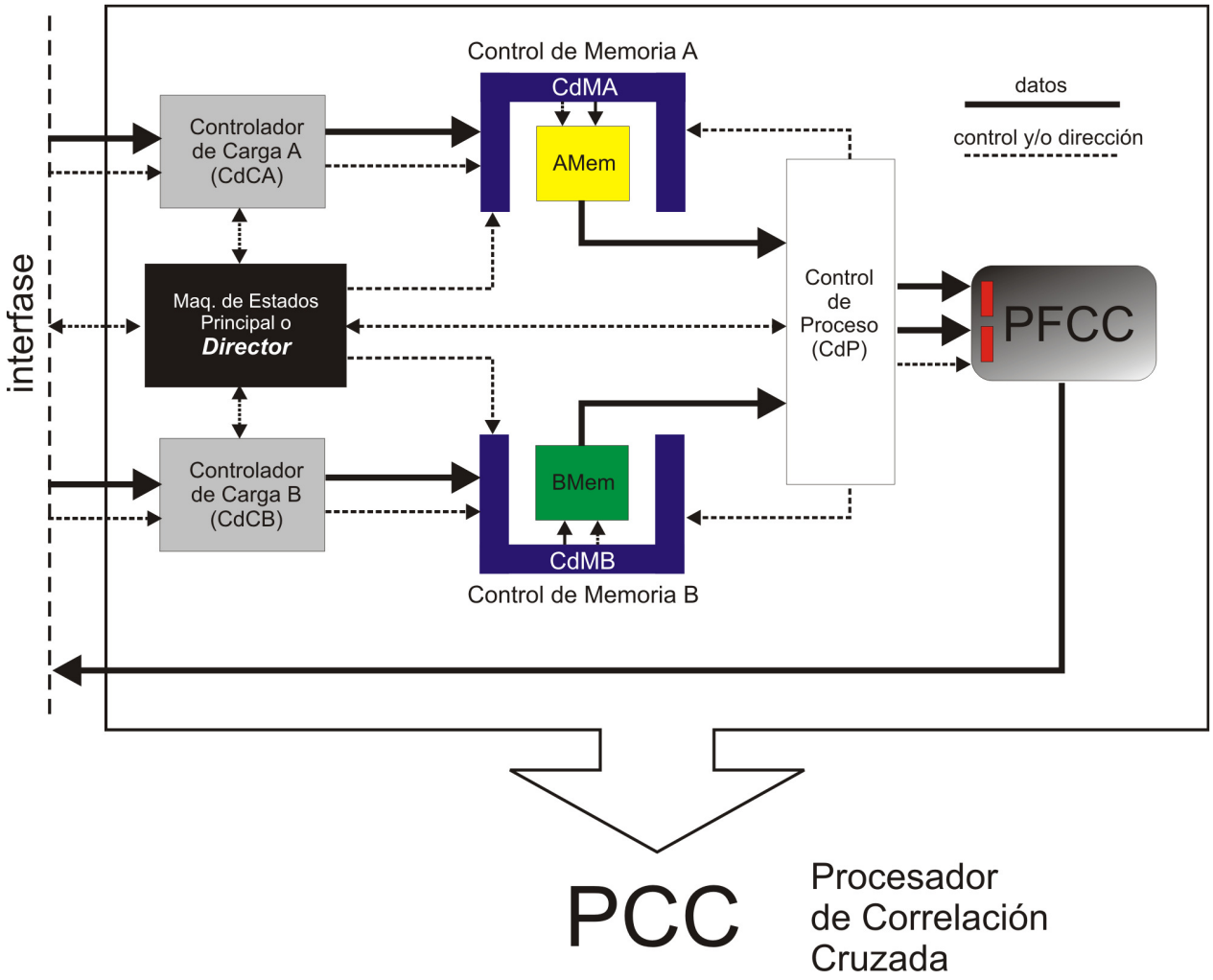


Figura 4.2. Diagrama en bloques del sistema básico procesador de correlación cruzada (PCC)

Separamos la máquina de estados *Director* de CdP por razones de simplificación del código y por la generalidad de las tareas del bloque Director ante diversas estrategias de

manejo de datos que pudiesen implementarse en CdP. Si pensamos en diversos posibles mecanismos para descargar los corrimientos de muestreo (como describimos más adelante), mantener inalterada la estructura de Director agiliza los nuevos diseños a evaluar.

Los *Controladores de Carga* CdCA y CdCB se encargan de la adaptación de los datos de entrada y la ejecución de contadores necesarios en la carga de las memorias AMem y BMem respectivamente. Además sincronizan la lectura de nuevos datos y escritura de las memorias a partir de la recepción y generación de habilitaciones de escritura.

Los *Controladores de Memoria* CdMA y CdMB arbitran los accesos a los puertos de entrada en las memorias (datos, dirección, habilitaciones) demandados tanto por CdP como por los CdCA y CdCB.

Nuevamente, haber separado los bloques recién descritos aporta flexibilidad al diseño y facilita la reutilización cada parte del código.

4.1.4 Consideraciones para un óptimo y flexible uso de recursos

Un análisis de las principales necesidades del proyecto, como son la memoria interna y los multiplicadores, y sus posibles implementaciones, requiere el estudio de variantes para identificar los puntos críticos del sistema desarrollado.

Es necesario dejar en evidencia la vinculación directa que existe entre la organización de los espacios de memoria interna (tamaño de las palabras o puertos) y el número de multiplicadores N_{mult} implementados en PFCC. De un primer análisis señalamos que grandes simplificaciones se consiguen si igualamos la cantidad de píxeles en una palabra de memoria con el número de multiplicadores. De esta manera, al leer los datos de una posición de memoria, es posible alimentar todos los multiplicadores en un solo ciclo de reloj. Este modo de implementación nos permite conseguir un elevado factor de carga del módulo de procesamiento PFCC.

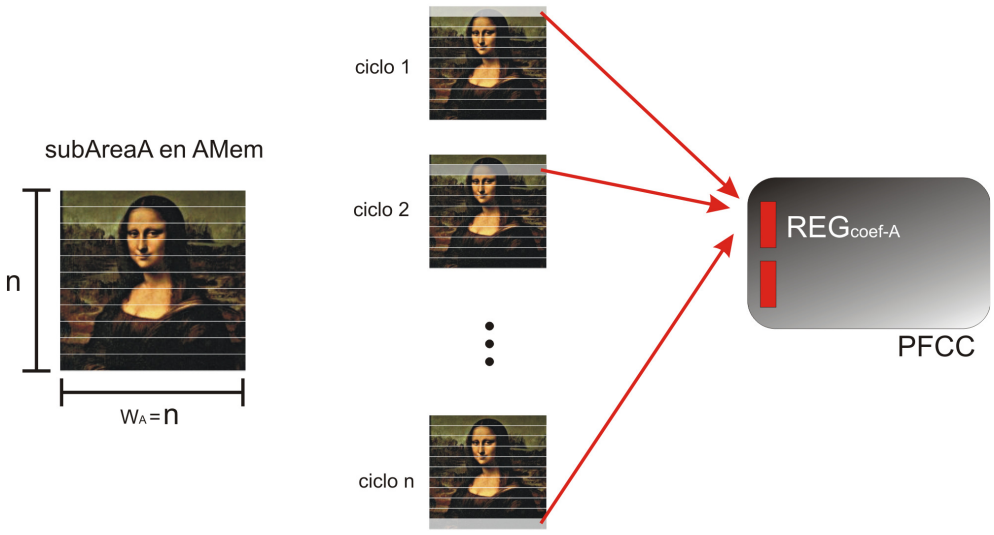


Figura 4.3. Caso N_{mult}=n

En la Figura 4.3 mostramos un posible modo de alimentar a los registros de PFCC, en cada ciclo de reloj con una línea horizontal completa de píxeles alineados del subAreaA. En este caso $N_{mult} = n$.

A mayor ancho de palabra en memoria equivale a decir más multiplicaciones por unidad de tiempo (y ello significa más velocidad), pero implementar dichas memorias de gran ancho de palabra es un inconveniente. No cualquier ancho es aceptado por las memorias RAM internas dedicadas (Anexo C) y deberá encontrarse un tamaño de palabra adecuado para que el diseño haga uso óptimo de sus recursos de memoria.

Si el ancho n del subAreaA es demasiado grande, el esquema de la Figura 4.3 podría verse limitado por la capacidad de almacenamiento e implementación lógica del chip FPGA. Las interconexiones complejas de señales de gran ancho de palabra requiere mayor cantidad de recursos, y las memorias de gran ancho de palabra requieren instanciar varias memorias internas (memorias en bloque o BRAMs) en serie. Tal implementación resultaría de baja eficiencia e inflexible.

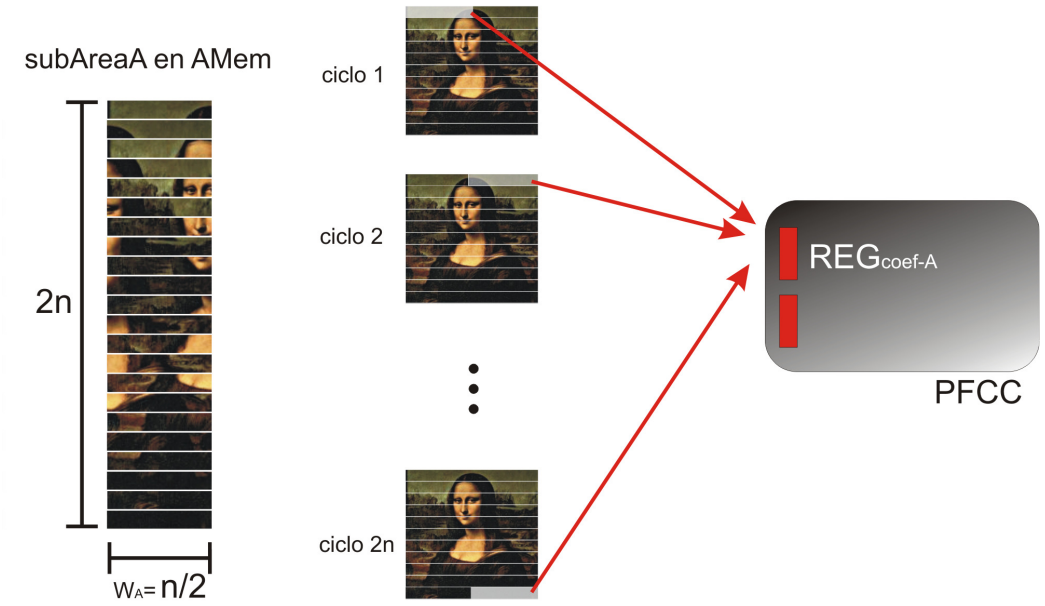


Figura 4.4. Caso $N_{mult}=n/2$

Por otro lado cada multiplicador demanda importantes recursos. A mayor número de multiplicadores más extensa la estructura en cascada de sumadores y registros intermedios en el PFCC (Figura 4.1). Podemos implementar de dos maneras multiplicadores en el chip FPGA: la primera es designando multiplicadores dedicados, la segunda es sintetizar multiplicadores embebidos en la lógica configurable. Existen casos donde el número de multiplicadores dedicados es lo que limita a un diseño; entonces podemos optar por implementarlos en las celdas lógicas generales. En este trabajo consideramos para el módulo PCC la implementación optativa de cada tipo de multiplicador.

En síntesis, el valor de N_{mult} es muy importante y su apropiada elección requiere conocer las particularidades de la arquitectura y disponibilidad de recursos del chip FPGA destino del diseño.

Si $N_{mult} \neq n$, el esquema de la Figura 4.3 debe replantearse. En la Figura 4.4 muestramos el caso $N_{mult} = n/2$, que según la ec. (4.2) resulta en un procesador PCC más lento que el de la Figura 4.3.

Opuestamente, con una FPGA de mayores recursos podríamos hacer que N_{mult} sea un múltiplo entero de n, y PCC más rápido. Desde los comienzos de éste trabajo pensamos en una aplicabilidad genérica del módulo, teniendo en mente las arquitecturas Spartan-3E (con aprox. 20 multiplicadores dedicados para la version SXC3S500E) y Virtex 4 (aprox. 32 multiplicadores dedicados para el chip V4 FX12 FF668 y unos 150 en versiones más potentes) de Xilinx.

Las memorias en bloque de simple y/o doble puerto de la FPGA sirven de memoria interna para almacenar información de las ventanas de interrogación hasta que calculamos el vector. El gráfico de la Figura 4.5 muestra el número de estas BRAMs necesarias para almacenar una subAreaA, con n = 32 y cada píxel de 8 bits (n_bit = 8), en función de N_{mult} en una arquitectura Xilinx SXC3S500E Spartan 3E (según Anexo C). Las BRAMs disponibles podemos configurarlas con un ancho de palabra máximo de 32 bits (con 512 palabras que hacen 16384 bits disponibles).

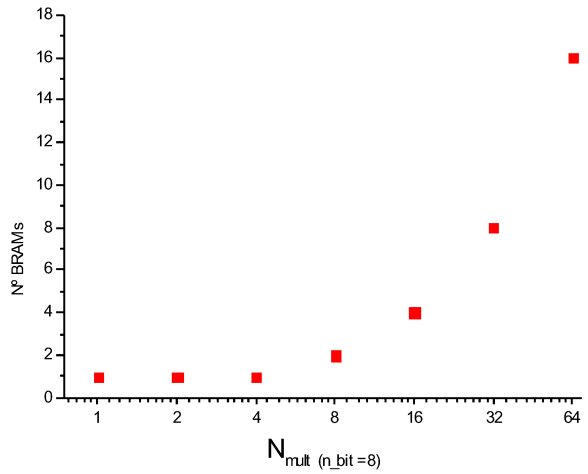


Figura 4.5. Número de bloques de memoria a implementar en función del número de multiplicadores en PFCC. El ancho de palabra de BRAMs es 32 [bits]

Solo destacamos los casos en que N_{mult} es potencia de 2 y podemos efectivamente aplicar el esquema en cascada de la Figura 4.1.

4.1.5 Requerimientos funcionales del Controlador de Proceso

El módulo CdP, entre otras cosas, administra la transacción de datos desde la memoria AMem y permite el acceso a cada píxel en el momento adecuado. En particular, la lectura de AMem y escritura en el registro $REG_{coef\text{-}A}$ es una tarea sencilla. Para todos los

corrimientos de muestreo accedemos a los datos del mismo modo: secuencialmente desde el primer al último elemento.

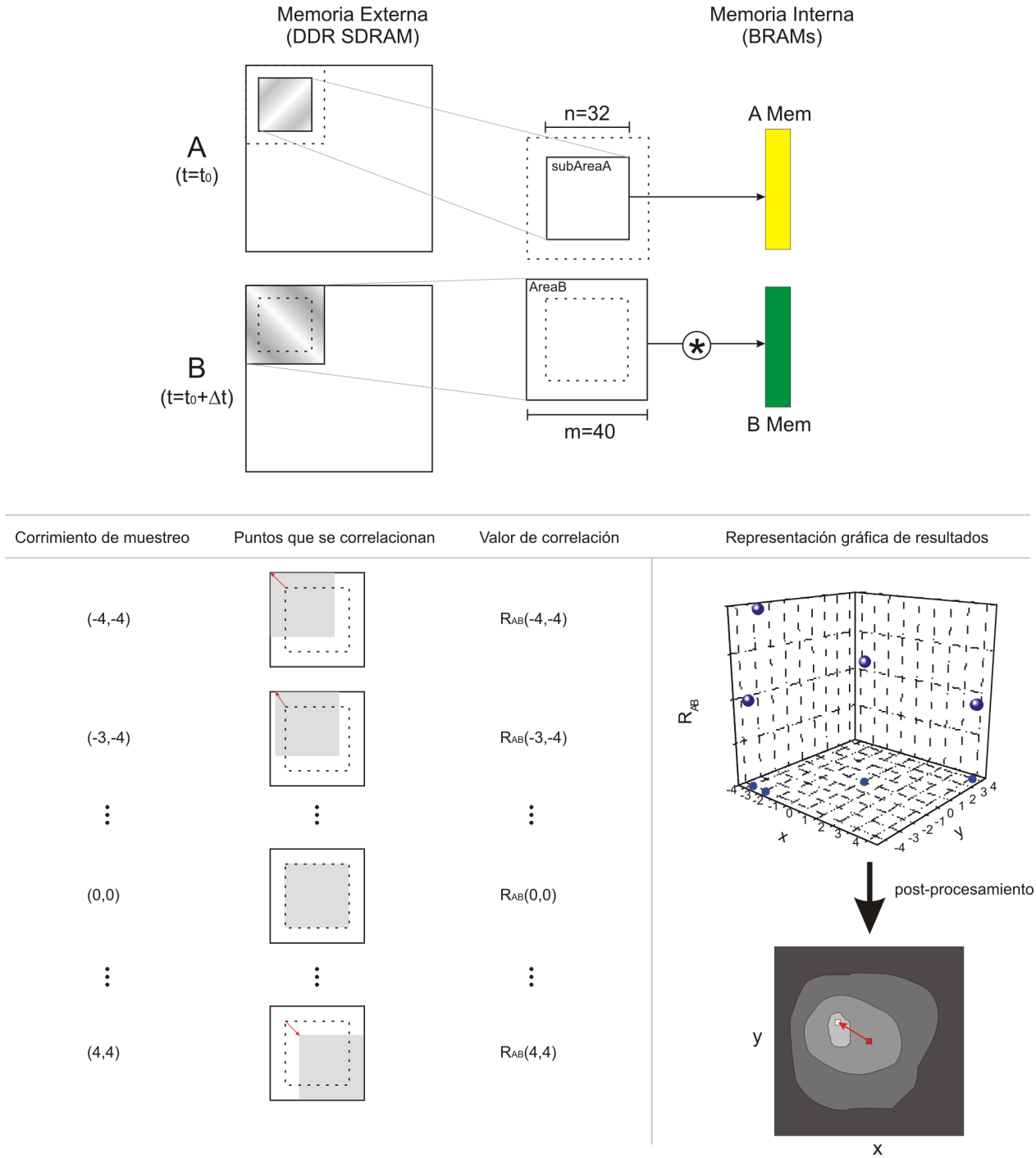


*Figura 4.6. Principales funciones a desempeñar dentro del procesador PCC. Resaltamos los corrimientos de muestreo, que dificultan las tareas del módulo CdP para generar las subAreasB. Tras la comparación de los valores de correlación y un post-procesamiento el vector estimador del movimiento de las partículas. El * representa el bloque lógico que implementa alguno de los diferentes criterios para cargar el AreaB en la memoria interna.*

En la Figura 4.6 esquematizamos una de las funciones del módulo CdP. Dado un corrimiento de muestreo (x ,y), CdP reinicia el acumulador del procesador PFCC y luego transfiere organizadamente los correspondientes píxeles de la subAreaA y la subAreaB a los registros REG_{coef} con el mejor factor de carga posible. Administra los tiempos para habilitar y eventualmente finalizar la acumulación de valores en PFCC. Tras el fin de la acumulación habilita las funciones del comparador/post-procesador.

La *primera condición* para conseguir el más alto factor de carga de PFCC es suministrar sin interrupciones todos los datos de subAreaA y subAreaB en (n^2 / W_A) ciclos de reloj (las unidades de W_A son [píxeles]). Esta cantidad multiplicada por el periodo de reloj es el tiempo mínimo para acceder a todos los sitios de memoria que almacenan la subAreaA (Figuras 4.3 y 4.4).

La *segunda condición* es la rápida disponibilidad de los datos del subAreaB para un nuevo corrimiento de muestreo.

La carga del AreaB en la memoria interna podemos implementarla de diferentes maneras. Un método se describe en [1] y consiste en fijar las dimensiones de BMem idénticas a las de AMem (ancho de píxeles y capacidad de palabras, como en las Figuras 4.3 y 4.4), y para cada corrimiento de muestreo se recarga BMem con la subAreaB correspondiente. Esto significa que necesitamos tantas transacciones entre la memoria externa y la interna BMem como corrimientos de muestreo existan (I_x x I_y = 81 en el caso de la Figura 4.6). La principal ventaja de esta estrategia es lo simple del diseño lógico que descarga los datos a PFCC, ya que tanto la lectura de AMem como de BMem es secuencial y de simultáneo acceso a posiciones iguales en cada memoria interna (con un solo contador se actualizarían las direcciónes de lectura).

La desventaja de este esquema es la necesidad de realizar numerosas transferencias de datos desde la memoria externa, cuyo tiempo de acceso es mucho mayor al tiempo de acceso de las memorias internas. Estas demoras pueden compensarse usando memorias internas intermedias (*pipelining*), por ejemplo duplicando BMem. Esta modificación permite procesar una memoria mientras la otra se carga con nuevos datos del exterior. Luego, un sistema multiplexor invierte cíclicamente sus roles. En síntesis, el costo de una mayor velocidad de procesamiento y simplificación lógica lo pagamos con más área de memoria interna y otras dificultades que pondremos de manifiesto más adelante.

Versión	Memoria BMem	**Funciones de CdP**			Solicitud de memoria interna	Velocidad
		Multiplexado	Almacenamiento (buffering)			
			simple	avanzado		
1	W_B=m	✔			Alta	Alta
2	W_B<m	✔	✔		Media-baja	Media-Baja
3	W_B<m	✔		✔	Media-baja	Media-Baja
4	W_B<m	✔	✔	✔	Baja	Media-Alta
5	W_B>m	✔			Alta	Muy Alta

Figura 4.7. Versiones de un método que requiera pocas transacciones entre las memorias externas e internas.

Otra manera de atacar la *segunda condición* del factor de carga es disponer de toda el AreaB en BMem e implementar en el controlador CdP almacenamientos intermedios y continuos multiplexados de datos guardados en BMem. Esto dificulta el diseño de CdP pero reduce los accesos a la memoria externa y optimiza el uso de la memoria interna. Con un somero análisis de esta idea descubrimos que a menor longitud de palabra W_B de BMem la complejidad de la lógica de CdP crece. En la Figura 4.7 resumimos algunas posibles versiones de éste método que requiere pocas transacciones y explicamos en la próxima sección.

4.2 Estrategias de optimización

4.2.1 Administración de datos

A continuación describimos el algoritmo implementado en el Controlador de Proceso (CdP). Haremos mención a pautas mínimas que rigen a un diseño digital sincrónico (descritas en el capítulo anterior, basadas en [5] y [6]) y como las utilizamos en la descripción del HW.

A continuación analizamos la distribución elegida del AreaB en la memoria interna BMem, la cual configuramos para un ancho de palabra de 32 bits (W_B = 4 [píxels] de 8 bits cada uno). Por cuestiones ya analizadas también eligimos el ancho de AMem como W_A = 4, y con esto quedó definido el número de multiplicadores: N_{mult} = 4 (de 8 x 8 bits cada uno). Diseñamos el sistema para tratar áreas cuadradas con valores de n=32 y m=40 (81 corrimientos de muestreo).

Luego debemos establecer el modo de lectura de la memoria BMem, que almacena el Area B, para conseguir alimentar a PFCC con los píxeles asociados a un corrimiento de muestreo, o sea los píxeles de la respectiva *subAreaB*.

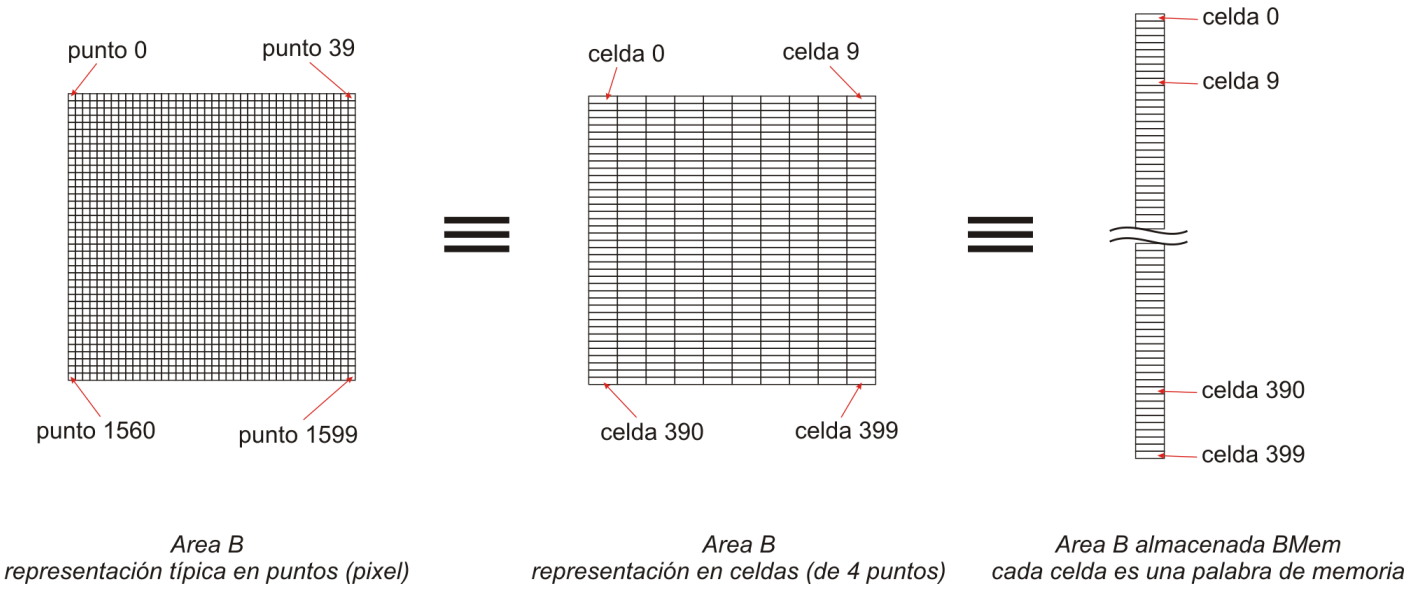
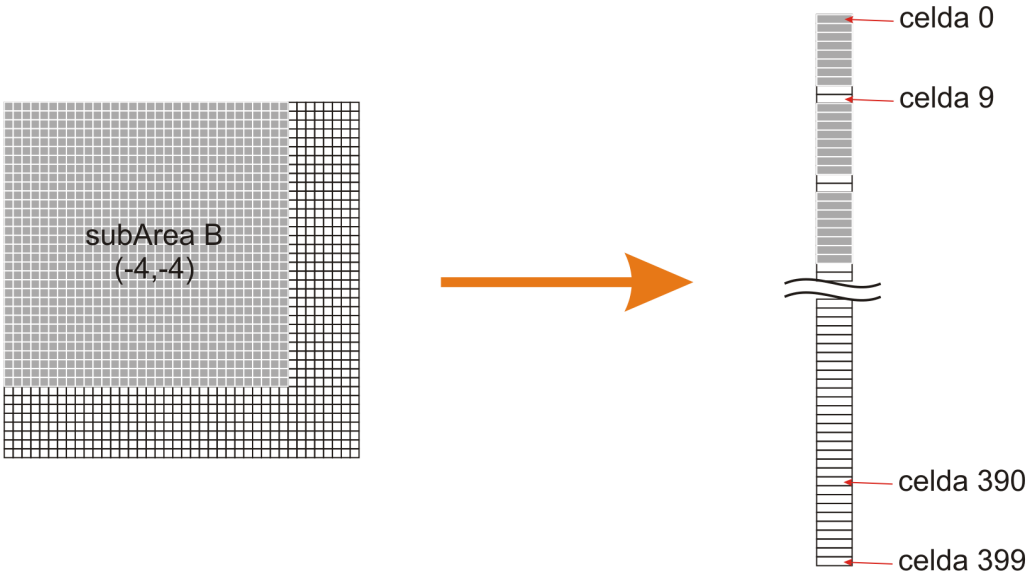


Figura 4.8. Organización en celdas de 4 píxeles de los datos del Area B en la memoria interna (BRAM) BMem

La Figura 4.8 muestra como distribuimos los 1600 píxeles del Area B en BMem. Definimos una celda como el conjunto de cuatro píxeles. Las ordenamos en BMem según la numeración mostrada en la Figura 4.8, ordenadas de izquierda a derecha y de arriba hacia abajo. En síntesis, los 1600 píxeles del Area B los configuramos en un arreglo de 400 celdas y cargamos las mismas ordenadamente en 400 lugares continuos de la memoria BMem.

Supongamos ahora que vamos a leer todos los píxeles asociados al primer corrimiento de muestreo *(x, y)* = (-4, -4) desde BMem. Para este caso la Figura 4.9 facilita apreciar la organización de los píxeles de interés en la AreaB y aquellos en BMem. Los píxeles que destacamos son 1024, la misma cantidad de píxeles que existen en total almacenados en AMem y con los cuales debemos correlacionar para calcular de R_{AB}(-4, -4).

Para este corrimiento de muestreo (-4, -4) participan del cálculo 256 celdas tanto de BMem como de AMem. La lógica para leer la subAreaB (-4, -4) y correlacionar esos píxeles con los de la subAreaA requiere de un contador y decisiones condicionales muy simples.



Representación de un corrimiento de muestreo (-4,-4) en Area B

Organización de lo puntos del subArea B (-4,-4) en Bmem

Figura 4.9. Organización de los datos indicando como almacenamos los píxeles de un corrimiento de muestreo en la memoria BMem. Notar el ausente uso parcial de celdas en BMem para este corrimiento.

La Figura 4.10 indica los 27 casos (entre los 81 corrimientos de muestreo posibles), donde la lógica de acceso a BMem es equivalente a la recién descrita: accediendo en BMem a igual número de celdas que en AMem. De ese modo consiguimos leer los 1024 píxeles asociados al corrimiento. Esta situación es la más favorable porque leyendo simultáneamente una posición de AMem y una de BMem en cada ciclo de reloj, consigo en

256 periodos de reloj consecutivos correlacionar los píxeles de cada uno de estos corrimientos. Y esto aprovecha al máximo el acceso a las memorias (solo es posible acceder a una celda por ciclo de reloj en la configuración de puerto simple de las BRAMs) y representa el mayor factor de carga posible para el procesador fundamental PFCC.

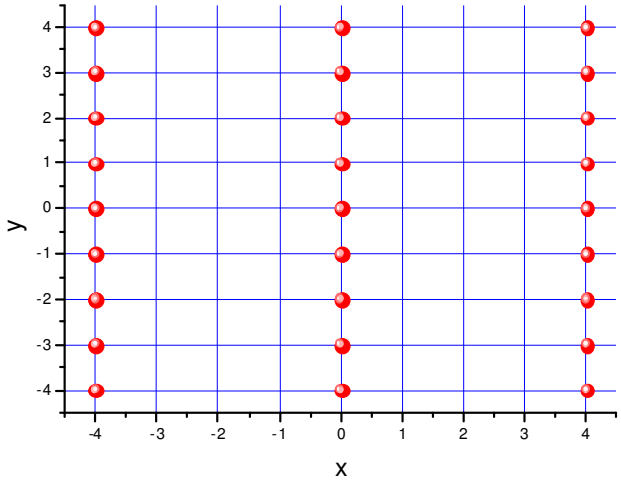
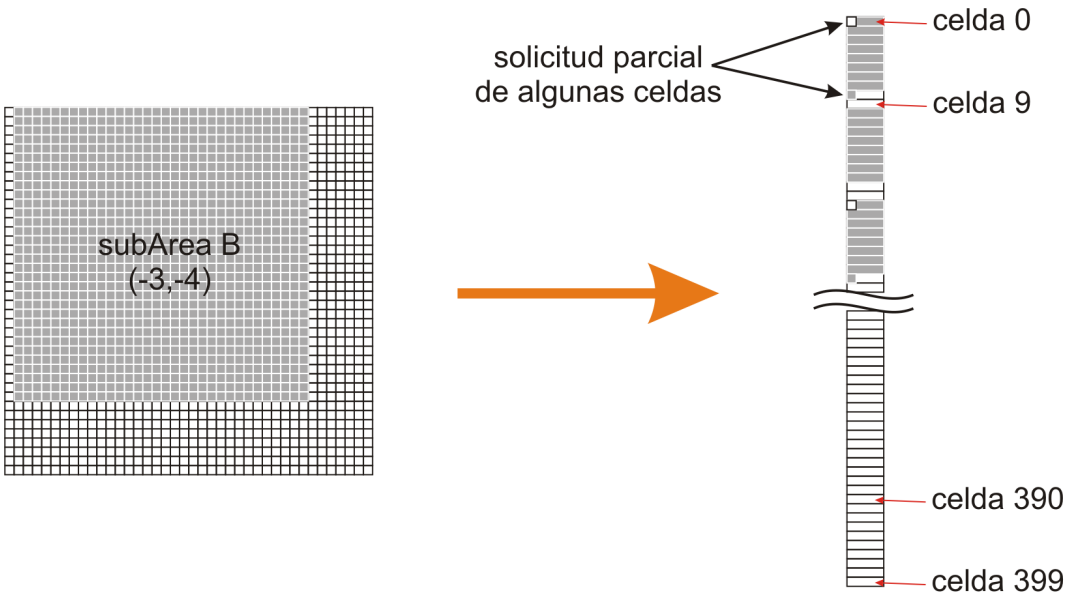


Figura 4.10. Rango de los 81 corrimientos de muestreo. Destacamos con círculos los 27 casos que requieren leer 256 celdas de la memoria BMem para captar los 1024 píxeles a correlacionar de la correspondiente subAreaB.

El acceso a los datos es más complejo para los restantes 54 corrimientos. Comparando la Figura 4.9 con la Figura 4.11 es más evidente ya que existen corrimientos que necesitan acceder al menos 288 veces a BMem y 256 veces a AMem. Los 1024 píxeles que necesitamos de BMem para correlacionar con la subAreaA ahora se ubican en 288 celdas diferentes, en lugar de las 256 de los casos anteriores.

Lo arriba señalado agrega dificultad al algoritmo que controla el acceso a las memorias, ya que para el acceso a los datos requerimos realizar almacenamientos intermedios y multiplexados como indica la Figura 4.7. La conformación de un *bloque* de 4 píxeles para escribir en $REG_{coef\text{-}B}$ requiere la lectura de dos celdas consecutivas. Esta lógica podría implementadase fácilmente mediante software (SW) secuencial, pero realizarla mediante hardware (HW) concurrente y en funcionamiento sincrónico con otros módulos resulta de una complejidad mayor. Los beneficios de aplicar un más esfuerzo en el diseño están en una mayor velocidad de procesamiento del algoritmo y menor uso de recursos del sistema PCC, que repercute fuertemente en las capacidades y flexibilidad generales de nuestro sistema PIV final. La posibilidad de configurar las prestaciones del sistema es algo deseable ya que permite adecuar la implementación a los recursos del dispositivo FPGA disponible.



Representación de un corrimiento de muestreo (-3,-4) en Area B

Organización de lo puntos del subArea B (-3,-4) en Bmem

Figura 4.11. Organización de los datos que indica como almacenamos los píxeles de un corrimiento de muestreo en la memoria BMem. Notar el uso parcial de algunas celdas en BMem para este corrimiento.

Con el objetivo de optimizar recursos lógicos y maximizar la velocidad del algoritmo optamos por implementar la lógica mencionada en una máquina de estados específica. Su apropiado diseño sincrónico, que resulte en un subsistema confiable y capaz de trabajar en altas frecuencias de reloj, debe respetar consignas difundidas en el ambiente del diseño digital. El Capítulo 3 menciona estas cuestiones teóricas que enmarcan la implementar la lógica de un algoritmo arbitrario en un diseño digital sincrónico.

4.2.2 Importancia del *buffering*

Los corrimientos mencionados que requieren leer 256 celdas tanto de AMem como de BMem (Figura 4.10) tienen la ventaja de proveer nuevos bloques de 4 píxeles a los respectivos registros $REG_{coef\text{-}A}$ y $REG_{coef\text{-}B}$ en cada ciclo de reloj.

Los restantes corrimientos necesitan levantar píxeles de 256 celdas de AMem, pero de 288 celdas de BMem. Para cada conjunto de 4 píxeles que conforman un bloque necesitamos píxeles de dos celdas en BMem.

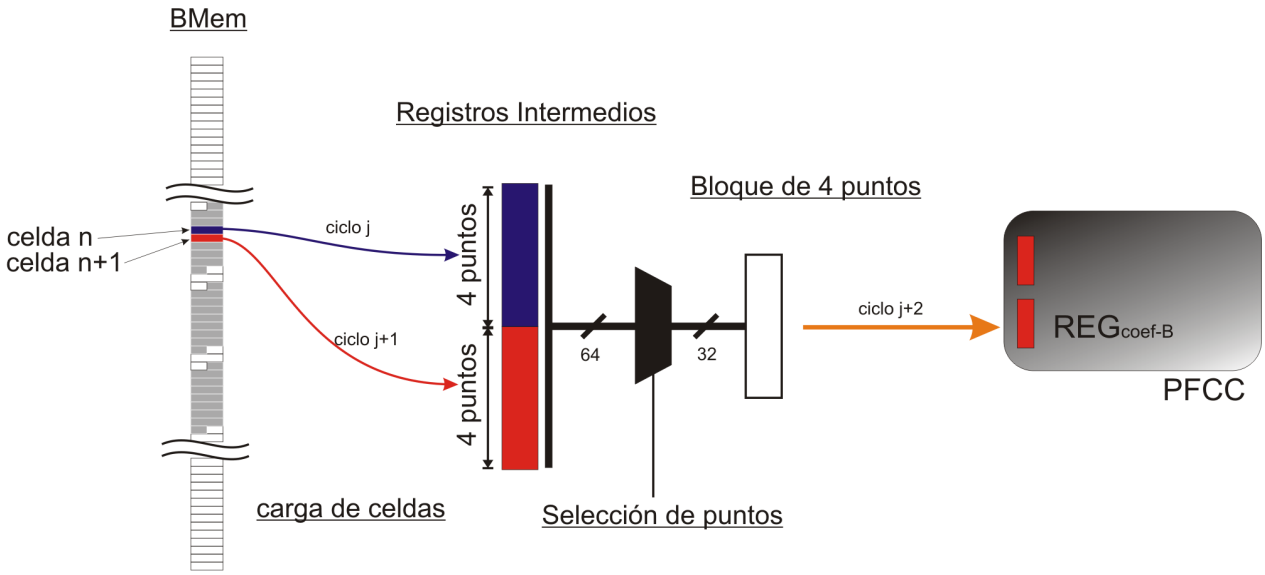


Figura 4.12. Carga de datos desde memoria interna BMem al Procesador Fundamental PFCC. Indicamos el almacenamiento intermedio de datos (buffering) y la selección de un bloque apropiado para correlacionar.

Definido el corrimiento de muestreo, una solución es leer las dos celdas consecutivas de BMem que guardan los 4 píxeles de interés (Figura 4.12) en registros externos a la memoria BMem. Como ahora puedo tener acceso simultáneo a cada bit de esos registros, eligiendo de aquí los que conforman los 4 píxeles del bloque (empleando un multiplexor de 8 píxeles a 4 píxeles, o sea 64 bits a 32 bits), dispongo de los datos para correlacionar en PFCC con el correspondiente bloque de datos de AMem.

El modo trivial de implementar esto es usando dos ciclos de reloj para obtener las celdas n y n+1, y en otro ciclo de reloj capturar con el multiplexor el bloque objetivo y escribirlo en $REG_{coef\text{-}B}$. Con este método los 1024 píxeles de estos corrimientos necesitarían al menos (3 ciclos x 256 bloques) 768 ciclos de reloj para correlacionarse.

Si usamos más registros de almacenamiento intermedio, aprovecharíamos el ciclo de escritura en PFCC para ya empezar a cargas dator desde BMem. Los diseñadores de lógica programable [7] llaman a esto la fórmula del diseño tridimensional:

$$Prestaciones + Tiempo + Memoria = cte. \tag{4.4}$$

Esta relación manifiesta los compromisos y potenciales optimizaciones de todo diseño en FPGA. El esquema de la Figura 4.13 muestra como tras mejorar el almacenamiento intermedio y las lecturas desde BMem el sistema realiza operaciones simultáneas en algunos ciclos de reloj. En este caso cada dos ciclos de reloj entregamos datos nuevos a REG_{coefB}.

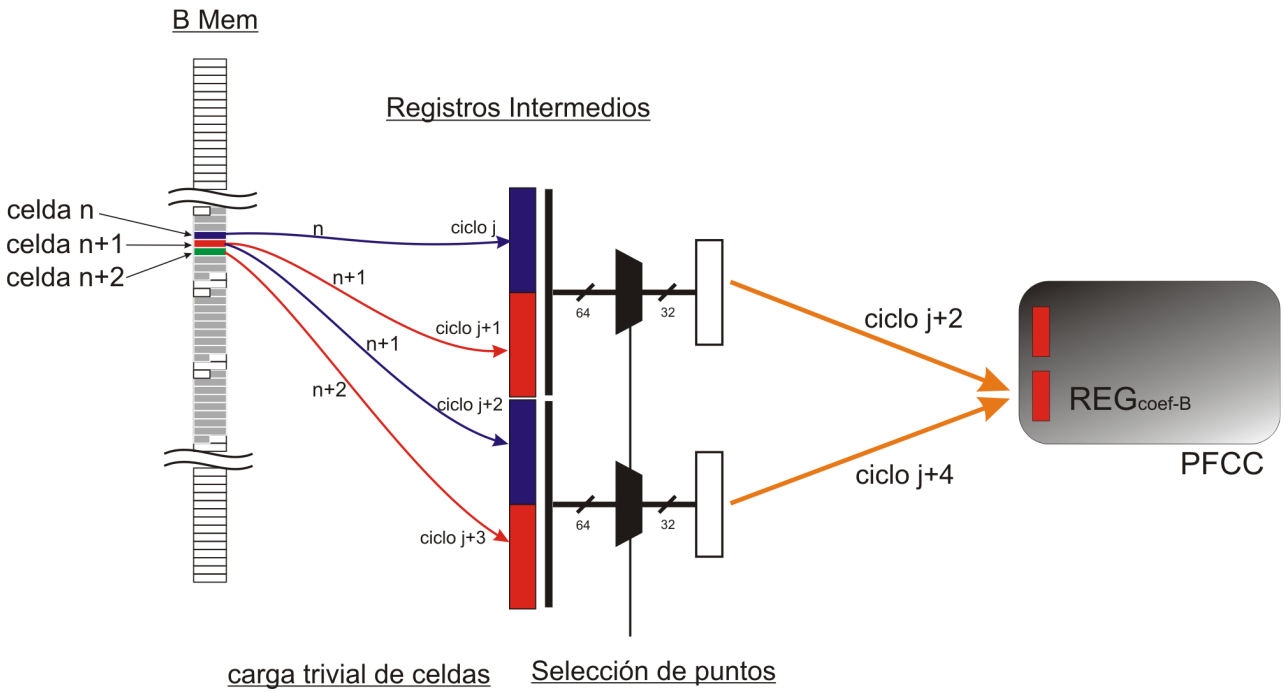


Figura 4.13. Estrategia trivial de paralelización de las tareas indicadas en la Figura 4.12

En este segundo método necesitamos implementar más lógica por tratarse de un algoritmo de manejo de datos más complejo. El beneficio resulta en una reducción a 512 ciclos de reloj del tiempo mínimo necesario para completar esta correlación.

En el caso extremo de utilizar un almacenamiento intermedio de 288 celdas, donde capturar datos leídos desde la memoria BMem y simultáneamente cargar los registros del PFCC a través del multiplexor, conseguiríamos una velocidad mayor aún. Podemos escribir ordenadamente las celdas de BMem que necesito y simultáneamente hacer trabajar al multiplexor, generando bloques que estén disponibles para alimentar a PFCC. De la ec. (4.4) afirmamos que aumentando la cantidad de memoria que interviene en el proceso conseguimos reducir los tiempos del mismo, teniendo ahora la posibilidad de correlacionar los 1024 píxeles del *subAreaB* en al menos 288 ciclos. Esto representa una amplia mejora para el factor de carga y nos acerca al ideal de $n^2 \ / \ N_{mult} = 256$ ciclos de reloj por corrimiento de muestreo.

Es posible conseguir un rendimiento similar de 288 ciclos con solamente 3 celdas de almacenamiento intermedio en lugar de las 288. Para ello debe existir una lógica adicional que recicle el uso de los registros intermedios. Las velocidades son similares pero el uso de recursos menor.

Implementamos el sistema de éste trabajo con cuatro celdas de almacenamiento intermedio a los fines de relajar el análisis de tiempos entre las partes del algoritmo que cumplen tareas independientes de carga de datos desde BMem y descarga de datos hacia PFCC.

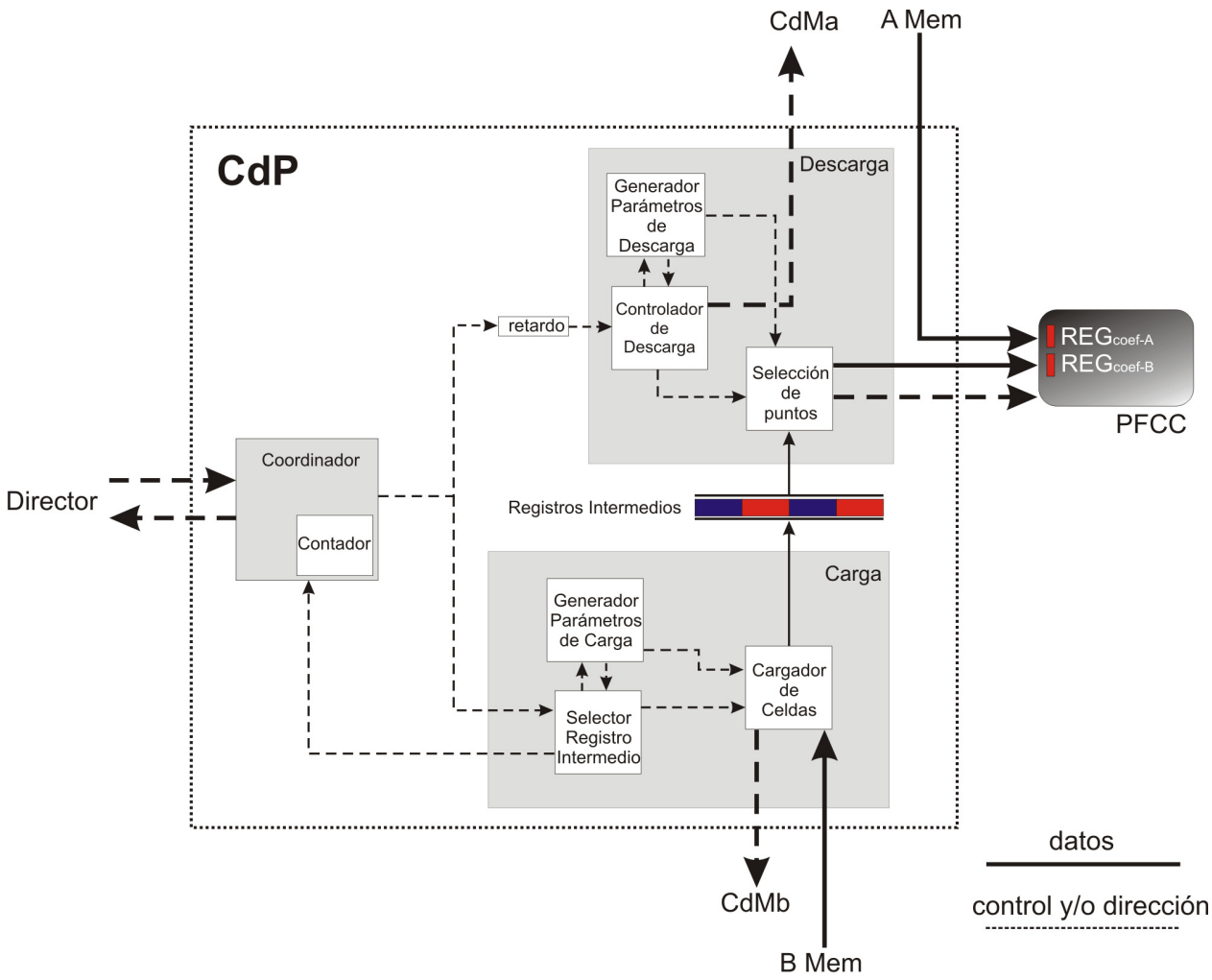


Figura 4.14. Diagrama del sistema Controlador de Proceso CdP. Resaltamos la división del algoritmo en pequeñas máquinas de estado y las lineas de comunicación con otras partes del procesador PCC

4.2.3 Controlador de Proceso

La implementación del algoritmo genérico, que sirve a todos los tipos de corrimientos de muestreo e implementamos en máquinas de estado "independientes" lo representamos con el diagrama en bloques de la Figura 4.14. Mostramos el código VHDL de algunas partes importantes del proyecto en el Anexo D.

Si la arquitectura del chip FPGA utilizado nos permite configurar las memorias internas AMem y BMem del modo doble puerto y existen recursos para duplicar el sistema de control de proceso CdP y el procesador fundamental PFCC (que posee los multiplicadores), podemos doblar la velocidad de procesamiento del módulo PCC. Las memorias de doble puerto admiten lecturas simultáneas de dos palabras cualesquiera por ciclo de reloj.

También podemos pensar en una estrategia de almacenamiento intermedio más avanzada para lograr el nivel óptimo de factor de carga de PFCC, pero la relación costo – beneficio nos muestra su inconveniencia.

4.3 Interfase externa e integración de módulos

4.3.1 Captura de Video

Para este trabajo elegimos utilizar una cámara de video con sensor tipo CCD y salida de video analógico (Sony ExwaveHAD SSC-DC50A) en formato NTSC. En la Figura 4.15 diagramamos como conectamos la cámara mediante un cable tipo S-Video con una placa marca Digilent de conectores apropiados y portadora del chip digitalizador de video Analog Devices Video Decoder ADV 7183B. La salida digital es el estándar YCrCb 4:2:2 y formato NTSC. Encontramos la configuración de los pines del chip FPGA para esta aplicación en la Figura 4.16.

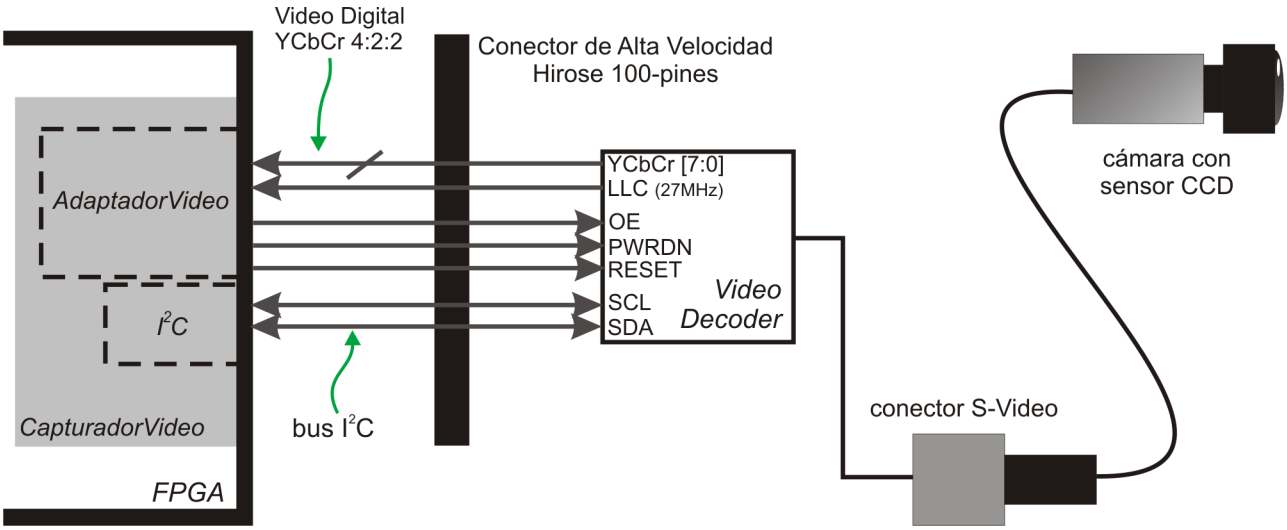


Figura 4.15. Conexiones principales del sistema de captura de video

Del lado receptor de video (FPGA) implementamos dos bloques principales (Figura 5.15). El módulo I^2C es el encargado de realizar la configuración del chip ADV7183B. El módulo AdaptadorVideo controla señales de habilitación del chip externo (OE, RESET, PWRDN) y captura los datos de video que llegan sincrónicos con un reloj de 27MHz (LLC) generado en la placa Digilent.

Componemos a AdaptadorVideo de dos sub-módulos: *Sincronizador* y *ConversorY*. Cuando ingresa la señal de 8 bits *YCbCr_in*[7:0] discriminamos en ConversorY la información de iluminación (Y) del formato YCbCr. Como el muestreo de los colores tiene la mitad de frecuencia que Y (relación 4:2:2), sintetizamos una señal de habilitación de datos (*ceo*) cuya frecuencia es de 13.5 MHz cuando están disponibles nuevos píxeles (línea activa del video).

Internamente el sub-módulo Sincronizador extrae la información del sincronismo de los datos y controla el nivel lógico de las señales de *línea activa* H_0 (*Horizontal Sync*), *habilitación vertical* V_0 (*Vertical Sync*) y *paridad de campo* F_0 (*Field*).

```
#### INTERFASE CON DIGILENT VDEC-1
# inout
NET opb_iic_0_Sda_pin  LOC = A4 | IOSTANDARD = LVTTL  | DRIVE = 8 | PULLUP = TRUE;
NET opb_iic_0_Scl_pin  LOC = D5 | IOSTANDARD = LVTTL  | DRIVE = 8 | PULLUP = TRUE;
# in
NET perif_videototal_0_YCrCb_in_pin[2] LOC=D11 | IOSTANDARD = LVTTL;
NET perif_videototal_0_YCrCb_in_pin[3] LOC=E9  | IOSTANDARD = LVTTL;
NET perif_videototal_0_YCrCb_in_pin[4] LOC=F9  | IOSTANDARD = LVTTL;
NET perif_videototal_0_YCrCb_in_pin[5] LOC=E8  | IOSTANDARD = LVTTL;
NET perif_videototal_0_YCrCb_in_pin[6] LOC=E7  | IOSTANDARD = LVTTL;
NET perif_videototal_0_YCrCb_in_pin[7] LOC=B6  | IOSTANDARD = LVTTL;
NET perif_videototal_0_YCrCb_in_pin[8] LOC=A6  | IOSTANDARD = LVTTL;
NET perif_videototal_0_YCrCb_in_pin[9] LOC=C5  | IOSTANDARD = LVTTL;
#
NET perif_videototal_0_LLC_CLOCK_pin   LOC=E10 | IOSTANDARD = LVTTL;
## out
NET perif_videototal_0_RESET_VDEC1_Z_pin  LOC=B4  | IOSTANDARD = LVTTL | DRIVE = 8;
NET perif_videototal_0_VDEC1_OE_Z_pin     LOC=F7  | IOSTANDARD = LVTTL | DRIVE = 8;
NET perif_videototal_0_VDEC1_PWRDN_Z_pin  LOC=A16 | IOSTANDARD = LVTTL | DRIVE = 8;
```

Figura 4.16. Configuración de las conexiones externas de la FPGA. Notar la propiedad "PULLUP" en los puertos bidireccionales de la interfase I^2C, y "DRIVE" para regular corrientes de salida

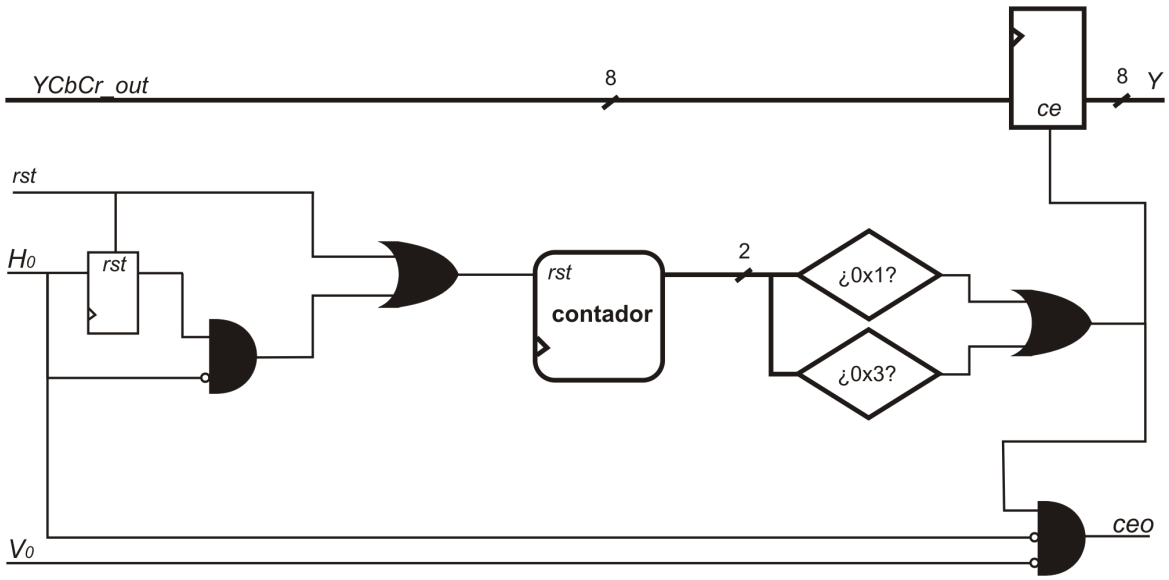


Figura 4.17. ConversorY. Toma la señal YCbCr y extrae de ella los componentes de intensidad Y

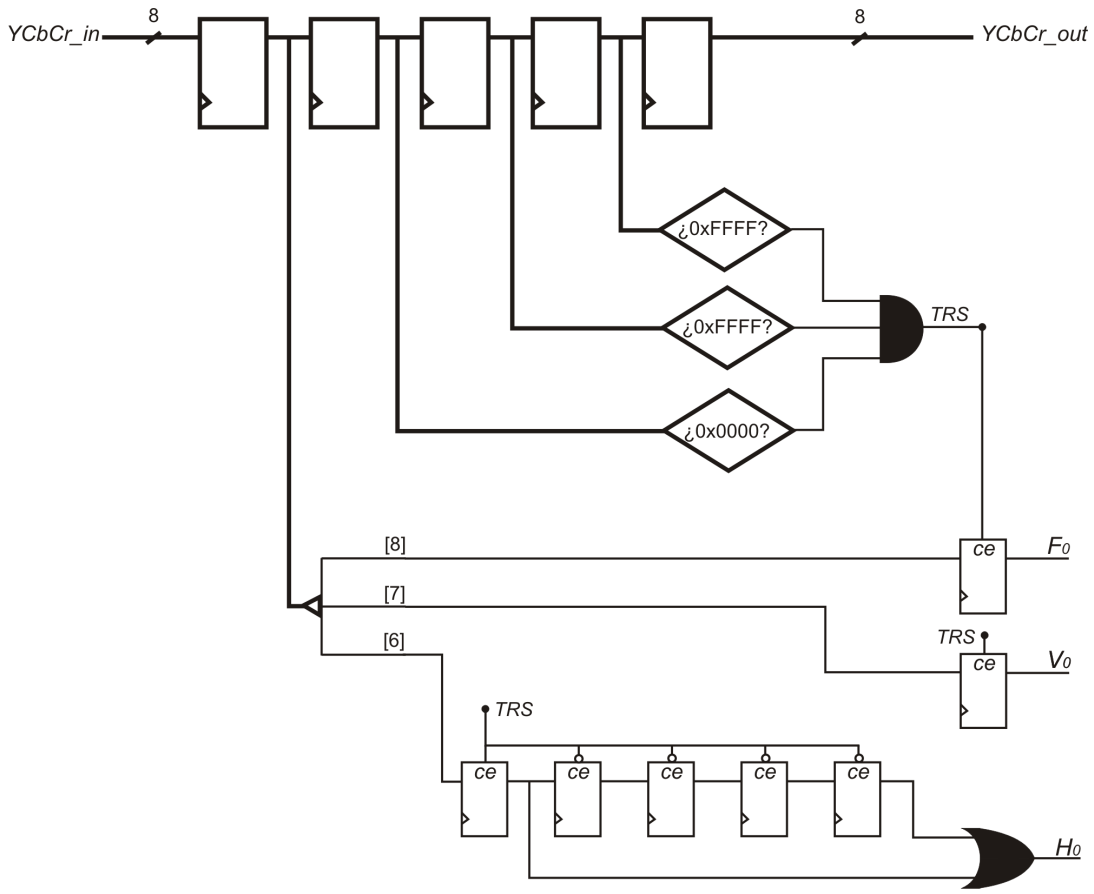


Figura 4.18. Sistema Sincronizador. Genera las señales de sincronismo H_0, V_0, y F_0

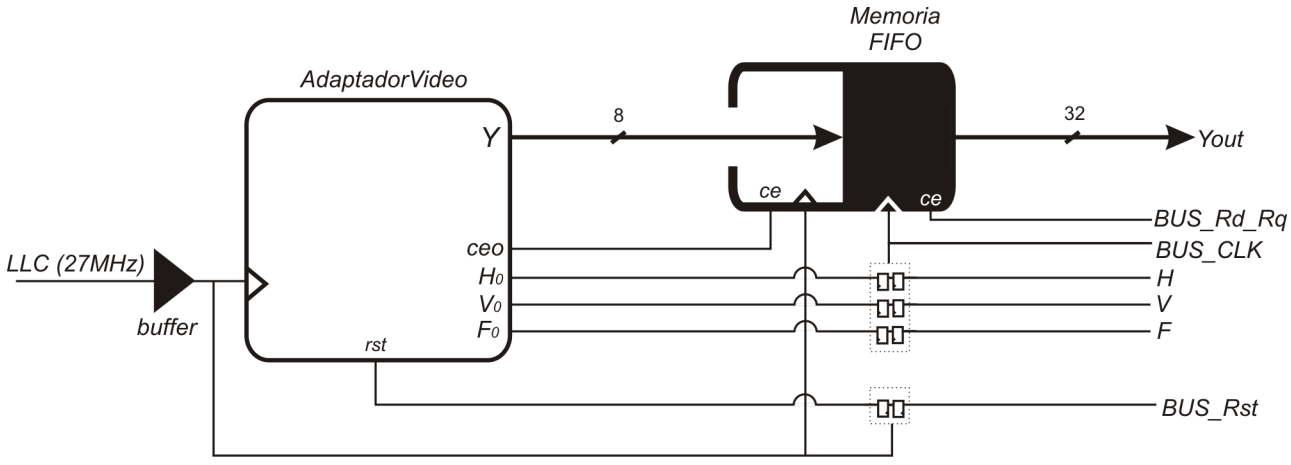


Figura 4.19 Módulo SincDominio. Lo utilizamos para transferir datos (memoria FIFO) y señales de control (dobles flip-flops) desde el dominio de reloj del AdaptadorVideo al dominio de reloj principal de la FPGA.

Existe un cambio de dominio de reloj que deben efectúan los datos de video y las señales de sincronización. Existe un dominio de reloj principal, en nuestra implementación de frecuencia 50 MHz, al cual debe ingresar para ser procesada la información de video capturada junto al reloj externo de 27MHz antes mencionado. La Figura 4.19 esquematiza la *memoria FIFO* usada para la transferencia de los datos de video, y los bloques *dobles flip-flops* usados para transferir con bajas probabilidades de metaestabilidad las señales H_0, V_0 y F_0. En sentido opuesto también transferimos una señal de *reset* por doble flip-flops al dominio más lento para inicializar el módulo AdaptadorVideo. La memoria FIFO y el conjunto de dobles flip-flops constituyen el sistema de sincronización *SincDominio*.

En la siguiente sección explicaremos la adaptación de éste módulo de captura de video a una arquitectura tipo *bus* que conforma el sistema embebido en el dispositivo FPGA.

4.3.2 Arquitectura de *bus* embebida

En el Capítulo 3, al introducir los sistemas embebidos, comentamos el potencial que guardan las arquitecturas organizadas alrededor de canales principales de datos y control como son los *buses*. Conectar varios periféricos, con funciones muy específicas (capturar información de video, procesarla con un algoritmo PIV y extraer vectores de movimiento), en un diseño globalmente controlado por un microprocesador de propósito general, reduce las múltiples tareas de comunicación y control de las partes al manejo de un mapa de memoria.

En la Figura 4.20 representamos el bus *OPB* (*On-chip Peripheral Bus)* implementado, el microprocesador (tipo RISC de 32 bits *MicroBlaze*) y los distintos periféricos conectados a él. Destacamos la conexión al bus del CapturadorVideo, un controlador de DMA, un módulo UART con conexión externa RS-232 y el módulo procesador del algoritmo de PIV que internamente puede poseer uno o varios módulo de procesamiento de correlación cruzada (PCC). Los módulos diseñados para este trabajo (CapturadorVideo y PCC) poseen una interfase con el protocolo del bus OPB basada en el adaptador de periféricos IPIF que provee Xilinx en su herramienta de diseño de sistemas embebidos Xilinx Platform Studio – EDK.

El programa descrito en lenguaje C e implementado en el microprocesador, al inicializarse, habilita los componentes de HW encargados de las funciones de DMA, los servicios de interrupciones y la configuración por I^2C del digitalizador externo de video. El periférico de I^2C lo consideramos un sub-módulo de CapturadorVideo. Conjuntamente activamos funciones importantes de SW como las rutinas de manejo de interrupciones (ISR).

A medida que llega video por el puerto YCbCr, y cargamos la memoria FIFO (de CapturadorVideo), se generan interrupciones al microprocesador asociadas a las señales de V, F y H, que alertan sobre eventos de sincronización (inicio de imagen, cambio de campo, fin de una línea). Al ejecutar el servicio ISR este identifica el tipo de interrupción y se actualizan los parámetros de acceso DMA; el paso siguiente es una transferencia de datos desde la FIFO a la memoria externa DDR SDRAM tomando transitoriamente el controlador DMA acceso maestro al bus. Cada una de estas transferencias es de 720 bytes (el número de píxeles de una línea activa en NTSC), y realizamos casi 16000 por segundo (525 lineas por cuadro en NTSC).

El microprocesador actualiza la posición de destino (una dirección en la memoria externa) de la transferencia DMA tras cada interrupción, componiendo de ese modo el entrelazado del video y el orden de los cuadros en la memoria DDR SDRAM (con capacidad de 64 MB) [12]. El sistema de comunicación RS-232 (módulo UART) trabaja a 115.2 kbps y lo empleamos para extraer los resultados del proceso. En las etapas de diseño es de utilizadad la conexión RS-232 como mecanismo de evaluación de los resultados de generados en el procesador PCC y en la formación de los cuadros de video en la memoria externa.

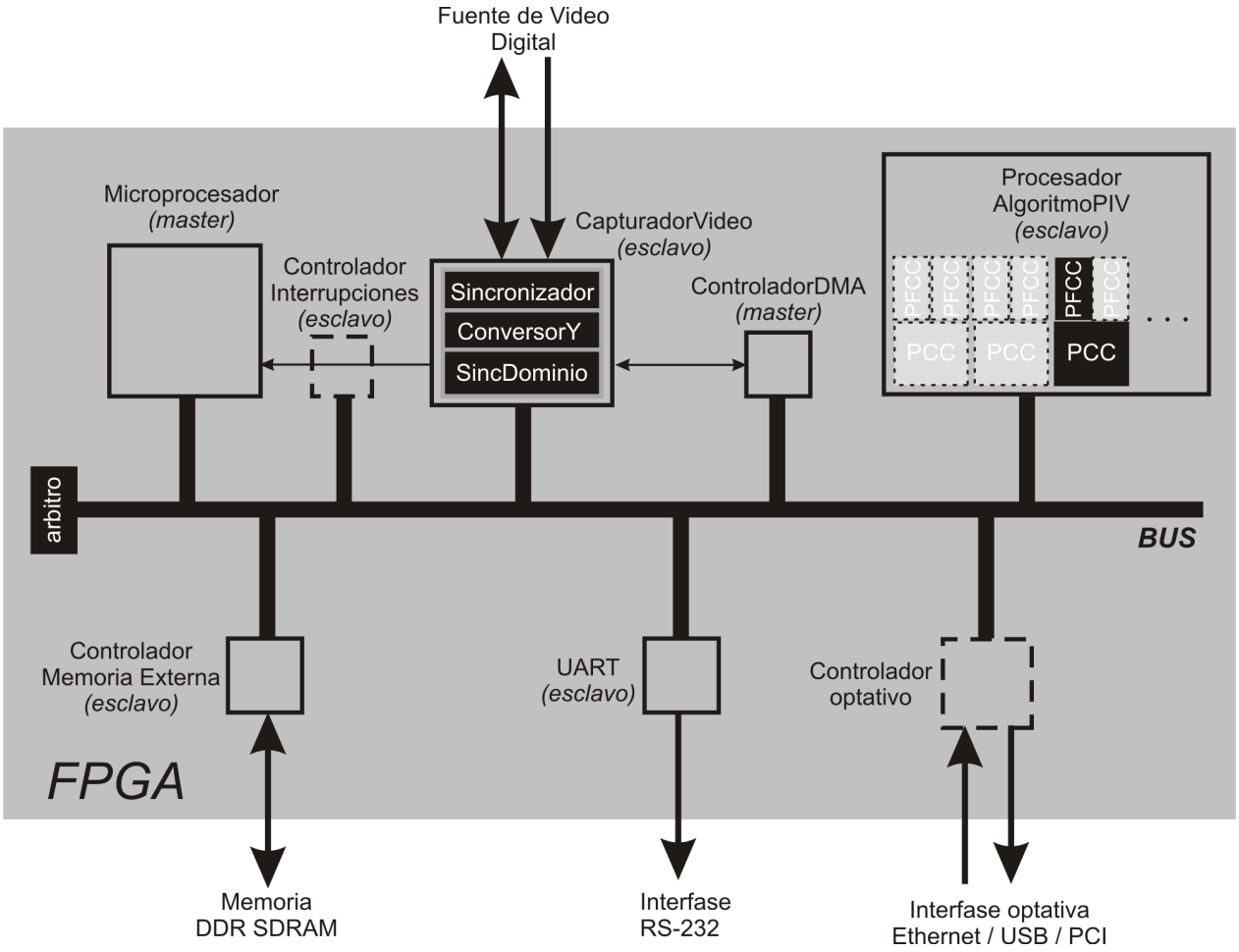


Figura 4.20. Sistema embebido con las partes principales de una plataforma de captura, almacenamiento y procesamiento en paralelo para un sistema PIV en tiempo real.

Los recuadros a trazos de la Figura 4.20 (*Ethernet/USB/PCI* y *ControladorInterrupciones*) indican la posibilidad de adaptar con facilidad las funciones de esta plataforma embebida a otros modos de conexión externa distintos de la captura directa mediante CapturadorVideo (o un módulo equivalente). Algunas aplicaciones pueden buscar velocidad de procesamiento *off-line* en lugar de tiempo real. Mencionamos a Ethernet, USB y PCI como interfases de comunicación de alta velocidad, con HW dedicado presente en muchas plataformas de desarrollo FPGA. Desarrollamos éste trabajo con el kit de desarrollo Xilinx Spartan-3E Starter Kit Board, con capacidad para Ethernet solamente,

cuya factibilidad estudiamos aunque se priorizó la captura directa desde la cámara. Detallamos características de la FPGA utilizada en la Tabla 4.2 [11] y en la Figura 4.21 los componentes del sistema.

Dispositivo	Compuertas Lógicas	Celdas Lógicas	RAM distribuida	RAM en bloque	Multiplicadores dedicados	Administradores digitales de reloj (DCM)
XC3S500E	500 K	10476	73 Kb	360 Kb	20	4

Tabla 4.2 Características de la FPGA (Spartan-3E) empleada en el diseño

En la próxima sección mostramos los primeros resultados obtenidos con la implementación de la Figura 4.20.

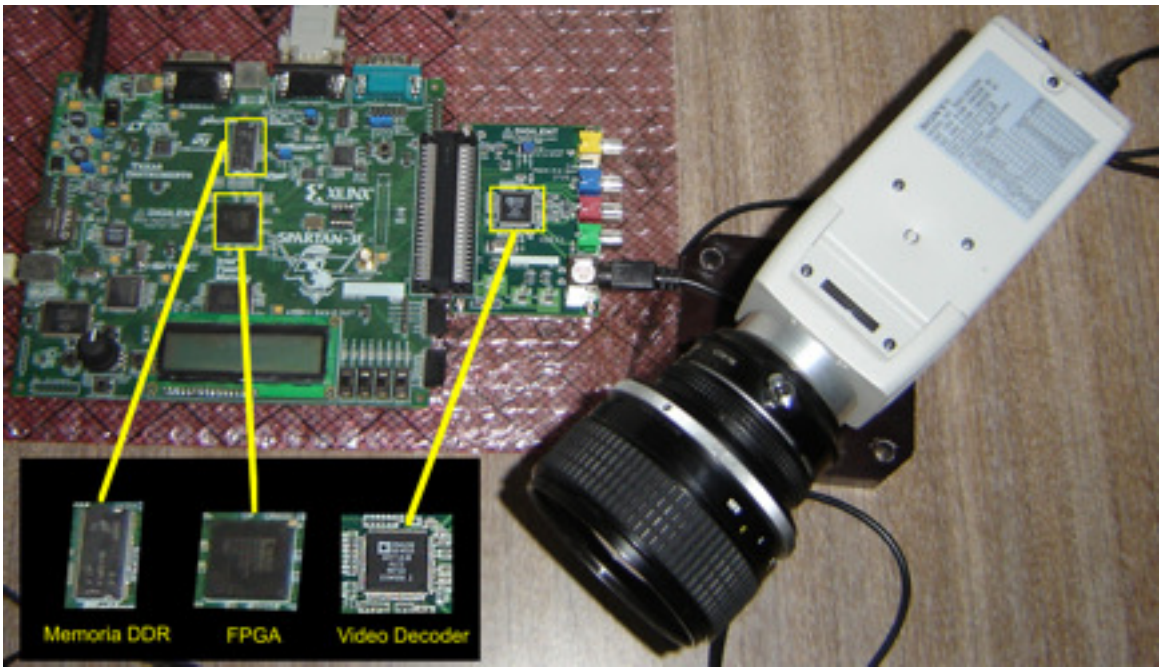


Figura 4.21 Kit de desarrollo FPGA junto a la placa digitalizadora y la cámara de video conectada al módulo mediante S-Video. Destacamos la conexión RS-232 en el lado superior

4.4 Resultados: análisis y discusión

Las siguientes líneas de tiempo permiten diagramar cualitativamente parámetros de cada proceso de transferencia y computo dentro la arquitectura de bus del diseño. Representamos con el área de rectángulos los volúmenes de datos, que pueden ser parte de una transferencia o de un procesamiento. El ancho de los rectángulos es una medida proporcional del tiempo necesario en el proceso.

En la primera línea temporal describimos la puesta en marcha del sistema. Señalamos la inicialización de componentes de HW y servicios de SW, y seguidamente el comienzo de las transacciones periódicas de video (cada una de 720 bytes mediante acceso DMA) entre la memoria FIFO de CapturadorVideo y la memoria externa DDR SDRAM.

Una vez conformados al menos dos cuadros de video en la memoria externa, se puede habitar el proceso de carga de las ventanas de interrogación en los procesadores PCC.

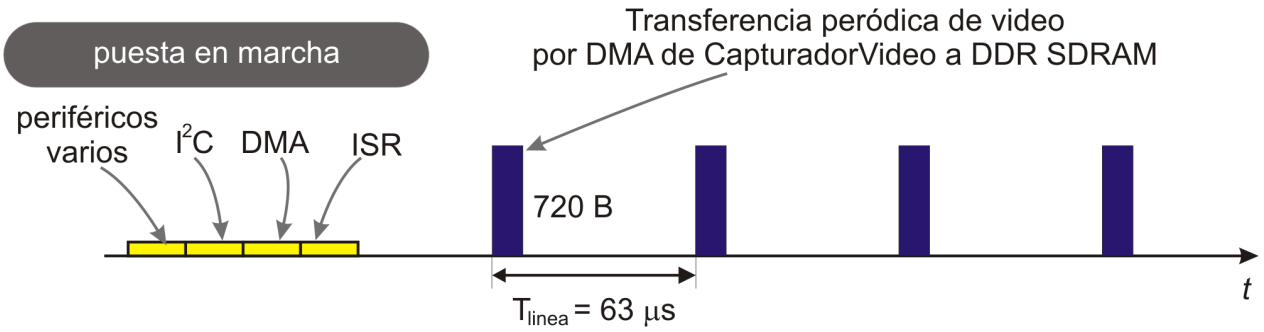


Figura 4.22 Puesta en marcha de sistema embebido y comienzo de transacciones de video

La carga de datos desde la memoria externa a las memorias de los procesadores PCC (AMem y BMem) podemos realizarla de varias maneras, con complejidades y beneficios diferentes. En la siguiente Figura representamos un caso donde se transfiere una ventana de interrogación a un PCC. Notar la menor altura del rectángulo inferior debida a las menores eficiencias que en general tienen estas transferencias. En el caso de esa Figura también suponemos una transferencia DMA desde la memoria externa a PCC. La Figura 4.24 posee una línea de tiempo adicional para describir lo que ocurre en un procesador PCC. Según las simulaciones de nuestro diseño, trabajando a una frecuencia de 50 MHz y con un solo 4 multiplicadores (un módulo PFCC), el tiempo de procesamiento es $T_{PCC} \approx 0.5$ ms.

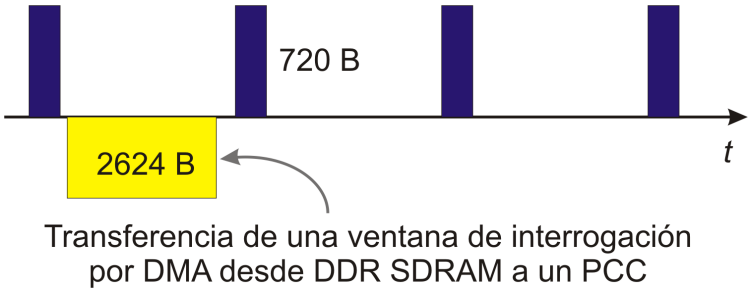


Figura 4.23 Transferencia de una ventana de interrogación (1024 bytes para subAreaA y 1600 bytes para AreaB) desde la memoria externa a los coprocesadores.

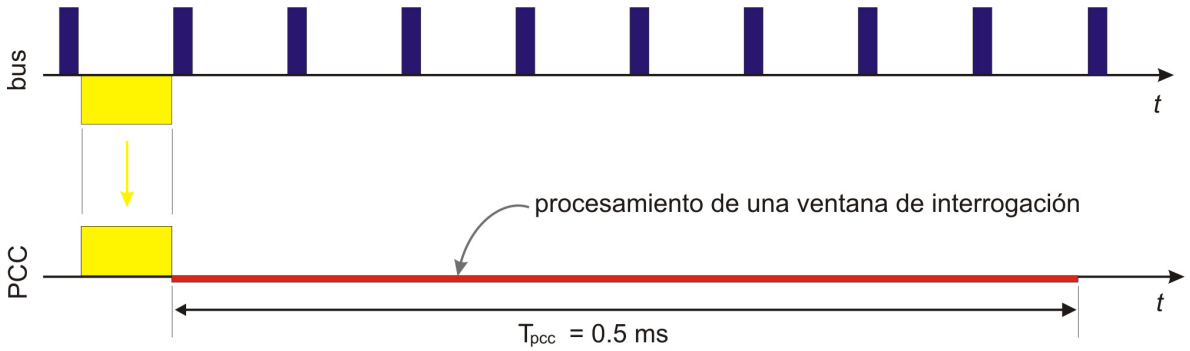


Figura 4.24 Relación aproximada entre los tiempos de transferencia de lineas de video, carga de PCC y procesamiento de PCC con 4 multiplicadores a 50 MHz

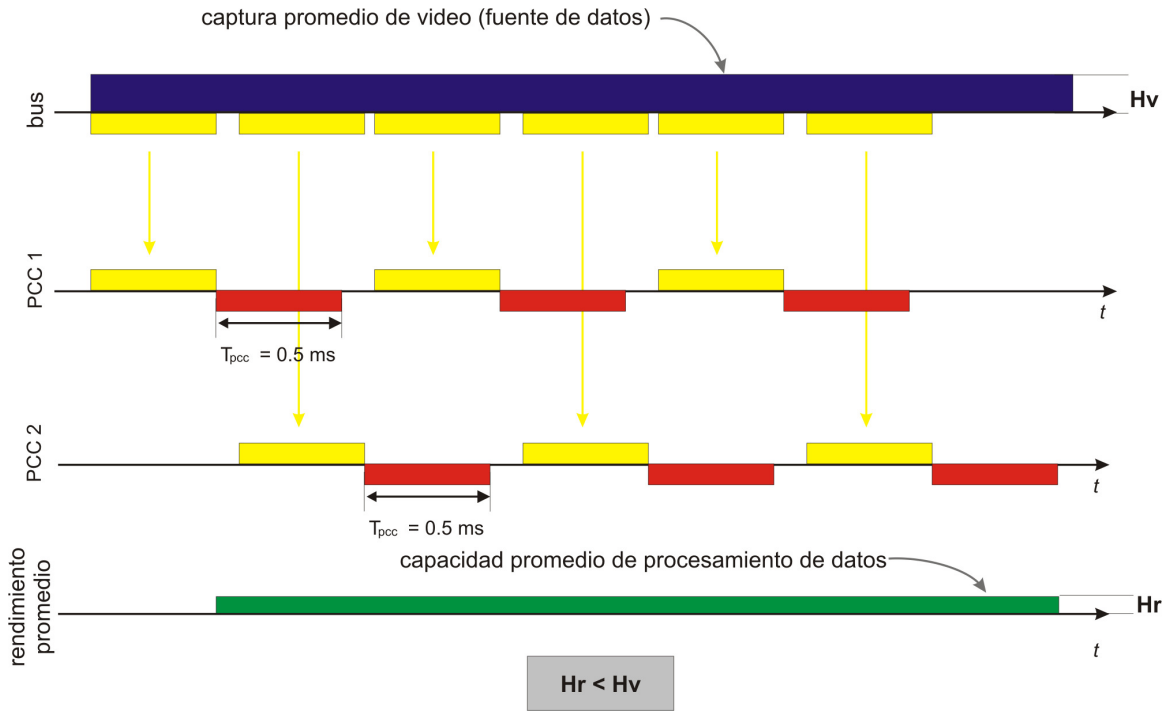


Figura 4.25 Sistema continuo de cálculo. En este caso la capacidad media de procesamiento es menor a la cantidad de información disponible

Conseguir que el sistema funcione en estado estacionario significa tener capacidad de capturar el video, transferir ventanas de interrogación, procesarlas, tomar los resultados y enviarlos al exterior. La ocupación del bus generada por las transferencias de video no representa el mayor desafío para el procesamiento continuo, en gran parte a las facilidades otorgadas por DMA y los servicios de interrupciones. Las búsquedas de estrategias deben enfocarse en el número a implementar de procesadores PCC y su configuración interna. Cuantos más multiplicadores consigamos hacer trabajar en paralelo y las transferencias de ventanas de interrogación sean fluidas, el sistema crecerá en potencia de cálculo. Poder llevar la arquitectura a mayores frecuencias de trabajo es otro camino para incrementar la velocidad de procesamiento. La Figura 4.2.5 muestra el caso de un sistema con dos procesadores PCC de 4 multiplicadores (un PFCC) cada uno y transferencias de ventanas sin acceso DMA. La línea inferior posee una muestra integral del rendimiento (*thoughput*) de este supuesto sistema dado por la altura de su rectángulo.
El caso de la Figura 4.26 consiste en módulos PCC de 8 multiplicadores y transferencias DMA entre la memoria externa y los PCC.

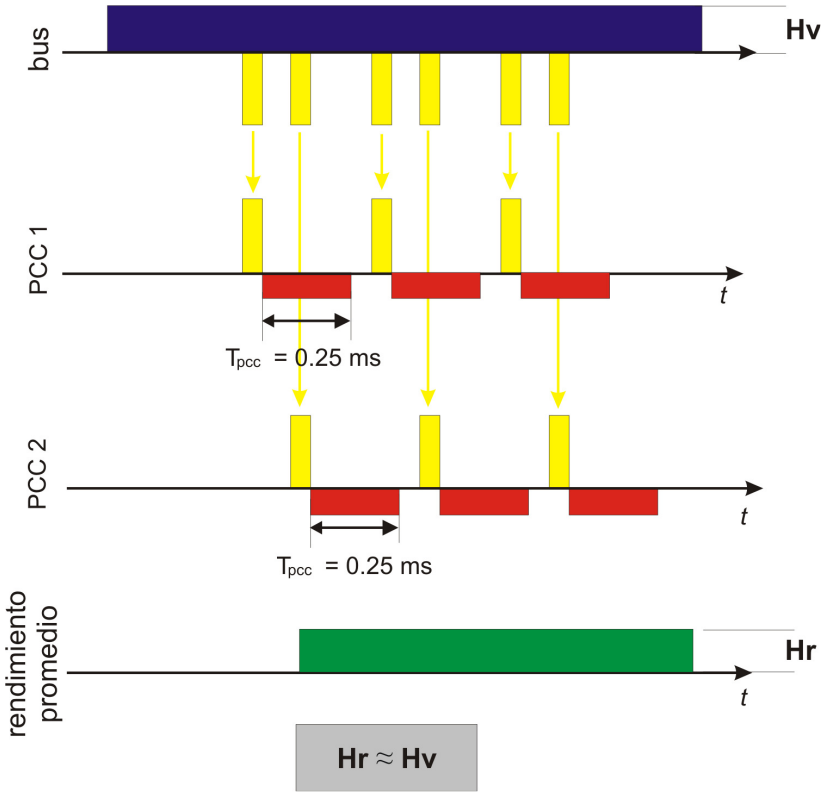


Figura 4.26 Sistema con 8 multiplicadores por procesador PCC y transferencias DMA entre la memoria externa y PCC

Habiendo presentado los posibles modos de trabajo del sistema, describimos el estado actual de la integración de los componente de nuestro diseño.

El sistema diseñado se encuentra implementado con un procesador PCC de configuración básica (1 módulo PFCC y 1 módulo CdP). Los recursos más importantes del mismo son dos memorias en bloque, cuatro multiplicadores y la lógica del Control de Proceso (CdP). En el chip Spartan-3E SXC3S500E se implementó la solución. El controlar CdP sintetiza para un máximo de 140 MHz y utiliza 221 *slices* (medida celdas lógicas) con 256 flip-flips y 380 LUTs de 4 entradas. El procesador fundamental PFCC implementado con multiplicadores distribuidos (en lugar de dedicados) sintetiza a 230 MHz y emplea 174 *slices* con 52 flip-flops y 299 LUTs. En CapturadorVideo la mayor solicitud se encuentra en la memoria de doble puerto (BRAM) necesaria para las funciones de FIFO.

En la siguiente Figura mostramos una imagen cargada desde la cámara Sony Exwave en la memoria externa DDR SDRAM, con capacidad para almacenar cerca de 5 s de video continuo (cuadros cada 30 ms).

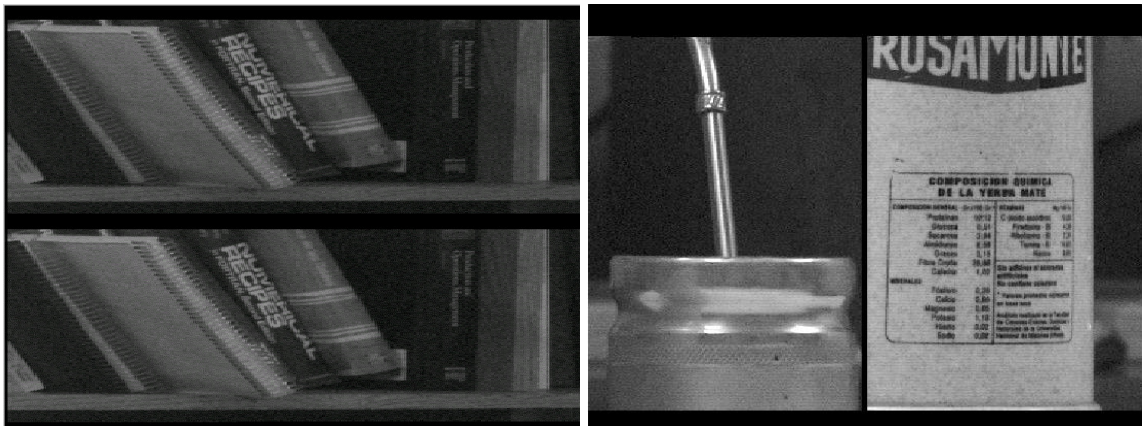


Figura 4.27 A la izquierda mostramos un cuadro de video sin componer su entrelazado. A la derecha se verifica la correcta formación de imágenes entrelazadas en la memoria DDR SDRAM. En ambos casos la imagen es monocromática en formato NTSC ocupando un espacio continuo de memoria.

La Figura presenta un muestreo reducido (por haber transferido una de cada 6 líneas mediante RS-232 -que es un puerto lento para transmisión de video-) de varios campos de video almacenados secuencialmente en la memoria externa, donde se aprecia un patrón de partículas pintadas sobre un fondo oscuro que se desplazan por un movimiento de vaivén de la cámara. Ese patrón de partículas se fija estático respecto a la visión de la cámara para detectar movimientos de una porción de área rectangular alargada e igualmente pintada.

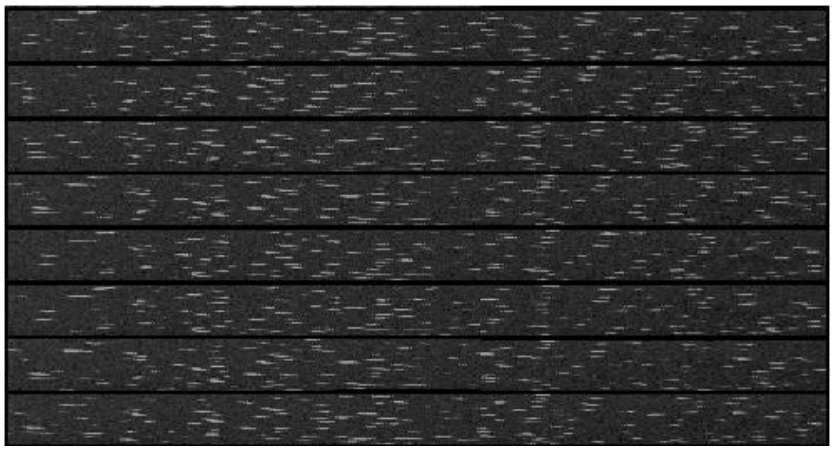


Figura 4.28 Muestreo reducido (1 de cada 6 líneas) de 8 cuadros consecutivos con entrelazado compuesto de imágenes de partículas pintadas. Los desplazamientos son producto de pequeños movimientos horizontales de la cámara.

El siguiente campo de velocidades se obtuvo tras el movimiento relativo del conjunto ordenado en un rectángulo de partículaso. El procesamiento se realizó de la siguiente manera: tras la captura continua de durante un segundo de video, se deshabilitaron las interrupciones de sincronismo con la llegada de video y transferencias internas por DMA y se puso en marcha un mecanismo no optimizado (por no emplear aún DMA) de transferencia de datos entre la memoria externa y las memorias AMem y BMem dentro del periférico que contiene al módulo PCC. Se correlacionaron ventanas de interrogación de los

primeros dos cuadros del video capturados y se evaluaron una a una 150 ventanas de interrogación (cubriendo un área de 600 píxeles x 400 píxeles). Queda en evidencia el desplazamiento vertical que realizaron las partículas ordenadas en un rectángulo, donde la magnitud de los vectores es del orden de 1 cm/s. La iluminación del sistema es cada 30 ms como la frecuencia de captura de la cámara. En un experimento PIV se tiene control del tiempo entre pulsos laser de iluminación que resultan ser muchos menores a estos 30 ms como mostramos en el Capítulo 1. De ese modo se podrían medir velocidades mayores en un problema de plano laser con este mismo dispositivo.

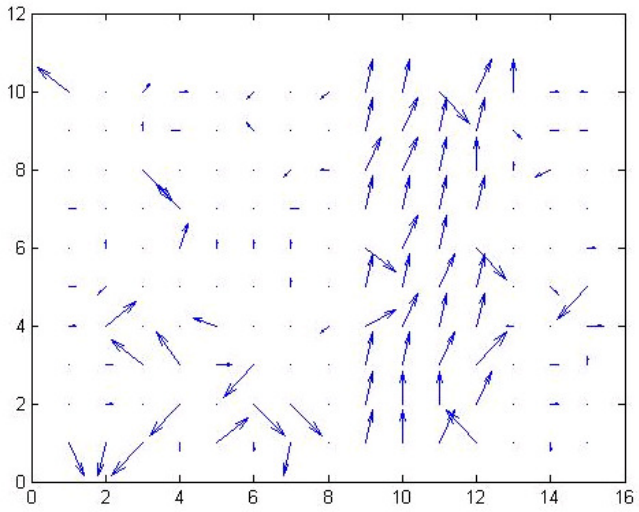


Figura 4.29 Resultados tras capturar con el sistema el movimiento relativo entre puntos blancos pintados en un rectángulo móvil y puntos similares sobre un fondo oscuro estático

Una vez comprobada la correcta integración y funcionamiento del conjunto de componentes del sistema embebido en la FPGA, lo configuramos para un procesamiento continuo. Tras esta primera instancia del proyecto conseguimos que el instrumento puede adoptar un modo con refresco del campo vectorial menor a un segundo. Otra configuración permite capturar en forma continua varios segundos de video y tras detenerse el sistema se captura se procesa tantos pares de cuadros como se desee y de ese modo es posible capturar un transitorio del problema bajo estudio. En la Figura 4.30 mostramos la aplicación corriendo en una PC convencional que recibe por RS-232 la información desde la FPGA y en ella se refresca el campo de velocidades directamente en el monitor. La tasa de refresco es de 2 Hz, lo que significa 25 E6 multiplicaciones de correlación por segundo. Creemos que es posible incrementar en un orden esta cifra empleando la misma plataforma Spartan-3E.

Las herramientas usadas en el diseño fueron *Xilinx ISE 9.1i*, *Xilinx Platform Studio (EDK) 9.1i*, y en las simulaciones Mentor Graphics ModelSim 6.1. La codificación de los módulos se realizó en su mayoría con lenguaje VHDL al igual que los estímulos usados en las simulaciones.

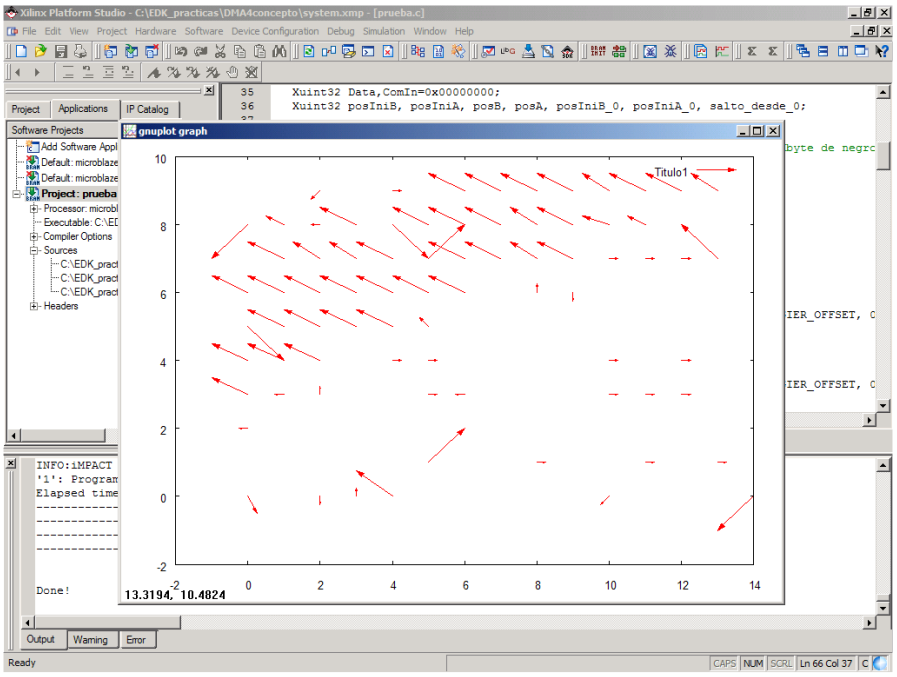


Figura 4.30 Programa de captura continua por RS-232 de los campos de velocidades calculado en la FPGA. El refresco del campo vectorial es cada 0.5 segundo.

4.5 Planes futuros

El sistema descrito puede, en el corto a mediano plazo, adoptar nuevas configuraciones con mayor rendimiento (*throughput*) y mejor robustez en su resolución, incluso en la misma plataforma Spartan-3E donde se implementó.

Sugerimos como primer paso a seguir invertir esfuerzos en el aumento del número de multiplicadores trabajando en paralelo, en primer lugar configurando las memorias en bloque como doble puerto y duplicando el procesador PFCC y lógica asociada dentro de cada PCC. El paso siguiente sería la implementación de varios PCC dentro de la arquitectura. Si el uso del bus llegase a comprometer el manejo fluido de datos, proponemos implementar acceso DMA en la transferencia de las ventanas de interrogación.

En este trabajo explicamos algunas estrategias para implementar en el HW un algoritmo más robusto que el de correlación cruzada directa, donde la opción que ponderamos es la correlación cruzada normalizada.

Este sistema puede programarse en una FPGA más potente, con mayores recursos dedicados y frecuencia de trabajo. Es de interés crear una versión para la captura de información por otros medios, como Ethernet, USB o PCI, para ampliar el campo de acción del dispositivo.

Como todo sistema de medición requerirá de calibraciones, que en este caso suele hacerse con flujos de solución conocida. El sistema de manejo de ventanas de interrogación permite reconfigurar el HW para adaptar la estrategia presentada en éste trabajo a modalidades de PIV que usen otro criterio en la forma y tamaño de las áreas.

Tenemos en cuenta la posibilidad de emplear dos circuitos FPGA para implementar en uno la resolución de las componentes de velocidad en un plano mediante PIV, y en el

segundo la integración de la ecuación de continuidad para calcular la tercer componente de velocidad (PIV traslativo).

4.6 Resumen

En este capítulo analizamos las posibles arquitecturas e interconexión de los bloques lógicos para el desarrollo del módulo de procesamiento PIV. Los principales aspectos que tuvimos en cuenta para el diseño son: minimización de los recursos lógicos dedicados a cada módulo, diseño flexible, código reutilizable. Además realizamos un análisis sobre la implementación de los multiplicadores y el manejo eficiente de la memoria interna.

Un bloque en cascada denominado procesador PFCC es responsable del cálculo de suma de productos solicitado para computar la correlación cruzada en pocas etapas. La administración de los corrimientos de muestreo queda bajo responsabilidad del módulo denominado Controlador de Proceso.

El desarrollo detallado expone una solución para implementar en lógica programable el algoritmo de correlación cruzada directa (con método de correspondencia de patrón) aplicable al problema de velocimetría PIV por plano laser. Esta aproximación nos permite afirmar que con un máximo de 10 procesadores PFCC trabajando en paralelo (40 multiplicadores en total) a una frecuencia de reloj de 50 MHz, obtendríamos capacidad de procesamiento en tiempo real con rendimiento equivalente al reportado en la literatura [1]. Resulta un sistema flexible que permite modificar el número de procesadores PFCC según las capacidades del chip FPGA que se disponga. A su vez, es muy simple cambiar parámetros de cada procesador PCC para aprovechar particularidades de la arquitectura FPGA o del problema PIV: memorias en bloques, memorias distribuidas, multiplicadores dedicados, forma y tamaño de las ventanas de interrogación, etc.
Otro aspecto de ésta estrategia es la administración de los corrimientos de muestreo (consecuente manejo de datos del AreaB) realizada en un HW específico (CdP).

Referencias

[1] H.Yu, M. Leeser, G. Tadmor, S. Siegel, *Real-time Particle Image Velocimetry for Feedback Loops Using FPGA Implementation*, American Institute of Aeronautics and Astronautics, 2003.

[2] E. B. Arik, J. Carr, *Digital Particle Image Velocimetry System for Real-Time Wind Tunnel Measurements*, 1995

[3] Xilinx, Inc., *Synthesis and Simulation Design Guide*, v9.1i, 2007

[4] H. Zheng, L.Liu, L. Williams, R. Shandas, *Effect of Contrast Microbubble Concentration on Quality of Echo Particle Image Velocimetry (Echo PIV) Data: Initial in Vitro Studies*, 2003

[5] Mark Balch, *Complete Digital Design – A Comprehensive Guide to Digital Electronics and Computer System Architecture*, McGraw Hill, 2003

[6] D. Perry, *VHDL*, fouth edition, McGraw-Hill, 2002

[7] Ken Chapman, Xilinx Inc., *Performance + Time = Memory (Cost Saving with 3D-Design)*, White Paper, 2008

[8] Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, Springer, 2001

[9] S. Kilts, *Advanced FPGA Design*, Wiley, 2007
[10] A. Prasad, *Particle Image Velocimetry*, Review Article, Current Science, Vol. 79, No.1, July 2000
[11] Xilinx Inc., *Spartan-3E FPGA Family: Complete Data Sheet*, 2007
[12] Xilinx Inc., *Spartan-3E Starter Kit Board User Guide*, UG230(v1.0), 2006
[13] V. Pedroni, *Circuit Design with VHDL*, MIT Press, 2004
[13] Xilinx Inc., *Spartan-3E Libraries Guide for HDL Designs*, ISE 8.2i, 2006
[14] Xilinx Inc., *Development System Reference Guide*, 9.1i, 2007
[15] Xilinx Inc., *Microblaze Processor Reference Guide*, UG081 v5.4, 2006
[16] Xilinx Inc., *OPB IPIF Architecture*, DS414 v1.3, 2003
[17] Xilinx Inc., *Constraints Guide*, 9.1i, 2007
[18] Xilinx Inc., M. Maaref, *Creating an OPB IPIF-based IP and using it in EDK*,App.Note 967, 2007
[19] Xilinx Inc., *XST User Guide*, v9.1, 2005
[20] Xilinx Inc., K. Parnell, N. Mehta, *Programmable Logic Design Quick Start Handbook*, 2003

Capítulo 5

Sumario y Conclusiones

El procesamiento en tiempo real para realizar velocimetría PIV por plano laser es una facilidad experimental que brinda flexibilidad a un experimento de fluidodinámica, y abre la posibilidad de implementarse en lazos de sistemas de control.

El alto costo computacional, propio del procesamiento estadístico de imágenes, es causa de una alta latencia entre que se realiza un experimento PIV y se obtiene su campo vectorial.

La tecnología FPGA puede adoptar arquitecturas de procesamiento muy paralelas y funcionalidades muy dedicadas. De mucha importancia son las opciones de conectividad de estos circuitos programables, característica que pone al alcance del diseñador digital un enorme conjunto de técnicas de transmisión de datos en diversos anchos de banda y estándares eléctricos.

El estudio de las técnicas digitales básicas empleadas para implementar algoritmos, funciones matemáticas, manejar flujos de datos, y realizar interfases de comunicación, nos permitió distinguir bloques funcionales necesarios para nuestro proyecto. A partir de ellos el trabajo se dividió en etapas que se ordenaron desde las primeras experiencias en VHDL y simulaciones de comportamiento del hardware descrito, hasta la puesta en funcionamiento del sistema embebido en la FPGA; aprendiendo el manejo de diversas herramientas de desarrollo y reconociendo como debe ser un código sintetizable.

En este trabajo desarrollamos un módulo de hardware que recibe la información de una ventana de interrogación y devuelve como resultado un vector de desplazamiento. En él ejecutamos el algoritmo PIV de correlación cruzada directa usando una estrategia de correspondencia de patrón. Su condición final es servir como coprocesador en sistemas embebidos, y admite configuraciones que optimizan su empleo en diversas arquitecturas FPGA. Sobre todo brinda versatilidad ante distintos niveles de recursos en una FPGA (es simple implementar tantos módulos como sea posible).

Presentamos el diseño de arquitectura de bus embebida con periféricos dedicado a la captura de imágenes obtenidas desde una cámara de video, el procesamiento de ventanas de interrogación y ejecución del algoritmo PIV. El sistema se comunica por RS-232 con una computadora genérica que recibe los resultados. Se emplean las técnicas de interrupciones y acceso directo a memoria como modalidad principal de trabajo en la arquitectura implementada. Exploramos la comunicación de datos entre el

dispositivo FPGA y una computadora mediante el protocolo de alta velocidad Ethernet (IP/UDP). Queda abierta la posibilidad de utilizar además los puertos USB y PCI en un trabajo futuro.

Concluimos que la implementación de algoritmos a nivel de hardware concurrente permite velocidades de procesamiento necesarias en aplicaciones de tiempo real. Es una posibilidad efectiva realizar velocimetría PIV basada en plano laser con diseños implementados en lógica programable. Los sistemas en FPGA son flexibles al modo de incorporar la información de video (*on-line* / *off-line*) y para diferentes niveles de paralelización del cálculo.

En cuanto al diseño, verificamos la confiabilidad en las técnicas de transmisión de datos y señales entre dominios diferentes de reloj: memorias FIFOS y dobles flip-flops. Destacamos el papel de las máquinas de estados finitos en la ejecución de los algoritmos, y como pueden optimizarse con particiones y la incorporación previa de cálculos en tablas LUTs.

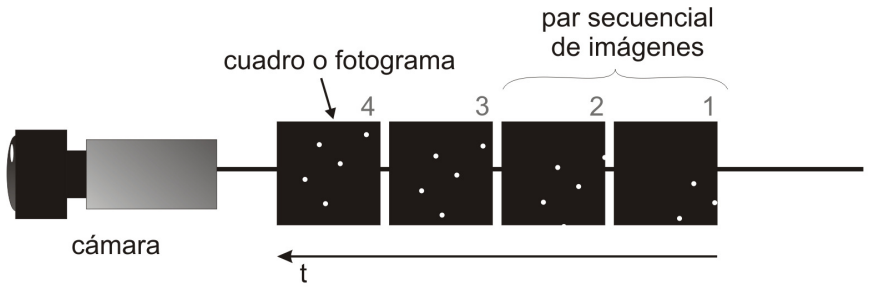
Fue necesario realizar simulaciones en varias etapas del flujo de diseño para verificar los comportamientos esperados en la lógica y el cumplimiento de los tiempos. El lenguaje descriptor de hardware VHDL, empleado predominantemente en el diseño, nos permitió describir tanto los módulos de síntesis como los estímulos utilizados en las simulaciones. Las herramientas graficas son de gran ayuda para realizar descripciones esquemáticas de un circuito con muchas conexionados. Basamos el trabajo en programas de desarrollo de la empresas Xilinx Inc. (ISE y EDK v9.1) y de simulación de Mentor Graphics Corp. (ModelSim v6.1). Empleamos otras herramientas frecuentes de ingeniería de Matlab 7-R14 (MathWorks Inc.) en la evaluación de los algoritmos y las estrategias de diseño de las máquinas de estado.

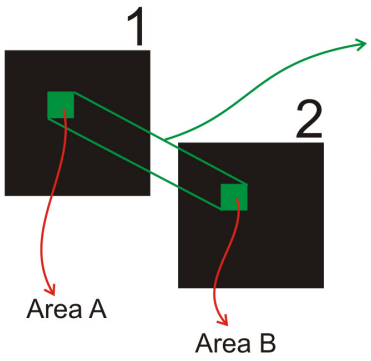
Encontramos en la arquitectura básica de computadora la flexibilidad suficiente, a través de accesos directo a memoria y el empleo de servicios de interrupciones, para integrar de forma eficiente todos los módulos del sistema.

Anexo

A. Definiciones para el algoritmo de Correlación Cruzada

En particular se trata de la versión correlación cruzada espacial usando ajuste de patrón





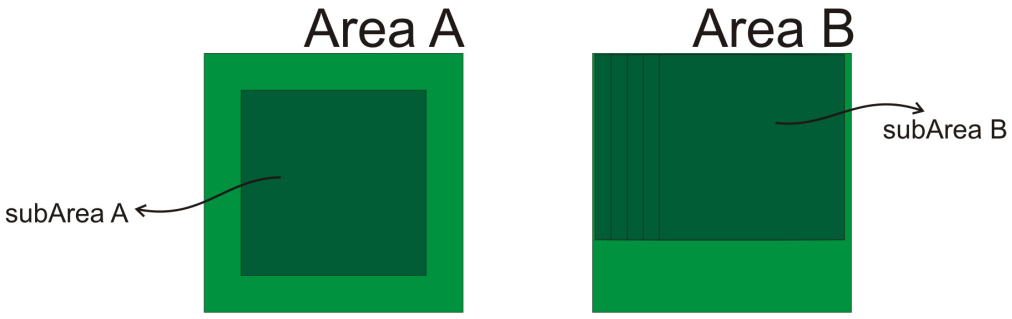
Ventana de interrogación:

referencia a un conjunto de píxeles de **ambos cuadros** en idénticas posiciones. O sea, ocupan la misma área relativa en ambos cuadros. Tamaño nominal: 40 x 40

La parte asociada al primer cuadro se llama **AreaA**
La parte asociada al segundo cuadro se llama **AreaB**

Convención:
1) siempre que se mencionan de forma genérica, las palabras ***patrón de intensidad***, ***imagen***, ***área*** o ***señal***, estas pueden tomarse con significados equivalentes.

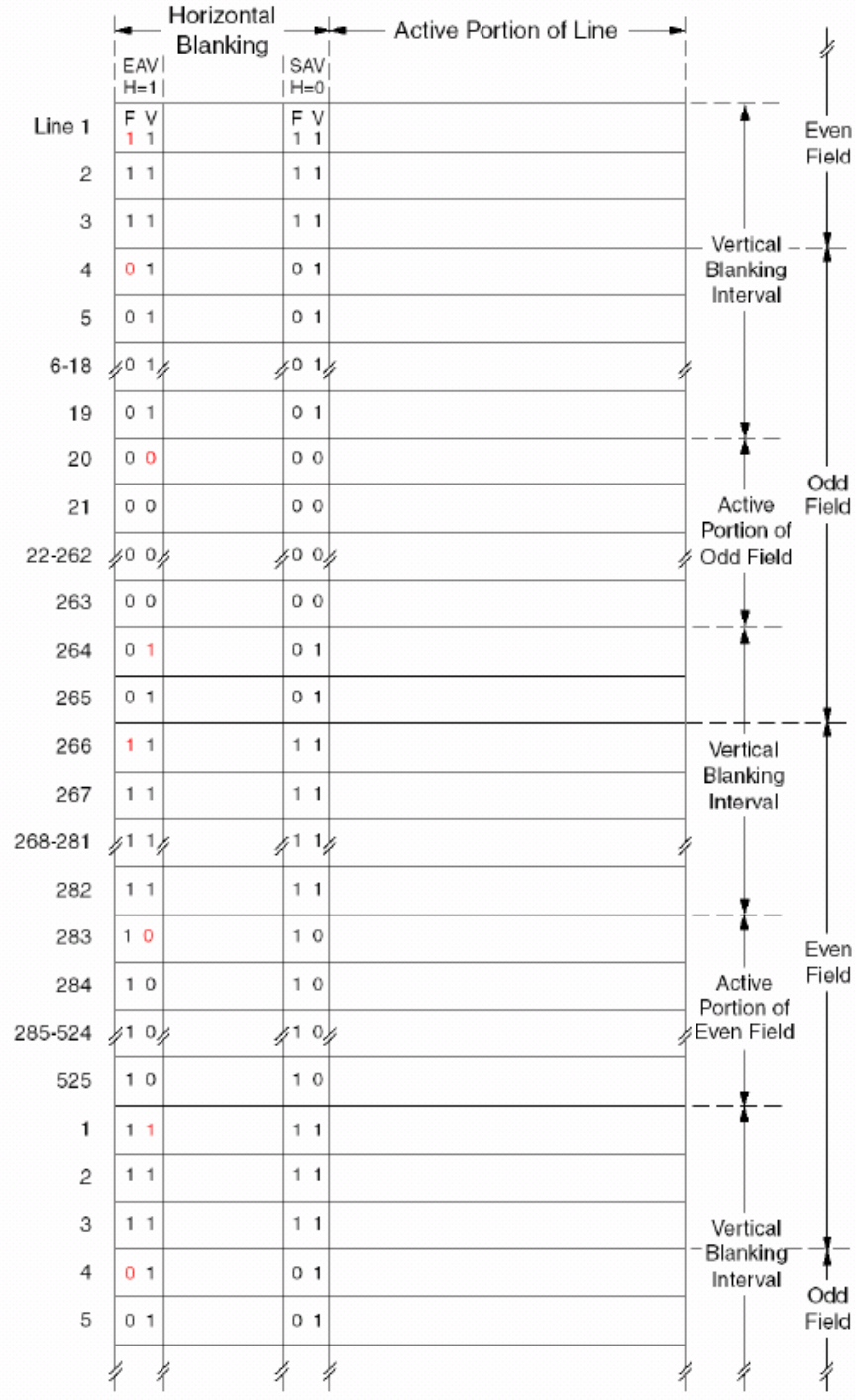
2) cada referencia usando la frase ***ventana de interrogación*** implica una mención exclusiva a la definición recién indicada.



subArea A: es una porción interna de Area A, también cuadrada y en general constituida por los píxeles centrales. Siempre se define solo una subArea A por ventana de interrogación. Tamaño nominal: 32 x 32

subArea B: es cada una de las posibles porciones interna de Area B con identicas dimensiones que subArea A. A cada subArea B se asocia un vector (x,y) llamado ***corrimiento de muestreo***. Cantidad nominal: 81 subAreas B (de 32 x 32 cada una)

B. Modo de transmisión de un cuadro de video según el estándar ITU-R BT.605



Referencias de tiempo (TRS)

P9	P8	P7	P6	P5	P4	P3	P2
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	F	V	1(H)	E3	E2	E1	E0

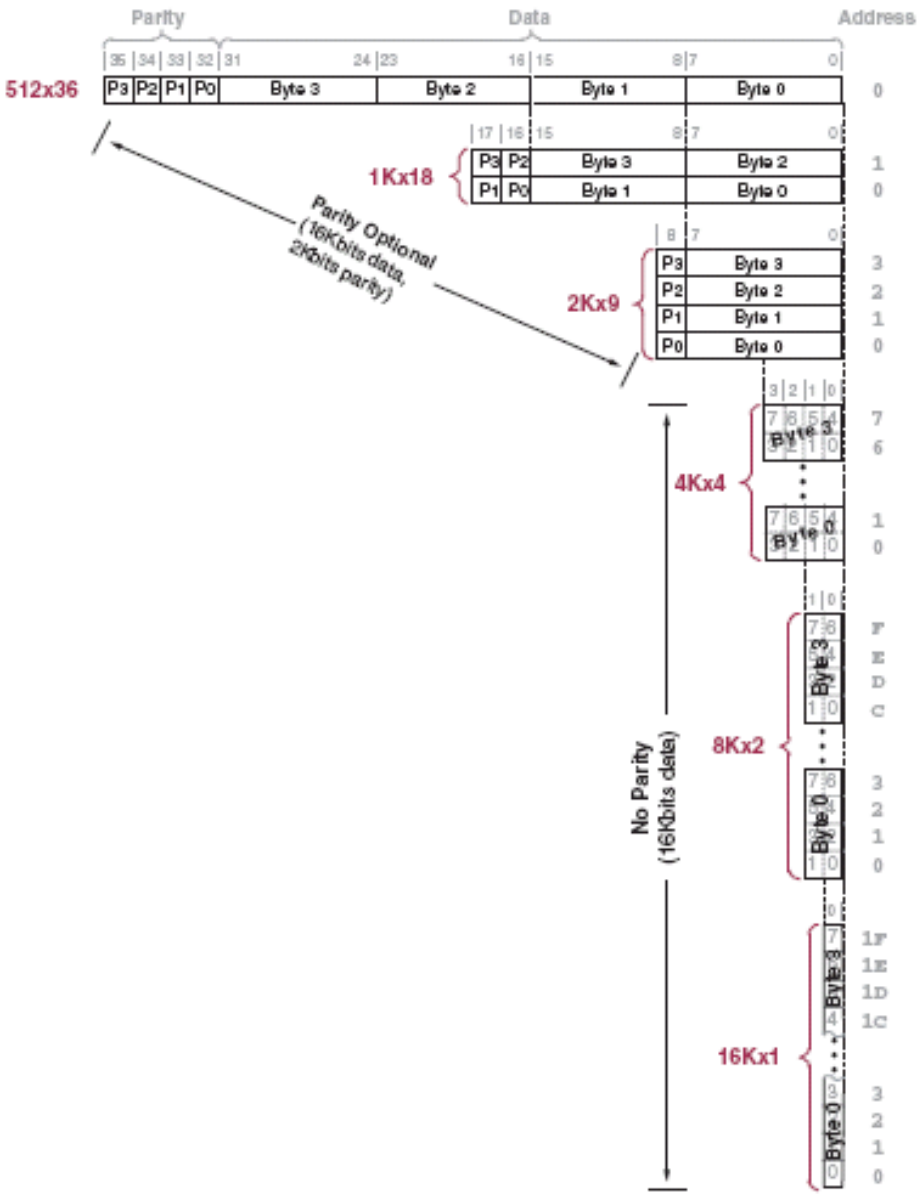
Fin de video activo: EAV

P9	P8	P7	P6	P5	P4	P3	P2
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	F	V	0(H)	E3	E2	E1	E0

Inicio de video activo: SAV

donde
E3 = V xor H
E2 = F xor H
E1 = F xor V
E0 = F xor V xor H

C. Configuraciones de las memorias en bloque (BRAM) en una FPGA Spartan-3E



(fuente: hoja de datos FPGA Spartan-3E Family)