Esta tesis doctoral contiene un índice que enlaza a cada uno de los capítulos de la misma.
Existen asimismo botones de retorno al índice al principio y final de cada uno de los capítulos.

Ir directamente al índice

Para una correcta visualización del texto es necesaria la versión de Adobe Acrobat Reader 7.0 o posteriores

Aquesta tesi doctoral conté un índex que enllaça a cadascun dels capítols. Existeixen així mateix botons de retorn a l'index al principi i final de cadascun deis capítols .

Anar directament a l'index

Per a una correcta visualització del text és necessària la versió d' Adobe Acrobat Reader 7.0 o posteriors.

TESIS DOCTORAL

MODELO PARAMÉTRICO DE ARQUITECTURA PARA LA GENERACIÓN DE PRIMITIVAS COMPUTACIONALES

UNIVERSIDAD DE ALICANTE

TESIS DOCTORAL

MODELO PARAMÉTRICO DE ARQUITECTURA PARA LA GENERACIÓN DE PRIMITIVAS COMPUTACIONALES

Presentada por
María Teresa Signes Pont

Dirigida por
Dr. Juan Manuel García Chamizo

Departamento de Tecnología Informática y Computación

Mayo de 2005

A Teresa

A José Tomás

A la memoria de mis padres

AGRADECIMIENTOS

Me toca escribir ahora la página más difícil de este documento. ¿Cómo resumir en unas pocas palabras las sensaciones, reflexiones, sentimientos, enormemente diversos y tal vez contradictorios, que he podido acumular a lo largo de estos años? ...Pero, tal vez interese menos valorar los pormenores de un proceso largo y complejo, como es la realización de una investigación y la escritura de una tesis, que apreciar un resultado, sobre todo cuando éste ha merecido la pena. Añadiré, además, que este logro no es sólo cosa de esta doctoranda.

Deseo expresar mi profundo agradecimiento al director de mi tesis, Dr Juan Manuel García, por haber confiado en mí y haberme guiado por el camino correcto. Gracias, Juanma, por tu dedicación, tu respaldo, tus múltiples enseñanzas y... por tu infinita paciencia.

A Jero, Higinio y Goyo, amigos y conocedores de las convoluciones: gracias por vuestra valiosa ayuda en el trabajo científico, así como por el apoyo anímico que me habéis prestado.

No olvido la contribución de mis compañeros del grupo de investigación: Jorge, Paco Flórez, Andrés, Javi, Joan Carles, Dani, José Luís, Toni, Paco Pujol, Mora, José García, Antonio Soriano, Paco Maciá, Vicente, Ana, Juan Antonio y David, así como Andrés Almela y Anabel. A todos vosotros os agradezco vuestras aportaciones y vuestra amabilidad.

A José Tomás y a mi pequeña Teresa, ¿qué deciros? ¡Sois mi vida!

Al final, pero sobre todo, a la memoria de mis padres.

RESUMEN

Esta investigación está orientada a la mejora de prestaciones de los computadores. Dentro de las posibles estrategias encaminadas a proporcionar una respuesta, la línea que se propone es la de buscar técnicas operatorias sistemáticas que instrumenten, al nivel de primitivas, operaciones que las arquitecturas convencionales proporcionan como de alto nivel. Más concretamente, el problema que se plantea es el de evaluar funciones con la máxima especificidad, esto es, con el nivel de derivación más bajo posible.

Se aborda la búsqueda de las primitivas adoptando inicialmente la convolución de funciones como método de evaluación de éstas. Se proporciona así una solución de partida de propósito general que tiene un alto grado de derivación. La solución final se obtiene convirtiendo la convolución en una operación recursiva que tiene la forma de suma ponderada. Desde el punto de vista de la algoritmia de obtención de los cálculos, el método consiste en obtener un nuevo valor de la función en cada paso de la iteración a partir del valor que tenía en el paso de iteración precedente. Los factores de ponderación son de gran importancia: son caracterizadores de la función evaluada. Ello puede interpretarse como que sus valores constituyen una expresión condensada de la lógica relacional algebraica presente en la expresión explícita de las funciones de partida.

Los criterios de utilización del paso de iteración abren la vía para la incorporación, de manera intrínseca, de paralización de la operatoria. Se puede operar en el extremo de la secuencialidad estricta, con un paso de iteración básico, o utilizar pasos múltiplos del paso básico para obtener valores semilla de iniciación a partir de los cuales se lanzan hilos paralelos de cálculo; esto último puede hacerse a varios niveles de profundidad.

Se ha aplicado este método con éxito en el caso de las rotaciones y de un grupo de transformaciones configurables como rotaciones.

Se han mostrado también ejemplos de funciones cuya evaluación no es abordable a nivel de primitivas, adelantando de forma empírica, de momento, su posible evaluación con más niveles de derivación.

RESUM

Aquesta investigació està orientada cap a la millora de prestaciones dels computadors. De totes les estratègies possibles encaminades a satisfer aquest objectiu, la línia seguida és la de buscar tècniques operatòries sistemàtiques que instrumenten a nivell de primitives les operacions que les arquitectures convencionals proporcionen com de alt nivell. Concretament, el problema plantejat és el d'avaluar funcions amb la màxima especificitat, i açò significa avaluar-les amb el nivell de derivació més baix possible.

La busca de les primitives s'aborda fent servir inicialment la convolució de funcions com un mètode per a avaluar-les. Així es proporciona una solució de partida que implica un nivell de derivació elevat. La solució final s'obté convertint la convolució en una operació recursiva que té forma de suma ponderada. Des del punt de vista de l'algorítmia que permet l'obtenció dels càlculs, el mètode consisteix en obtenir un nou valor de la funció en cada pas de la iteració, a partir del valor que tenia en el pas de la iteració precedent. Els factors de ponderació ténen gran importància: són els que caracteritzen la funció avaluada. La interpretació que pot fer-se d'açò és que els valors dels factors són l'expressió condensada de la lògica relacional algebraïca palesa en l'expressió explícita de les funcions de partida.

Els criteris d'utilització del pas d'iteració obrin la via per a incorporar, de manera intrínseca, la paralel·lelització de l'operatòria. Es pot operar seguint la seqüencialitat estricta, amb un pas d'iteració bàsic, o utilitzant pasos múltiples del bàsic per a calcular uns valors que iniciaràn linies paral·leles de càlcul. Aquesta paralel·lelització pot repetir-se formant un arbre cada vegada més profund.

Aquest mètode s'aplica amb èxit en el cas de les rotacions i en un grup de transformacions configurables com a rotacions.

Es mostren també exemples de funcions no avaluables directament per les primitives i se'n proposa una solució, empírica de moment, que involucra més nivells de derivació.

ABSTRACT

This research is oriented to computer performance improvement. Among the whole range of strategies that can lead to this objective, the proposal approach deals with systematic operating technics which have the capability to implement at the primitive level the same operations that conventional architectures provide as high level ones. Particularly, the problem raised aims the maximum specific function evaluation, which means achievement of the lowest level of derivation possible.

The search of primitives is tackled by developing function evaluation by means of convolution. This way, a general purpose starting point solution with a high degree of derivation is provided. The final solution is obtained by converting the convolution into a recursive operation shaped as a weighted sum. From an algorithmic point of view, the calculation method consists in obtaining a newer value at every iteration step by using a former calculated function value in a previous iteration step. The weighting parameters are quite significant, as they characterize the resulting function. This fact can be understood as if the whole algebraic relational logic laying in the formal expression of starting point functions was embedded in parameter values.

Criteria on using iteration step open paths for an intrinsic incorporation of operation parallelization issues. This way, it is possible to operate in both

extreme strict sequentially border way, using a basic iteration step, and in multiple basic step way in order to obtain initialization seeds for which parallel calculating threads can be run. This latter fact can be achieved by using multiple depth levels.

The method proposed has been successfully used for performing rotations and also for a set of transforms that can be configured as rotations.

Some function examples which cannot be evaluated at the primitive level are also shown. For these cases an empiric solution has been suited that introduces some principles for future research of derivative solution of the method.

TABLA DE CONTENIDOS

LISTA DE FIGURAS

Volver al índice/Tornar a l'índex

LISTA DE TABLAS

Volver al índice/Tornar a l'índex

Capítulo 1

INTRODUCCIÓN

1 Motivación y objetivos

El procesamiento de información es un fenómeno ubicuo e inherente a la dinámica de los sistemas. Se identifican distintos estilos de computar entre los que destacan aquéllos que han sido ideados por el ser humano para resolver problemas concretos y los que caracterizan los sistemas naturales, físico-químicos o biológicos, como resultado de su interrelación con el entorno.

El grado de consolidación de las ideas y conceptos en disciplinas como la física y las matemáticas junto con el nivel de desarrollo tecnológico alcanzados en el siglo XX han propiciado la emergencia de la informática, que engloba en su cuerpo de conocimiento cuestiones relacionados con el procesamiento de información. A pesar de lo moderno de la disciplina como tal, la preocupación por el tema, tanto a nivel conceptual como de realización, no es nueva. Se sabe que en el año 450 A.C., Sócrates ya preguntaba a Anaxágoras por "...un procedimiento efectivo de cálculo...", entendiéndose la pregunta como una alusión a un sistema en el cual, dadas unas entradas, se obtendría sistemáticamente un resultado. Se tiene constancia igualmente de un dispositivo antiguo que sistematiza el cálculo: el ábaco, inventado por los egipcios en esa misma época y perfeccionado en China a principios del siglo segundo D.C. Su vigencia es tal que, en 1945, en un concurso en el que competían un ábaco y una calculadora, en lo que respecta a la velocidad

y a la exactitud de los cálculos, ganó el ábaco [Cardona, 1996]. A lo largo de la historia, la búsqueda de soluciones a problemas concretos de cálculo ha proseguido, cristalizando en implementaciones consonantes con la tecnología del momento, hasta que un volumen suficiente de conocimiento empírico ha acabado generando método. De la interrelación entre el método y las múltiples resoluciones de los problemas concretos emerge una vía enormemente productiva que explica la situación actual: los avances conseguidos y la velocidad a la que se siguen produciendo.

La evolución ocasionada por el refinamiento de los métodos de cálculo y de las realizaciones propicia una diversificación de los paradigmas de computación, todos ellos animados por un objetivo común que es el de resolver cada vez mejor lo que abordan. Desde la arquitectura de computadores, la noción de rendimiento es indisociable de la idea de bondad. Las CPUs convencionales implementan primitivas como la suma y la multiplicación, realizadas bit a bit de forma secuencial por los circuitos electrónicos. Esta base física sostiene una jerarquía de niveles de computación que se materializan por lenguajes, cada vez más sofisticados, que van desde expresar las operaciones de forma muy elemental, en términos de transferencia de datos a registros o desde ellos (ensamblador), hasta la expresión más refinada representada por la forma algebraica concisa que utiliza variables (lenguajes de alto nivel). Cada nivel de abstracción oculta detalles al nivel superior y, de la diversidad de realizaciones de cada nivel, se desprenden ejecuciones más o menos productivas de una misma computación. Existen técnicas reconocidas de mejora del rendimiento como son la segmentación [Schwarz, 1996], [Beaumont-Smith et al, 1998] y la anticipación [Schmookler, 2001], [Lang, 2004], que aumentan la rapidez de respuesta, o la compartición que reduce la cantidad de recursos [Tan, 2003], generalmente, un compromiso razonado entre todas ellas es el que acaba proporcionando soluciones realistas.

En las arquitecturas actuales, una instrucción de lenguaje de alto nivel implica muchas instrucciones de lenguaje máquina y cada una de éstas, a su vez, pone en juego gran cantidad de sumas y desplazamientos a nivel de la electrónica digital de base. Cabe pensar que bajar hasta un nivel de operatoria trivial como la suma y, más propiamente, al de las primitivas al uso para resolver, pueda no ser la mejor solución. Tal vez, un planteamiento en el cual el nivel más bajo fuera ya más potente que el nivel de las primitivas hardware habituales, redundaría en una mejora global del rendimiento.

Dentro de las posibles orientaciones encaminadas a proporcionar una respuesta, la línea que se propone es la de buscar métodos de operar que instrumenten, al nivel de primitivas y para procesadores de propósito general, operaciones que, por tener cierta complejidad, han sido implementadas tradicionalmente como operaciones de alto nivel. Con ello se prevé una repercusión notable en la dinámica de ejecución de los cálculos dado que, la obtención de una potencia de cálculo equivalente deberá requerir menor número de niveles. Asimismo, estas arquitecturas tendrán presumiblemente implicaciones en la línea de facilitar la formulación, el modelado y el cálculo de problemas que actualmente se resuelven con dificultad apreciable. Por poner un ejemplo, en la actualidad, la simulación de la respuesta en forma de trenes de pulsos de alta frecuencia que tienen las neuronas biológicas ante estímulos pulsátiles, se realiza concibiendo la neurona como un complejo circuito electrónico y generando la respuesta como la salida funcional de dicho circuito. Pensar que la interacción neuronal en un sistema nervioso es, a su vez, una función compleja de la respuesta de cada neurona sugiere que la tecnología actual está lejos de la potencia de un sistema nervioso si el planteamiento sigue siendo de síntesis funcional a partir de las primitivas computacionales.

El planteamiento que aquí se hace recuerda en cierta medida, a los pasos seguidos en computación gráfica, esto es, implementar al nivel más bajo operaciones que

en el pasado se obtenían por síntesis de otras más simples. Las ventajas son de todos conocidas.

El objetivo de este trabajo es contribuir a la mejora del rendimiento de los procesadores. Generalmente, las mejoras suelen producirse en los casos en que la resolución de un problema se hace de forma dedicada, a la medida de sus características; éste es el principio sobre el que se basan los procesadores de propósito específico. En cambio, en los procesadores de propósito general, las mejoras dependen de un equilibrio entre varias de sus prestaciones y, por tanto, conseguir mejores rendimientos plantea un reto de mayor envergadura. Esta investigación se sitúa en la línea de aumentar el rendimiento buscando al nivel del hardware modos de operar capaces de implementar funciones que otras arquitecturas proporcionan como de alto nivel y ello para propósito general. Este objetivo general se desglosa en objetivos más específicos que condicionan, orientan y organizan el trabajo. A continuación se enuncian estos objetivos.

- Proponer una modelización de la operatoria que permita diseñar primitivas computacionales implementables a nivel de hardware, cuya potencia expresiva sea superior a la que proporcionan las primitivas habituales. Generalmente las máquinas organizan la computación siguiendo un esquema de niveles ordenados jerárquicamente y cada nivel contiene operaciones de mayor potencia que los niveles precedentes. La propuesta va en la línea de iniciar la jerarquía a partir de un nivel más alto que el habitual, que es el de las primitivas suma y multiplicación, instrumentadas por el hardware de la máquina. Así, las primitivas que hay que diseñar para ser implementadas por ese primer nivel de hardware deberán proporcionar operaciones más sofisticadas que la suma y la multiplicación.

- Formular la base conceptual del modelo de operatoria. Conseguir operaciones de alta potencia expresiva al nivel de las primitivas obliga a que los métodos para calcular sean suficientemente sencillos. Evaluar o medir consiste en realizar una apreciación o estimación de una cantidad tomando otra, de la misma especie, como unidad. También puede entenderse como una manera de "ver" una cantidad, un objeto, una función,..."a través" de otra cantidad, objeto o función, respectivamente. La convolución proporciona la cobertura conceptual a la metodología propuesta.

- Pasando al nivel de realización, diseñar una arquitectura que implemente el modelo computacional propuesto y valorar su utilidad para la resolución de algunos problemas, proporcionando asistencia complementaria o alternativa a la propuesta existente.

La memoria que sustenta este trabajo se organiza en seis capítulos.

El capítulo primero, de introducción, está dedicado a expresar las motivaciones que han llevado a la realización de la investigación recogida en este documento, a exponer los objetivos propuestos y a describir y analizar el conocimiento actual en torno al tema tratado, finalizando con una definición del problema planteado y su propuesta de resolución.

Siguiendo al capítulo de introducción, el capítulo segundo desarrolla el Método de evaluación Recursiva Basado en Convolución (CBRM). Después de revisar algunas nociones de la teoría de la medida y de la teoría de representación de grupos se propone la convolución de funciones como método de evaluación. El objetivo de rendimiento computacional hace buscar una operatoria con menor coste que el cálculo directo de la convolución. Para ello se regulariza la convolución sobre una operación recursiva paramétrica que calcula los puntos de

las funciones por iteración. Seguidamente, se definen las primitivas propias de este tipo de operatoria como base para proponer un procesador.

El capítulo 3 se ocupa del diseño y evaluación de la arquitectura asociada al Método de valuación Recursiva Basada en Convolución (CBRM). Se describen los módulos funcionales de la arquitectura CBRM facilitándose para el módulo de cálculo estimaciones del tiempo de cálculo y del espacio ocupado, tanto en unidades independientes de implementación como a partir de la simulación en una plataforma de lógica reconfigurable. La posibilidad de introducir paralelismo en algunas etapas de la arquitectura CBRM sugiere distintas vías de mejora de las prestaciones. El capítulo concluye con diversas propuestas de paralelización, con el estudio correspondiente de rendimiento.

El capítulo 4 trata algunas aplicaciones del CBRM. Se aborda en primer lugar la evaluación a nivel de primitivas de las rotaciones como funciones relativamente sencillas pero pilares básicos de un gran número de transformadas matemáticas, éstas últimas con presencia destacada en diversos ámbitos de la ciencia y la ingeniería. Se propone la transformada de Hough como ejemplo de función basada en rotaciones y se realiza una comparación entre el CBRM y otras propuestas en relación con el tiempo de cálculo y el área ocupada. Se estudian igualmente las transformadas ortogonales con la propuesta de un patrón común de cálculo, basado en rotaciones, para todas ellas. En este caso, actúa como paradigma la transformada de Fourier. Finaliza el capítulo cuarto con el estudio de la evaluación de la transformada de Fourier por el CBRM y por otros métodos, facilitándose una comparación en términos de área ocupada y de tiempo.

En el capítulo 5, se aborda el problema de las funciones no evaluables a nivel de primitivas por el CBRM. Como ejemplo, se presentan algunos comportamientos de difícil formalización proporcionados por subsistemas neuronales. Se adelanta una solución empírica cuyo propósito es hacer entrever la capacidad del CBRM

para estos casos con unos resultados iniciales apreciables, razón por la cual se deja esta cuestión como línea detrabajo futuro.

El capítulo 6 es el de las conclusiones. Primeramente se recuerdan las aportaciones de esta investigación, entre las cuales destaca especialmente el papel relevante que se otorga a la convolución. Se sugieren además algunas líneas de trabajo futuro como robustecer el CBRM ampliando su campo de aplicación.

2 Conocimiento actual y problemas abiertos

En primer lugar se revisan las aportaciones más relevantes en la aritmética del computador donde aparecen principalmente dos orientaciones: por una parte, la mejora en la implementación de las operaciones primitivas básicas habituales, suma y multiplicación y, por otra, la propuesta de evaluación de funciones desde el hardware de la máquina, basándose en dichas primitivas. En un segundo tiempo, teniendo en cuenta los objetivos que se propone esta investigación, he considerado necesario estudiar el caso de algunas funciones sofisticadas que se implementan como primitivas para procesadores especializados en otros campos de la informática. Aunque no únicos, los principales campos tratados en relación con esta cuestión y mencionados en esta memoria son la morfología matemática dentro del análisis de imagen y la computación gráfica. Seguidamente, se hace una breve reseña de los hitos destacables en el campo de la computación cuántica, teniendo en cuenta los aspectos más relacionados con el tema central de esta investigación, que son los métodos y las arquitecturas dedicados a la evaluación de funciones. Para finalizar este apartado se mencionan los modelos computacionales más

importantes utilizados en el campo de la neurociencia, ya que existe también la motivación de abordar en esta investigación la reproducción de las funcionalidades que manifiesta el tejido vivo.

2.1 Aritmética del computador

La norma IEEE-754 para coma flotante establece un esquema discreto para la representación de los números reales en un computador [Cornea-Hasegan, 1999], [Schwarz,1999], [Schwarz,2003], [http://cch.loria.fr/documentation /IEEE-754/index.html]. Dicha norma estructura la representación del número real en tres componentes: signo, mantisa y exponente y establece la precisión de las operaciones básicas que todo procesador debe implementar; estas operaciones son la suma, la multiplicación, la división y la raíz cuadrada. Por ser primitivas, la suma y la multiplicación son las operaciones que se efectúan más frecuentemente sobre operandos reales.

Para la suma, es habitual concentrar el esfuerzo de mejora en el tratamiento de las mantisas debido a que es la parte que entraña más complejidad ya que requiere desplazamientos de normalización y alineamiento, complemento y redondeo. El tratamiento del exponente se realiza en paralelo con el cálculo anterior y se reduce a desplazamientos y sumas de muy pocos bits [Obermann, 1997]. Cada operación se completa con una etapa de redondeo [Even, 2000], [Parks, 2000]. La segmentación es el método más comúnmente empleado para aumentar la cantidad de sumas por unidad de tiempo, pero esto no reduce necesariamente el tiempo de latencia del algoritmo. La reducción de la latencia suele venir dada por la realización de operaciones en paralelo, casi siempre a costa del empleo de hardware adicional [Quach y Flynn, 1990], [Quach y Flynn, 1992], [Kantabutra, 1993], [Dadda, 1996], [Bruguera 2001]. Las propuestas más novedosas profundizan en el paralelismo y en el desarrollo de esquemas de latencia variable aprovechando que no todos los componentes del sumador son

necesarios con determinados operandos de entrada, reduciéndose así la latencia media [Beaumont-Smith et al. 1998], [Takagi y Horiyama, 1999], [Cheng et al, 2000], [Kalampukas et al, 2000], [Um y Kim, 2001].

La multiplicación, consta de la suma de varios productos parciales desplazados, cada uno de los cuales, en su forma más sencilla, es un producto del multiplicando por un dígito del multiplicador [Callaway y Schwarzlander, 1997], [Stelling et al., 1998]. El algoritmo se realiza en tres etapas: generación de los productos parciales, reducción el número de productos y suma final con propagación del acarreo. El aumento de prestaciones puede realizarse incidiendo en cada una de las etapas mencionadas. La construcción de los productos parciales puede realizarse mediante un conjunto de puertas AND que operan sobre cada dígito del multiplicando y el multiplicador, de forma clásica o bien utilizando algoritmos avanzados como el de Booth [Booth, 1951] o alguna de sus variantes [Altwaijry y Flynn, 1995], [Goto, 1997], [Yeh y Jen, 2000]. Otros autores proponen técnicas de generación de los productos parciales que obtienen ventajas recodificando los operandos antes de la generación [Law et al, 2000], [Skin y Jeon, 2000], [Seidel, 2001]. Después de la generación de los productos parciales, los multiplicadores de alto rendimiento no suman directamente, sino que combinan los productos parciales, reduciendo su número hasta un número total de dos, que se suman al final. La combinación mencionada se realiza por circuitos contadores [Wallace, 1964], [Dadda, 1965] o compresores [Weinberger, 1981], [Song y De Michelli, 1991], [Kwon et al, 2000], [Bickerstaff, 2001], basados en elementos de suma sin acarreo [Bewick y Flynn, 1992], [Omondi, 1994], [Oklobdzija et al, 1996], [Choi et al., 1997], que son conectados entre sí formando una topología determinada, por ejemplo, en árbol [Ta kagi et al, 1985].

En la suma final con propagación de acarreo, los esquemas empleados son los que ya se han mencionado en el párrafo dedicado a la suma, pudiéndose emplear también algoritmos de suma entera.

El cálculo de funciones es de enorme importancia para aplicaciones en áreas diversas de la ciencia y la ingeniería, entre las que destacan, por ejemplo, el procesamiento digital de la señal, la computación gráfica y científica y las aplicaciones para tiempo real. En la actualidad, los planteamientos relativos al cálculo de funciones son de dos tipos: los de propósito general, que pretenden dar una respuesta global al problema: entre ellos, destaca el algoritmo *CORDIC* (*CO*ordinate *R*otation *DI*gital *C*omputer) [Volder, 1959], [Walther, 1971] [Haviland, 1980], [Nakayama, 1989], como técnica iterativa que sólo requiere operaciones de suma y desplazamiento y, por otra parte, las propuestas especialmente pensadas para aumentar las prestaciones en el cálculo de alguna función o grupo de funciones en particular. Al respecto, se puede citar como ejemplo la aritmética asociada al sistema logarítmico [Coleman, 2000], [Chichyang, 2000] que mejora la eficiencia en el cálculo de funciones como las funciones trigonométricas. También existen algoritmos con menos peso conceptual, especialmente pensados para el cálculo de las funciones elementales más habituales (inversa, raíz cuadrada, inversa de la raíz cuadrada...). Cada uno de estos algoritmos ha dado lugar a una gran variedad de implementaciones hardware, con resultados altamente competitivos debido a los avances recientes de la tecnología VLSI [Koren, 1993], [Ercegovac et al, 2000a], [Ercegovac et al, 2000b].

A continuación se revisan los métodos de cálculo de funciones más relevantes en la actualidad.

Algoritmo *CORDIC*

Introducido por Volder en 1959 para la evaluación de funciones trigonométricas y la conversión de coordenadas rectangulares a coordenadas polares, el *CORDIC* fue generalizado y unificado por Walter en 1971 para obtener además funciones hiperbólicas y lineales. Es el algoritmo más comúnmente empleado en los coprocesadores clásicos (I8087, HP35, M68881, M68882...). En los

últimos años, este algoritmo ha dado lugar a un intenso desarrollo de múltiples implementaciones hardware para aplicaciones específicas, como procesamiento digital de la señal y de la imagen [Ahmed, 1990], [Hu, 1992], álgebra matricial [Cavallaro, 1988], [Cavallaro, 1991], [Ercegovac, 1994] robótica, aplicaciones gráficas y simulación, que demandan alta velocidad y precisión. Existen diferentes arquitecturas que implementan el algoritmo *CORDIC*. La arquitectura deseable es, naturalmente, la que consigue el mejor compromiso en cuanto a área y velocidad para la aplicación requerida.

La arquitectura *CORDIC*-iterativa es la que procesa con una cadencia igual al producto del número de iteraciones por el tamaño de los datos. El diseño bit-paralelo, que traduce al hardware las tres ecuaciones del *CORDIC*, tiene un número de entradas elevado y, por tanto, suele ser lento en las plataformas de lógica reconfigurable. En cambio, el diseño bit-serie es mucho más compacto y rápido debido a que usa aritmética serie: ésta representa un tipo de interconexión y una lógica simplificadas aunque tenga que utilizar tantos relojes como precisión tienen los sumadores.

La arquitectura *CORDIC*-on line es la que procesa las iteraciones de forma desarrollada, esto es, haciendo que haya un elemento de procesamiento por cada iteración [Wang, 1996]. Este modo aporta dos simplificaciones significativas, que son la posibilidad de cablear los desplazamientos (uno fijado por cada desplazador) y los ángulos (uno por cada sumador). Ello repercute en un ahorro importante de memoria, haciendo que esta arquitectura resulte finalmente en una cadena de sumadores-restadores interconectados. Este circuito enteramente combinacional puede segmentarse fácilmente insertando registros entre los sumadores. Este diseño puede transformarse en bit-serie reemplazando cada sumador por un sumador serie, separándolos por registros de desplazamiento de tantos bits como la precisión de los datos.

Conseguir altas prestaciones requiere disponer de varios procesadores bit-serie procesando en paralelo o bien montar un cauce paralelo de procesadores desarrollados.

En la última década, el algoritmo *CORDIC* ha motivado numerosos trabajos de investigación relativos a propuestas diversas (aumento del radix, técnicas de compensación del factor de escala, aplicaciones del algoritmo tridimensional, implementaciones para tarjetas de lógica reconfigurable..), que permiten cada vez mejores prestaciones [Bruguera, 1993], [Antelo, 1996a], [Antelo, 1996b], [Villalba, 1996], [Antelo, 1997a], [Antelo, 1997b], [Saéz, 1998], [Villalba, 1998], [Antelo, 2000].

Sistema Logarítmico

Frente a las limitaciones que impone la ejecución en punto flotante, está la propuesta de una aritmética basada en la representación logarítmica de los números con el objetivo de aumentar la eficiencia en el cálculo de las funciones trigonométricas [Das et al.1995] y sus inversas. Otra ventaja de este sistema es su precisión en las operaciones aritméticas: el error de la representación es menor que el del punto flotante así como el error de cuantización [Coleman, 1999]. El inconveniente del método está en que la implementación de la suma y la resta precisa unos términos auxiliares. Éstos pueden precalcularse y almacenarse en una tabla look-up, a la que se ha de acceder, o calcularse durante el proceso. El tamaño de las tablas crece exponencialmente con el tamaño de la palabra. Se han propuesto diversas técnicas de reducción del tamaño de la tabla [Lewis, 1990], [Chen, 1998], [Coleman, 2000], se ha recurrido a la aritmética redundante [Arnold et al 1990], a la representación semi-logarítmica [Muller, 1998] o también a la solución híbrida, que consiste en realizar la suma y la resta en punto flotante y la multiplicación y la división en sistema logarítmico [Lai, 1993]. La ventaja del cálculo es en cuanto a coste [Chichyang, 2000]. El cálculo, que consiste en evaluar iterativamente y de

modo paralelo una función exponencial y una función logarítmica, se realiza de forma segmentada con un cauce de N etapas [Arnold, 2003]. Las fuentes de error son, por una parte, el error debido al algoritmo de normalización, y por otra, el error de redondeo. En total, tomando como referencia de exactitud los valores de las funciones proporcionados por las librerías matemáticas del lenguaje C, el sistema logarítmico acarrea un error de aproximadamente 2^N en el caso de la doble precisión. El sistema logarítmico también motiva investigación de tipo aplicado [Paliouras, 2001], [Paliouras, 2002], [Arnold, 1990], [Arnold, 2001], [Arnold, 2002a], [Arnold, 2002b], [Arnold, 2003], [Dimitrov, 2001].

Algoritmos para el cálculo algunas funciones elementales

En este apartado se describen distintos algoritmos de cálculo de funciones elementales que cuentan en la actualidad con gran cantidad de propuestas de implementación hardware. Entre ellos destacan por su importancia aquellos que tratan del cálculo de la función inversa, seguidos por los de la raíz cuadrada e inversa de la raíz cuadrada.

Las propuestas que tratan del cálculo de la función inversa pueden separarse en cinco grandes grupos: de recurrencia digital, iteración funcional, de radix muy alto, de acceso a tablas look-up y de latencia variable [Obermann y Flynn, 1997], [Kuhlmann y Pahi, 1998], aunque lo habitual es encontrar combinaciones de dos o más de ellas con el propósito de aunar ventajas.

La implementación más común del cálculo de la inversa utilizando recurrencia digital [Ercegovac y Lang, 1994], [Lang, 2001] es la división SRT (Sweeny, Robertson, Tocher) que presenta un buen compromiso entre rapidez de cálculo y coste en términos de área ocupada [Montuschi y Cimiera, 1993], [Montuschi y Cimiera, 1994], [Harris et al, 1997], [Kornerup, 2003], [Mc Can, 2003].

Los métodos basados en iteración funcional utilizan la aproximación de Newton-Raphson, [Flynn, 1970], [Schulte, 1994], la interpolación polinómica o el algoritmo de Goldschmidt [Ercegovac et al, 2000a]. El algoritmo de Newton-

Raphson es un método iterativo con convergencia cuadrática, que consiste en buscar un valor inicial aproximado que se va refinando en cada iteración por multiplicación por un factor que mide la proximidad entre la aproximación actual y el valor exacto. Si el valor inicial se elige suficientemente cerca del valor exacto, se requieren menos iteraciones. Este requisito suele conseguirse habitualmente empleando tablas.

En el acceso a tablas, como método complementario a la iteración funcional, la tabla se utiliza para almacenar las semillas de inicio de las iteraciones, de forma que se reduzca el número de iteraciones del algoritmo. Por ejemplo, el método propuesto por Wong y Goto [Wong, 1995] consiste justamente en hallar este valor inicial direccionando la tabla de búsqueda por la mitad más significativa de los bits de la mantisa del número que hay que invertir. El método de Ito, Tagaki y Yajima [Ito, 1997] encuentra el valor inicial por aproximación lineal, buscando los dos coeficientes en una tabla direccionada por los m bits más significativos y, en vez de realizar una multiplicación y una suma, propone una modificación que consiste en sustituir las dos operaciones por una única multiplicación que se efectúa sobre un operando modificado. Otras opciones relevantes son las de Ball y Ercegovac [Ball y Bojanic, 2000], [Ercegovac et al, 2000b].

La aproximación utilizando tablas particionadas simétricas se basa en el desarrollo en serie de Taylor alrededor de un punto particionado en tres porciones, con longitudes en bits diferentes [Schulte, 1997]. El desarrollo se limita a los dos primeros términos que son los que se almacenan en la tabla bipartita. El primer término representa el valor de la función en la porción más significativa del punto, mientras que el segundo término representa el valor de la derivada de la función en la porción menos significativa. La ventaja de este método en cuanto al ahorro de memoria se debe a que, en los términos que hay que almacenar, unos son el complemento a uno de otros. Esta aproximación es aplicable a muchas funciones (logaritmo, raíz cuadrada, inversa,...) sólo con

variar el intervalo inicial de definición [Matula, 2001] y puede desarrollase para tablas multipartitas [de Dinechin, 2001].

Los métodos de interpolación (polinómica, racional, basada en el desarrollo de la función en serie de Taylor) precisan un cálculo de coeficientes que puede realizarse por métodos distintos (Lagrange, Stirling, Tchebyshev,...). En este sentido, la aproximación polinómica es un caso particular de aproximación racional [Koren, 1990]. Incluye tres etapas: la reducción del argumento a un intervalo de aproximación predeterminado, la evaluación de la aproximación racional del argumento y la obtención del resultado final. Los coeficientes de la aproximación se determinan de forma que minimicen el error relativo máximo cometido en el intervalo. Si se particulariza para un polinomio de segundo grado, la interpolación se basa en los puntos extremos y el punto medio del intervalo con la posibilidad de calcular los coeficientes durante la ejecución, a partir de los valores tabulados de puntos de la función, o bien de precalcularlos y almacenarlos. La primera opción es más ventajosa en términos de memoria (un tercio menos) en detrimento del aumento de hardware y de tiempo de ejecución empleado en el cálculo de los coeficientes. [Cao, 1997]. Existen propuestas más recientes que equilibran notablemente este compromiso [Cao, 2001].

Con respecto a los métodos de radix alto, la propuesta es bastante amplia [Tenca y Ercegovac, 1998], [Lang y Montuschi, 1999], [Montuschi y Lang, 2001], [Matula, 2003], [Pineiro, 2003].

Entre los algoritmos de latencia variable destacan los trabajos de Williams y Kuhlmann, [Williams et al, 1995], [Kuhlmann y Pahi, 1998]. También pueden encontrarse referencias a esta metodología en la bibliografía reseñada para otros tipos de divisores, SRT y radix alto, que combinan ambos aspectos.

Los algoritmos para la obtención de la raíz cuadrada son similares a los utilizados para obtener el cociente [Lang y Montuschi, 1999]. Sin embargo, la

repercusión en el rendimiento medio del sistema es menor que éste, por ser una operación, en general, menos frecuente.
La inversa de la raíz cuadrada aparece con frecuencia en aplicaciones gráficas y multimedia y motiva igualmente cierta atención en la línea de los avances en las operaciones de división y raíz cuadrada [Takagi, 2001], [Lang, 2001], [Lang, 2003].

Como resulta patente después de la presentación de esta panorámica, los avances en aritmética del computador se centran en mejorar las operaciones primitivas computacionales habituales, suma y multiplicación. Las operaciones que entrañan más complejidad se relegan habitualmente a niveles más altos, en los cuales el tratamiento algorítmico se hace más específico y, cada vez más, las implementaciones hardware dedicadas mejoran las ejecuciones de dichos algoritmos.

2.2 Procesadores especializados

En la segunda parte dedicada a la revisión del conocimiento actual, se consideran otros dos campos de la computación: el tratamiento digital de imágenes y la computación gráfica. Aunque no sean campos centrales al objeto de esta investigación en cuanto a objetivos ni contenidos, he considerado procedente introducir aquí algunos ejemplos que pueden servir de soporte a las ideas que han fundamentado los planteamientos de esta tesis.

Independientemente de lo específico de las motivaciones y necesidades en cada caso, en la morfología matemática aparecen nítidamente dos constantes: las transformaciones geométricas y la medición [Serra, 1989], [Henk, 1998]. Las transformaciones geométricas tienen como finalidad normalizar los resultados de la observación realizada sobre los objetos. Se realizan en términos de conjuntos de elementos a fin de traducir una apreciación morfológica en otra cuantitativa.

Quizás, el más importante de los nuevos movimientos en el mundo de los gráficos sea el interés por el modelado de objetos y no sólo por la creación de sus imágenes. Además, va en aumento el interés por la descripción de la geometría variable en el tiempo y el comportamiento de los objetos tridimensionales. Así, los gráficos tienen que ver cada vez más con la simulación, la animación y un movimiento de "regreso a la física" en el modelado y la generación de imágenes, a fin de crear objetos que se vean y comporten de la manera más realista posible [Foley et al., 1995], [Chester, 2003].

La morfología matemática en el tratamiento digital de imágenes

La correcta formulación de un problema requiere establecer una secuencia jerarquizada de restricciones. El cuerpo teórico de la morfología matemática establece tres grados de restricciones: los principios de cuantificación, los criterios y los algoritmos. Cuatro principios controlan las transformaciones y las mediciones: invarianza a traslaciones y a cambios de escala, principio de conocimiento local y de semi-continuidad. El significado de los dos últimos principios tiene que ver con la robustez y estabilidad de la transformada necesarias para la operatoria. Los dos primeros principios son más específicos: la invarianza a la traslación restringe la metodología al espacio euclídeo y a las plantillas regulares de puntos. Siguiendo los principios, de menor a mayor especificidad, están los criterios y los algoritmos, respectivamente. Los criterios se refieren a los requerimientos de un problema particular y los algoritmos expresan con precisión las instrucciones a ejecutar para resolverlo.

En la morfología matemática, sobresale por su importancia la transformación de ganancia o pérdida basada en consideraciones geométricas fundamentales, precursora de las operaciones de erosión y dilatación [Matheron, 1967], [Serra, 1969], [Barrera, 1998]. Esta transformación cuantifica genéricamente las

características consideradas de interés en la imagen, cualquiera que sea su procedencia, como el área, las particiones y el recuento de partículas. Ello precisa la definición de un elemento estructurante, que no tiene porqué ser único, y que actúa como "unidad". La transformación que se obtiene de la imagen inicial varía según la configuración que toma el elemento estructurante [Gasteratos, 1998]

Las particularizaciones de la transformación de ganancia o pérdida en el plano operativo conducen a la definición de la dilatación y la erosión, operaciones duales, repetibles de forma iterativa. Las propiedades algebraicas de estas operaciones tienen consecuencias tecnológicas importantes [Hadwiger, 1957], algunas de las cuales cito a continuación. La propiedad distributiva de la erosión (dilatación) con respecto a la intersección (unión) permite fijar una forma ventajosa de mecanizarlas sobre los materiales en la industria. En cambio, la erosión (dilatación) no es distributiva con respecto a la unión (intersección). De ello se desprende la posibilidad de definir una familia Ψ_λ de transformaciones generadas por erosión (dilatación), en la cual el parámetro λ indica el cambio de escala con respecto al cual la transformación es invariante (segundo principio). El orden en la aplicación de erosiones y dilataciones no es indiferente: es más severa la previa aplicación de la erosión [D'Ornellas, 1998]. La aplicación de la suma de Minkovsky a la descomposición del elemento estructurante en un conjunto de elementos estructurantes más simples o más pertinentes, permite realizar erosiones o dilataciones iterativas.

Cabe destacar que el elemento estructurante es crucial a la hora de realizar operaciones morfológicas sobre algún objeto, debido a su poder de condicionar el resultado. Por consiguiente, su elección deberá hacerse en función de los aspectos estructurales del objeto que interesa poner en evidencia. Existiría pues, la necesidad de una teoría que asociara con precisión la geometría de un elemento estructurante con el significado morfológico de la transformación que

provoca. En su lugar, la heurística, la experiencia y las recopilaciones de trabajos realizadas dan unas directrices, a título orientativo, de las características que éste debe tener [Serra, 1965], [Serra, 1978]

Computación gráfica

La computación gráfica desarrolla gran diversidad de métodos que, a su vez, ocasionan aplicaciones de gran variedad. Por ello, no hay principios fundamentales unificados. Por el contrario, cada método desarrolla formalismo, operatoria e implementaciones que le son propios.

En este apartado se consideran algunas aplicaciones que presentan mayor afinidad con el tema que nos ocupa, en la medida en que realizan una computación específica partiendo de primitivas no simples, coherentes con las características del problema a resolver.

En muchas de las aplicaciones de computación gráfica es necesario generar curvas y superficies suaves. Gran número de los objetos reales habituales son inherentemente suaves: formas vivas, accidentes geográficos, piezas mecánicas,... de forma que el modelado geométrico tiene gran relevancia [Gousseau, 2001], [Apu, 2004].

Los modelos basados en curvas cúbicas paramétricas como las polilíneas y los polígonos son aproximaciones por segmentos de curvas de y superficies, respectivamente. A menos que las curvas o superficies que se aproximan también sean lineales por segmentos, hay que crear y almacenar gran cantidad de coordenadas de puntos extremos para lograr una precisión razonable.

Una representación más compacta, que ocupa menos espacio de almacenamiento y ofrece mayor facilidad de manipulación interactiva se basa en el manejo de curvas suaves por segmentos. El método general consiste en emplear funciones que sean de un grado mayor que el de las funciones tratadas, ya que los polinomios de menor grado no ofrecen mucha flexibilidad para

controlar la forma de la curva y que los polinomios de mayor grado pueden introducir ondulaciones no deseadas. Los problemas de continuidad que afectan a la unión suave de las curvas pueden resolverse por distintos métodos: de Hermite [Foley et al., 1995], de Bézier [Bezier, 1970], [Bezier, 1974], método que emplea B-splines uniformes [Bartels, 1987] y no uniformes. Estos métodos tienen prestaciones diferentes en cuanto a diversos aspectos, como la posibilidad de interpolar los puntos de control, que sólo poseen los métodos de Hermite y Bézier; la facilidad de subdivisión, máxima en el método de Bézier y la continuidad, asegurada hasta la segunda derivada en el caso de las B-splines.

Las superficies bicúbicas paramétricas constituyen una generalización de las curvas cúbicas paramétricas (Superficies de Hermite, Bézier, B-spline).

Las superficies cuádricas son muy útiles en aplicaciones especializadas, como el modelado molecular [Porter, 1979], [Max, 1979] y también se han integrado a los sistemas de modelado de sólidos.

El modelado geométrico no es suficiente para representar el mundo real. Muchos fenómenos naturales no encajan en estos modelos. Por ejemplo, la niebla está formada por diminutas gotas de agua, pero no es oportuno usar un modelo en el cual se coloque cada gota, ya que éste no representaría correctamente nuestra percepción de la niebla. Existen técnicas de modelado avanzado que imprimen enormemente más realismo al objeto estudiado. A continuación se exponen en términos cualitativos algunas de ellas.

En fechas recientes, los modelos fractales han llamado mucho la atención [Voss, 1987], [Mandelbrot, 1982] y [Peitgen, 1986]. Cabe citar ejemplos famosos como el copo de nieve de von Koch y los conjuntos fractales de Julia-Fatou y de Mandelbrot. El calificativo fractal se refiere a aquellos objetos generados por procesos infinitamente recursivos, aunque la comunidad

científica acepta como aproximaciones al caso ideal aquellos objetos cuyo proceso es finito, pudiendo exhibir cambios imperceptibles después de alcanzar cierta etapa. La cualidad de dicho proceso es la autosimilitud. Hay que destacar el trabajo de Fournier, Fusell y Carpenter [Fournier, 1982] que desarrollan un mecanismo para generar una clase de montañas fractales con base en subdivisiones recursivas y utilizan igualmente este patrón para modificar formas bidimensionales.

El procesamiento de fractales requiere gran cantidad de recursos. Existen métodos que palían este inconveniente basados en la aplicación de la generación de fractales por línea de barrido, mejorados por Kajiya [Kajiya, 1983] y Bouville [Bouville, 1985]

En cuanto a los modelos gramaticales, Smith [Smith, 1984] presenta un método desarrollado originalmente por Lindenmayer [Lindenmayer, 1968] para describir la estructura de ciertas plantas, usando lenguajes gramaticales (gramáticas L) de grafos paralelos o graftales. Estos lenguajes se describen con una gramática que consiste en una colección de producciones que se aplican todas a la vez.

Estos grafos presentan autosimilitud ya que el patrón descrito por la palabra de la n-sima generación está contenido en la palabra de la (n+1)-sima generación.

La gramática no tiene contenido geométrico inherente, de manera que la utilización de un modelo gramatical requiere una interpretación tanto gramatical como geométrica del lenguaje. Las gramáticas pueden enriquecerse [Prusinkiewicz, 1988], para permitir llevar un registro de "edad" de la letra en una palabra de modo que las letras viejas y jóvenes se transformen de distinta manera. Gran parte del trabajo se ha centrado en la obtención de gramáticas que representen con precisión la biología de plantas durante su desarrollo.

Más allá de cierto punto, una gramática deja de ser manejable como descriptora de plantas, se puede controlar el crecimiento añadiendo una pequeña colección de parámetros descritos en términos biológicos (brote, internodo, brote auxiliar, ramificación, brote apical...) [De Reffye et al., 1988]. Las producciones de la gramática no se aplican de forma determinista sino probabilística.

De esta breve exposición se desprenden algunas líneas de interés con respecto al tema tratado en esta investigación. Dentro del análisis de imagen, la morfología matemática proporciona un método útil para objetivar las observaciones realizadas por observadores con distintos intereses, facilitando una técnica de medición basadas en cuatro principios. Destacan dos operaciones primitivas, la dilatación y la erosión, que se efectúan por medio del elemento estructurante. Los modelos fractales y gramaticales ponen de manifiesto técnicas de construcción recursivas basadas en reglas de producción e iniciadas por una semilla.

2.3 Computación cuántica

La idea de computación cuántica surge en los años setenta a raíz de la reflexión llevada a cabo sobre las consecuencias que iba a tener la ley de Moore en el campo de la computación. La miniaturización de los circuitos y su empaquetamiento cada vez mayor en los chips de silicio lleva a la consideración de que, en algún momento, estos elementos no serán mucho más grandes que un átomo. El problema que emerge entonces es que, a escala atómica, las leyes que han de gobernar el comportamiento y las propiedades de los circuitos ya no pueden ser las de la física clásica sino las de la física cuántica.

A la pregunta de si es posible construir un tipo nuevo de computador basado en los principios de la física cuántica, R. Feynman contesta proponiendo un modelo abstracto dle máquina que muestra cómo un sistema cuántico puede producir computación. Asímismo, explica la capacidad e idoneidad de una máquina

cuántica para simular experimentos de la física cuántica [Feynman, 1982]. En 1985, D. Deutsch publica un artículo, en cierta medida similar al de Turing de 1936, en el que demuestra que cualquier proceso físico puede en principio modelarse en un computador cuántico [Deutsch, 1999], [Deutsch, 2000a], [Deutsch, 2000b].

El bit es la unidad de información en la computación que realizan los computadores digitales. Su valor, que es 0 o 1, se materializa por el estado macroscópico de un sistema físico. En un computador cuántico, el qubit es la unidad de información y no exhibe dos valores diferenciados debido al fenómeno de interferencia cuántica que hace que la unidad pueda almacenar los dos valores a la vez con cierta probabilidad. La consecuencia computacional está en la posibilidad de realizar en un tiempo lineal, n, las operaciones que en un computador clásico requieren tiempo exponencial 2^n. En este sentido, Shor aborda en 1994 el problema de la factorización de grandes números [Shor, 1994].

La estructura de representación en el computador cuántico es el vector de n-qubits que tiene 2^n componentes, cada una con su probabilidad. En el procesamiento, el computador maneja directamente las amplitudes de las probabilidades. La arquitectura del computador cuántico se basa en un conjunto de cuatro operaciones primitivas materializadas por puertas y representables en notación matricial (matriz de operadores multiplicada por vector de amplitudes) [Gottesman, 1999]. La primitiva *bit-flip* intercambia la probabilidad de los dos estados mientras que la *fase-flip* cambia el signo entre ambas. La primitiva de *Hadamard* establece un estado "a medio camino" entre otros dos. La primitiva *no-controlado* realiza una negación si el qu-bit de control es igual a 1 y no realiza cambio en el caso contrario. La primitiva de Hadamard es la que hace de puente entre los dos tipos de computación, cuántica y digital, realizando a partir de sus entradas una qu-función booleana con la que es posible construir las funciones XOR, AND, etc..

Frente al potencial enorme que representa la superposición cuántica para la resolución de problemas de computación masiva, como la factorización de grandes números o la búsqueda en bases de datos, [Hogg, 1996], [Hogg, 2000a], [Maurer, 2001], [Hogg, 2003], permanecen todavía algunos obstáculos que impiden, por el momento, la construcción de computadores cuánticos más importantes que los que existen, de pocos qu-bits. Las dificultades mayores están en la necesidad de prever la corrección de los errores debidos a la decoherencia y en el propio diseño de arquitecturas capaces de instrumentar la computación cuántica. En efecto, los fenómenos cuánticos son inestables, conduciendo necesariamente a la decoherencia, esto es, al abandono de un estado cuántico como resultado de la interacción inevitable entre el estado y el entorno. En 1995 se propone la teoría de corrección del error, basada en la idea de coherencia en fase, como un medio indirecto de extraer información y reducir el error en un sistema cuántico sin tener que realizar medición alguna sobre él, evitando así su destrucción [Shor, 1995], [Laflamme, 1996], [Aharonov, 1997]. Por otra parte, el qubit no puede ser construido a partir del transistor ya que éste es un elemento que sólo funciona en las computadoras actuales; más bien se deben utilizar partículas o sistemas de partículas que manifiesten el fenómeno de la interferencia cuántica. En este sentido, se han realizado algunos avances. La técnica de los puntos cuánticos consiste básicamente en un electrón atrapado dentro de un conjunto de átomos (jaula de átomos), el cual, mediante un rayo láser de una frecuencia específica, se traslada de su estado no excitado ("cero") a su estado excitado ("uno") y viceversa. Si la duración de la exposición al láser es igual a la mitad del tiempo requerido para cambiar el nivel energético del electrón, este adquiere un estado de superposición de sus dos valores posibles. [Steane, 2001]. La técnica de las moléculas líquidas utiliza grupos de moléculas en lugar de una partícula elemental. Al ser sometidos a un campo magnético, los núcleos de las moléculas giran en una determinada dirección que puede ser utilizada para describir su

estado (giro hacia arriba "uno", giro hacia abajo "cero"). Mediante señales de radiofrecuencia, el giro puede modificarse. Así, el computador cuántico vendría a estar representado por las moléculas, y los qubits por los núcleos [Steffen, 2003].

En el campo del hardware cuántico, se cuenta con algunas realizaciones. La construcción de una puerta lógica de 2 qu-bits basada en la resonancia magnética nuclear, que puede ser adaptada para lograr los requerimientos de un computador cuántico, fue llevada a cabo en 1997 por un equipo mixto de Los Álamos y del MIT. En marzo de 2000, también en Los Álamos, se anuncia el desarrollo de un computador cuántico de 7 qu-bit que utiliza la resonancia magnético-nuclear para manipular las partículas en los núcleos de los átomos de las moléculas de ácido trans-crotónico, aplicándoles pulsos, a fin de forzar su posicionamiento a la manera de la codificación por bits en los computadores digitales. En agosto de 2000 científicos del Centro IBM-Almaden de investigación proponen un computador cuántico de 5 qu-bits capaz de resolver un problema típico en la criptografía, como es encontrar la periodicidad de una determinada función.

Cabe pensar que el progreso en computación cuántica planteará un problema de seguridad en la información, ya que, los métodos actuales de encriptación son realmente simples comparados con la potencia de la computación cuántica para desencriptar [Daemen, 2000], [Daemen, 2001].

En resumen, la computación cuántica se encuentra en sus inicios, la tecnología debe avanzar hasta ser capaz de asegurar el funcionamiento correcto de computadores de varias decenas de qu-bits para poder resolver problemas reales. La investigación ha de proseguir por la vía de buscar métodos para vencer el gran obstáculo que representa la decoherencia previo a plantearse el diseño y realización de hardware eficaz así como la consecución de algoritmos

cuánticos potentes [Hogg, 1998a], [Hogg, 1998b], [Hogg, 1998c], [Jozsa, 1998], [Jozsa, 1999], [Wiseman, 2000], [Ho, 2000b], [Hogg, 2000c], [Janzing, 2001], [Van Dam, 2002].

2.4 Modelos computacionales en neurología

En este apartado se hace un resumen del estado del conocimiento en el campo de la neurología, desde los primeros postulados que datan del siglo XIX hasta los modelos computacionales últimamente desarrollados. En cada caso se destaca el enfoque utilizado por el modelo así como su alcance. Al final de esta exposición se hace una reflexión sobre los hitos y limitaciones de los modelos y operatorias propios de este campo.

En 1889, S. Ramón y Cajal establece un postulado según el cual la neurona, con sus árboles dendrítico y axonal, es la unidad básica funcional del sistema nervioso [Ramón y Cajal, 1894]. Afirma que las dendritas y el soma son las áreas receptivas para las entradas procedentes de los terminales de otras neuronas y que los impulsos de salida se transmiten unidireccionalmente, por el axón y mediante conexiones axo-dendríticas, al árbol dendrítico de otras neuronas. También recon oce la estructura fina de las neuronas con dendritas, espinas y botones axónicos, mediante observación al microscopio óptico de neuronas teñidas por el método de Golgi. Con respecto al procesamiento de la información, Ramón y Cajal sólo llega a formular la pregunta de qué sucede cuando varias entradas compiten entre sí, suponiendo erróneamente que el soma decide la prioridad de paso.

Hasta 1940 aproximadamente, la tónica general en neurología es ignorar la morfología dendrítica, llegando a representar la neurona por un punto y el axón por una línea. Así, según McCullogh y Pitts, la excitación por sinapsis va desde un terminal axónico hasta un soma [McCullogh, 1943]. Esta neurona estilizada o neurona puntual es la base de modelos matemáticos que tienen como propósito

explorar el comportamiento dinámico de grandes redes de neuronas interconectadas. Hacen hincapié en la característica todo o nada del disparo de la neurona. La unidad funcional se representa como elemento digital, binario.

En 1952, el modelo de Hodgkin y Huxley de tres canales describe matemáticamente las diversas corrientes de membrana caracterizadas por la técnica de pinzamiento [Hodgkin, 1952]. Tiene por objeto estudiar la contribución de todas las corrientes iónicas al comportamiento eléctrico de la neurona modelizada. La formulación original del modelo consiste en cuatro ecuaciones diferenciales acopladas, una para el potencial de membrana y tres para las variables de estado que dependen del tiempo y del voltaje: dos para las corrientes de Na^+ y una para la corriente de K^+. A pesar de su utilidad para la comprensión de fenómenos como el umbral de disparo, oscilaciones subumbral y período refractario a partir de razonamiento físico, este modelo biofísico es complicado para el análisis matemático y acarrea un coste computacional demasiado elevado. Se ha intentado simplificarlo dejando sólo dos ecuaciones diferenciales a fin de poder aplicarle el análisis mediante plano de fase. Además, dicho modelo no aclara las cuestiones relativas a la importancia de las sinapsis distribuidas que llegan al árbol dendrítico.

Alrededor de 1964, Rall aborda el problema decidiendo tratar por separado las dificultades morfológicas y eléctricas [Rall, 1964]. Despreciando en un primer tiempo las no linealidades de la membrana, se centra en hallar la expresión matemática que describe la variación de corriente y tensión según el tiempo y el espacio en un árbol dendrítico, morfológicamente complejo, pero eléctricamente pasivo. Para ello desarrolla la teoría de cable. En esta aproximación, la ecuación diferencial de segundo orden del potencial en función del tiempo y el espacio (ya propuesta por Hodgkin y Rushton) se resuelve con condiciones impuestas por la estructura del árbol. Esta teoría permite caracterizar las propiedades entrada-salida de un árbol pasivo; es decir, calcular la tensión en cualquier

punto del árbol que sigue a una entrada de corriente en cualquier otro punto. Se llega a poder estimar la amplitud y constante de tiempo del potencial en el punto que interesa así como la atenuación en el soma y la distancia entre éste y la localización de la sinapsis, a partir de la forma del potencial postsináptico. De ello se deduce que la atenuación de la señal es distribuida y se rige por una suma de exponenciales. Entonces, en principio al menos, cualquier entrada a una dendrita distal produce una carga significativa en el soma. Conclusión importante: la combinación de propiedades eléctricas por una parte y morfología específica, por otra parte, de la membrana y del citoplasma determinan el procesado de la señal en el árbol dendrítico. Se definen como parámetros electrotónicos característicos del árbol dendrítico R_m, C_m, y R_i, que representan la resistencia específica de membrana, su capacitancia y la resistencia específica del citoplasma, respectivamente.

Más tarde, y para superar la restricción que se deriva de asumir que la membrana es pasiva, Rall desarrolla un modelo compartimental. En los últimos años esta visión es la que ha cobrado mayor prestigio, tanto a nivel de célula como a nivel de red, motivando avances como los conseguidos por los pioneros Rall y Shepherd que modelizaron el bulbo olfatorio [Rall y Sheperd, 1968]. Cabe destacar, entre otras, la aplicación de técnicas compartimentales para modelar la difusión de los iones en distintas partes del árbol y para emular el comportamiento oscilatorio observado en una porción del hipocampo. Posteriormente, se ha seguido avanzando hasta llegar a trabajos más recientes que tratan sobre todo de modelizaciones de conjuntos de células corticales [Suga, 1990], [Ekhorn, 1990], [Harth, 1990].

Con la salvedad del tratamiento formal no binario, el modelo de FitzHugh y Nagumo pertenece, por su carácter abstracto, a la perspectiva estructural [FitzHugh, 1961] En él, las propiedades eléctricas de la membrana de la neurona se representan por dos variables (dos ecuaciones diferenciales

acopladas) y cada una de ellas satisface una ecuación diferencial con no linealidades polinómicas. La interpretación cualitativa, facilitada por el plano de fase, explica aspectos matemáticos de las no linealidades de la membrana. Los parámetros que definen este modelo no tienen interpretación física directa.

Dentro de la perspectiva estructural pero con dos hipótesis adicionales referidas al número de subsistemas (que deberá ser grande) y a la imposibilidad de distinguir los subsistemas, se llega a un enfoque de grandes poblaciones. Las no linealidades, que aparecen como resultado de la interacción entre partes de un sistema, están en el origen de propiedades emergentes que no son inherentes a ninguna de las partes componentes. Esta teoría constituye la base de un cuerpo de conocimiento denominado Sistemas Complejos [Solé, 2003a]. La complejidad no es necesariamente sinónimo de complicación; lo que sí exige es abandonar la intuición lineal de los fenómenos y reemplazarla por una visión del mundo basada en la no linealidad [Bascompte, 1998], que requiere tratamiento estadístico [Solé, 2001a], [Solé et al, 2003b]. En biología son numerosos los sistemas complejos: los millares de genes que unos a otros se regulan entre sí en el seno de las células; las redes de células y moléculas que median en la respuesta inmune [Solé et al, 2003c]; las mallas de los ecosistemas repletas de especies en coevolución [Alonso, 2000]; [Solé, 2001b], las colonias de insectos sociales [Delgado, 1999] y, por fin, dentro del tema que nos ocupa, los miles de millones de neuronas de las redes nerviosas subyacentes a la conducta y al aprendizaje. Existen trabajos notables en todas estas ramas y cabe destacar el trabajo seminal de R.V. Solé [Solé, 94] que desarrolla un modelo basado en redes fluidas en el que se profundiza en las analogías entre un hormiguero y un cerebro. En este modelo, cada hormiga pasa a ser una neurona y las interacciones entre neuronas son las habituales en una red neuronal clásica. El modelo cuenta con una propiedad adicional, peculiar de los hormigueros reales, a saber, la activación espontánea caótica de los individuos. La simulación muestra la transferencia de información, entendida ésta como la

capacidad de un sistema complejo para captar y procesar información, para distintas densidades: A baja densidad, las fluctuaciones son muy irregulares; los individuos, aunque se activen, no pueden propagar sus cambios. A grandes densidades, las fluctuaciones del sistema se tornan periódicas. Entre ambas existe una densidad crítica para la cual la información transmitida, en el sentido de Shannon, se hace máxima. Estos autores postulan que una de las propiedades de los sistemas complejos es su capacidad para interaccionar con su ambiente y procesar la información recibida. No es que exista isomorfismo entre el cerebro y el hormiguero, más bien se trata de entender que el mismo modelo puede tener riqueza suficiente para dar cuenta de propiedades que ambos comparten.

Queda patente que, en los últimos años y gracias a los métodos analíticos y computacionales cada vez más eficientes, la neurología ha consolidado una enorme cantidad de conocimiento basada en datos experimentales, contrastados por modelos hechos a su medida. Ello ha proporcionando un conocimiento específico de la neurona y de ciertos subconjuntos del córtex acorde con la potencia de las operatorias y las arquitecturas propias, diseñadas desde el enfoque de la tecnología bioquímica. Hay que destacar las aportaciones hechas en investigación básica, celular y molecular, principalmente en la fisiología de la neurona o de los sistemas neuronales [Gao, 2003], [Hering, 2003], [Jackson, 2004]; en investigación de funcionalidades emergentes que están en el origen del comportamiento y del conocimiento [Markowska, 2002], [Cain, 2002], [Hyman, 2003], [Hamana, 2003], [Carlson, 2004] y en investigación en torno a la plasticidad y capacidad de reparación del sistema nervioso [Ueki, 2003], [Akazwa, 2004], [Strettoi, 2004]. Sin embargo, todo esto no ha sido suficiente para alcanzar un grado suficiente de madurez en lo que respecta a la comprensión de ciertas funcionalidades de alto nivel que resuelve el sistema nervioso, como percepción, aprendizaje, razonamiento, etc... No se ha llegado, por ejemplo, a entender el procesamiento de señales de forma sistemática, a definir códigos neuronales, a explicar fenómenos como el de la consciencia y

esto equivale a decir que el aspecto más relevante y genuino del sistema nervioso central como la representación, procesado, uso y actualización del conocimiento está por modelar.

2.5 Problemas abiertos

El análisis del estado del conocimiento actual se ha llevado a cabo en torno a las cuatro cuestiones relevantes para este trabajo de investigación, que son, las primitivas computacionales habituales suma y multiplicación y su intervención en la evaluación de funciones habituales, enfocadas ambas desde la aritmética del computador; el análisis de primitivas de alto nivel para procesadores específicos en ramas de la informática como el tratamiento digital de imágenes y la computación gráfica; la computación cuántica y, para terminar, los modelos y operatorias empleados en neurociencias, que tratan de emular las funcionalidades sofisticadas de los sistemas vivos. De este estudio se desprenden algunas observaciones:

- Cada tecnología, electrónica o bioquímica, acaba consolidándose en una operatoria y, en los campos tratados, aparece que el modelo de capas está vigente.
- Los planteamientos en cuanto a las primitivas a utilizar en los procesos de evaluación siguen dos directrices: primitivas muy sencillas (del tipo de la suma y poco más) para arquitecturas de propósito general; o primitivas sofisticadas para arquitecturas especializadas (operaciones gráficas, filtros de imágenes, etc...), independientemente de cuál sea la complejidad de la evaluación a realizar.

- Las distintas arquitecturas encontradas corresponden a particularizaciones de la tecnología, de la estructura de las capas o de la funcionalidad que proporciona la capa.
- De las diversas operatorias empleadas por los procesamientos especializados expuestos, hay que destacar la recursividad y la parametrización como recursos potentes de modelado; la iteración de patrones simples, geométricos o gramaticales, con introducción de una pequeña colección de parámetros controladores produce resultados espectaculares. Aparece el elemento estructurante como "unidad" en los procesos de evaluación de la morfología matemática. La computación cuántica y la computación digital tienen como nexo de unión la primitiva de Hadamard. Ésta se expresa en forma de suma ponderada de dos estados produciendo un estado intermedio capaz de hacer de puente entre ambas.
- La neurociencia utiliza la informática como herramienta de cálculo, recurriendo a sus técnicas de simulación y adaptándolas a su objeto de estudio. La informática se inspira a veces en lo vivo para proponer nuevos paradigmas de computación.
- Estas observaciones apuntan en la dirección de una gran dependencia entre la tecnología por una parte, la arquitectura y la operatoria por otra, a la vez que relacionan la sofisticación de las primitivas con el carácter, general o específico, de las resoluciones.

3 Formulación del problema y propuesta de resolución

Después de revisar, en el apartado anterior, el estado actual del conocimiento en cuanto a los aspectos relevantes del tema de estudio que motiva esta investigación, se pasa a definir el problema en términos formales con el propósito de explicitar el marco conceptual de diseño de soluciones así como de establecer el soporte expresivo necesario.

Como ya se ha señalado en los objetivos, se va a abordar el problema de evaluar una función cualquiera al nivel más bajo posible.
La evaluación de una función dada puede realizarse a diferentes niveles de derivación dependiendo de cuáles sean las primitivas del procesador e incluso, para un procesador dado, la misma función puede ser evaluada a diferentes niveles. Por ejemplo, la evaluación de la bifurcación se realiza para cualquier unidad central de procesamiento (o equivalentemente, para cualquier conjunto de primitivas) y, para una unidad central de procesamiento determinada, es común que la bifurcación se evalúe a cualquier nivel de procesamiento, desde los de ensamblador hasta los lenguajes de alto y muy alto nivel y los entornos de desarrollo.

Si una arquitectura A se define por la terna $A = \langle \Pi, \Lambda, \Delta \rangle$, siendo

$\Delta = \{\delta_1, \delta_2, \cdots, \delta_d\}$ el dominio, esto es, el conjunto de funciones que evalúa,

$\Pi = \{\pi_1, \pi_2, \cdots, \pi_p\}$ el conjunto de primitivas que incorpora su unidad central de procesamiento

$\Lambda = \{\lambda_1, \lambda_2, \cdots, \lambda_l\}$ la lógica de derivación

entonces, cualquier función del dominio de la arquitectura podrá obtenerse a partir de las primitivas mediante la correspondiente derivación:

$$\exists \lambda_k / \delta_i = eval(\Pi, \lambda_k) \quad \forall i = 1 \cdots d$$

Por razones de escalabilidad de los sistemas, en un procesador Π dado, la evaluación de las funciones se organiza en niveles derivados D_j a partir del elemental de las primitivas.

$$D = \{D_1, D_2, \cdots, D_n\}$$

$$D_j \subset \Delta / j = 1 \cdots n$$

$$D_j = \{eval(\Pi, \lambda_k)\}$$

El hecho de que los niveles derivados puedan no ser disjuntos se debe a que una misma función es evaluable por más de una regla de derivación a partir de Π; por tanto, la evaluación de una función dada no tiene por qué ser única

$$D_s \cap D_t \neq \Phi \Leftrightarrow$$

$$\exists \delta_i \in (D_s \cap D_t), \exists \lambda_j, \lambda_k / \delta_i = eval(\Pi, \lambda_j) = eval(\Pi, \lambda_k)$$

$$eval(\Pi, \lambda_j) \in D_s, eval(\Pi, \lambda_k) \in D_t$$

La ordenación de los niveles derivados induce una clasificación en los elementos del conjunto Λ de manera que una regla de derivación podrá expresarse como una función de otras reglas de derivación que intervienen en la evaluación de las funciones de los dominios inferiores

$$\forall \lambda_k / eval(\Pi, \lambda_k) = \delta_j \in D_j$$

$$\lambda_k = \varphi(\lambda_1, \cdots \lambda_q) / \forall r, 1 \leq r \leq q \quad eval(\Pi, \lambda_r) = \delta_i \in D_i \quad 1 \leq i < j$$

Una arquitectura tiene potencia para resolver un problema si éste puede expresarse como una función del dominio de la arquitectura

$$A\langle \Pi, \Lambda, \Delta \rangle resuelve\ el\ problema\ \wp \Leftrightarrow \exists D_j \subset \Delta / \wp \in D_j$$

El subíndice j de la expresión anterior indica el grado de derivación que se requiere en la arquitectura A para resolver $\wp$. Puede observarse que cuanto menor sea j, menor será la complejidad de la solución al problema en términos de costes computacionales. En ese sentido, podrá hablarse de que j expresa el grado de especificidad de la arquitectura para la resolución del problema.
La formulación del problema planteado en esta investigación, expresada en términos rigurosos es

> **encontrar una arquitectura A' de especificidad máxima para resolver un problema determinado, $\wp$**

Formalmente,

$$Si\ A' = \langle \Pi', \Lambda', \Delta' \rangle / \exists \lambda_{k'} \in \Lambda', \exists \delta_{h'} \in D_{i'}, D_{i'} \subset \Delta_{i'} / \wp = eval(\Pi', \lambda_{k'}) = \delta_{h'}$$
$$y\ A'' = \langle \Pi'', \Lambda'', \Delta'' \rangle / \exists \lambda_{k''} \in \Lambda'', \exists \delta_{h''} \in D_{i''}, D_{i''} \subset \Delta'' / \wp = eval(\Pi'', \lambda_{k''}) = \delta_{h''}$$
$$siendo\ A' \neq A''\ entonces$$
$$A'\ es\ de\ especificidad\ máxima \Leftrightarrow j' = \min(j', j'')$$

La expresión de A' deja abierta la posibilidad de resolver buscando la arquitectura más adecuada tanto por la vía de las primitivas de la unidad central de procesamiento como por la vía de la lógica de derivación, de manera que el

campo de búsqueda de soluciones de interés sigue siendo extraordinariamente amplio. Por esa razón, se ha determinado adoptar una técnica de aproximación sucesiva a la eventual solución estricta, abordando versiones cada vez más refinadas del problema.

En esta investigación se propone abordar la búsqueda de conjuntos de primitivas que constituyan Π para resolver un problema determinado, con la característica de que las primitivas estén inspiradas en la naturaleza formal de las funciones a evaluar. Para ello, se propone inicialmente una metodología de evaluación de funciones basada en la convolución que proporciona una solución de partida, de propósito general y con alto grado de derivación ya que las primitivas utilizadas son las habituales, esto es, la suma y la multiplicación. Una segunda etapa de aproximación a la resolución del problema de la evaluación, en la línea de más especificidad, la constituye la inducción, a partir de la convolución, de una forma recursiva de primitiva, capaz de emular por parametrización, la naturaleza formal de subconjuntos de estas funciones. La investigación se concreta en el diseño del procesador Π. Se trata de una unidad aritmética cuyas primitivas permiten calcular, funciones que otras arquitecturas obtienen con niveles más altos de derivación. La operatoria aprovecha la forma recursiva de la primitiva para evaluar iterativamente la función, logrando un resultado final a partir de la combinación de parciales. Esta técnica ofrece la posibilidad de fragmentar los parciales en bloques, favoreciendo la segmentación y un mayor nivel de paralelismo en el proceso de combinación. Los parciales pueden calcularse o figurar en tablas como datos precalculados que hay que extraer, potenciando la realización de compromisos entre el tiempo de cálculo y el espacio ocupado.

Capítulo 2

MÉTODO DE EVALUACIÓN RECURSIVA BASADO EN CONVOLUCIÓN (CBRM)

1 Introducción

La idea fundamental que anima este trabajo de investigación es que, encontrar formas de evaluar funciones con más especificidad que la que se consigue con el uso de las primitivas habituales, suma y multiplicación, ayudará a reducir el número de niveles de derivación necesarios y, por tanto, a aumentar el rendimiento. La propuesta de resolución enunciada en el capítulo primero consiste en encontrar primitivas que cumplan con este cometido para el mayor número de casos posibles. Con el fin de orientar esta búsqueda, se analizan algunos resultados del análisis matemático referidos a la teoría de la medida, que centran y formalizan el procedimiento habitual de la medición. También se han considerado los conceptos de la teoría de la representación de grupos, que permiten asociar estructuras algebraicas distintas para regularizar elementos u operaciones de una de las estructuras sobre sus homólogos en la otra. Esta

regularización facilita la resolución de problemas tratándolos en la estructura mejor entendida de las dos. De esta recopilación se desprende que la convolución ocupa un lugar de interés dentro del marco de construcción de la medida producto y, a lo largo de este capítulo, se demuestra que la convolución de dos funciones representa la evaluación de una de ellas tomando la otra como unidad. Llamaré evaluación estructural a este tipo de evaluación, porque se realiza por comparación con una referencia que tiene la misma naturaleza que el objeto evaluado. La convolución tiene un coste computacional notable. Tener que realizar evaluaciones con gran especificidad y para un amplio número de funciones aconseja la búsqueda de un tipo de operación paramétrica, en la que pequeños cambios en los parámetros puedan cubrir la gran variedad de casos. La condición relativa a la disminución de los niveles de derivación necesarios para realizar la evaluación puede abordarse por la vía de reducir la diversidad de estos mecanismos y de aumentar su potencia. Para ello, se ha optado por hacer recursiva la operación paramétrica por lo que, en cualquier caso, los mecanismos de derivación serán dos: inicializar e iterar. La evaluación realizada por la operación recursiva recibe, en este trabajo, el calificativo de no estructural porque se efectúa sin necesidad de elemento de referencia. Por último, entre la convolución y la operación recursiva paramétrica se establece la correspondencia pertinente, que hace que éstas sean operaciones homólogas. Ello permite evaluar funciones por convolución, pero con una operatoria que reduce notablemente el coste computacional.

Este capítulo se consagra enteramente a la exposición del método de evaluación recursiva basada en convolución, CBRM, como solución al problema planteado en el capítulo primero. Dado el carácter central que tiene la convolución en este trabajo de investigación, he creído oportuno dedicar un párrafo a repasar la historia y evolución de esta operación.

2 Acerca de la convolución

La convolución es conocida como un operador matemático entre funciones, muy presente tanto en el campo de las matemáticas como en las ingenierías. Por citar algunos ejemplos: en estadística, la densidad de probabilidad de la suma de dos variables aleatorias independientes es la convolución de sus densidades individuales; en ingeniería eléctrica, la salida de un sistema lineal es la convolución de la entrada con la respuesta del sistema a un impulso [Dorf, 1989], [Katsukiko, 1993]; la morfología matemática adapta el concepto de convolución implementándola con operaciones lógicas en vez de aritméticas [González, 1996].

También observamos convoluciones en fenómenos corrientes; muchos tipos de "manchas" se describen como convoluciones: una fotografía desenfocada es la convolución de la imagen correcta con el círculo borroso formado por el diafragma del iris; una sombra es la convolución entre la forma de la fuente de luz que crea la sombra y el objeto cuya sombra se está proyectando. El eco es también una convolución, entre el sonido original y una función que represente los objetos variados que lo reflejan.

En la historia de las matemáticas, la primera mención explícita al producto de convolución aparece en una memoria de Tchebyshev [Tchebyshev,1890] a propósito de cuestiones relativas al cálculo de probabilidades y un repaso de los acontecimientos previos que contribuyeron al desarrollo y maduración de la idea pone de manifiesto que, en poco más de un siglo, la convolución pasa de ser una herramienta de cálculo para la resolución de ecuaciones en derivadas parciales, a intervenir en la teoría de la medida para contribuir finalmente al establecimiento de postulados importantes en el dominio del álgebra, concretamente en la teoría de representación de grupos.

Antes de 1820, las ecuaciones en derivadas parciales venían utilizándose por Poisson, entre otros, para escribir las integrales de la ecuación del calor o por Fourier, para realizar la suma parcial de una serie sin advertir que, formalmente, se efectuaba la regularización de una función sobre otra. El ejemplo más célebre de regularización mediante núcleos positivos es propuesto en 1902 por Fejér en su tesis doctoral [Tandori, 1983] y, a partir de este momento, se convierte en el procedimiento estándar que interviene en la mayor parte de los métodos de sumación de las series de funciones. Sin embargo, estos trabajos, a causa sin duda de la disimetría de los papeles desempeñados por el núcleo y la función reguladora, no ponían de manifiesto con claridad las propiedades algebraicas del producto de convolución.

A lo largo del siglo XIX, tiene lugar la formalización de la teoría de la medida en el campo del análisis. La primera definición de medida *m(A)*, para un conjunto arbitrario acotado *A*, $A \subset R^n$ es establecida por Cantor en 1883. En 1887, Peano fija las condiciones que hacen que un conjunto sea medible y además demuestra que la medida es aditiva y que existe una relación entre medida e integración, por medio de la integral de Riemann cuyo valor representa precisamente la medida del espacio delimitado por la función integrada, los bornes de integración y el eje de las abscisas [Cohn, 1980]. Al hacer extensiva la medida al producto de una familia de espacios de medida, la integración correspondiente puede resolverse mediante el cálculo de integrales unidimensionales, en un proceso iterativo cuya justificación viene proporcionada por el teorema de Fubini [Hawkins, 1975]. En este contexto aparece el producto de convolución, como una construcción de la medida, de naturaleza probabilística, basada en el experimento compuesto.

En 1913, Volterra evidencia por primera vez las propiedades algebraicas del producto de convolución que considera como una generalización, por paso de finito a infinito, del producto de dos matrices [Volterra, 1913]. El giro definitivo

tiene lugar unos años más tarde, cuando Daniell relaciona la convolución de dos medidas cualesquiera sobre R con la transformada de Fourier de una medida del mismo tipo [Daniell, 1929]; se señala entonces explícitamente que la transformación de Fourier hace pasar de la convolución al producto ordinario. Pero, desde el punto de vista de la teoría de grupos, la importancia de la convolución no es reconocida plenamente hasta 1927, por H. Weyl [Weyl, 1927]. Éste postula que, en un grupo compacto, la convolución de funciones desempeña el papel de la multiplicación en el álgebra de un grupo finito, permitiéndole por tanto definir la representación regular y, al mismo tiempo, encontrar por regularización el equivalente del elemento unidad de un grupo finito, marcando una tendencia característica del siglo XX que es la algebrización del análisis.

En la actualidad, la convolución es objeto de una intensa investigación. Parte del trabajo sigue la línea teórica, abordando el estudio de la operatoria bajo condiciones nuevas [Seeger, 1996], [Bak, 1997], [Karasik, 1998], [Fernández, 2001], [Cardon, 2002], [Eijndhoven, 2003], [Dettweiler, 2003], [Grabiner, 2004], buscando métodos nuevos de cálculo con propósitos diversos [Stalling, 1995], [Médicis, 1995], [Berkner, 1999], [Sherstyuk, 1999], [Baeumer, 2003] [Chechile, 2003] o centrados específicamente en la resolución de ecuaciones [Peszynska, 1996], [Cápec, 1997], [Baesh, 1997], [Buegholz, 1999], [Oberlin, 2002], en la estimación de medidas [Li, 2003], [Ma, 2003] y en el análisis armónico [Capelle, 1996], [Chu, 1999]. La línea aplicada establece correspondencias en contextos como la teoría de control, el equilibrio en sistemas económicos, la probabilidad y la teoría de procesos estocásticos, entre otros [Parr, 2002], [Corazza, 2002], [Ye, 2002].

3 Fundamentación teórica del CBRM

"...En un grupo compacto, la convolución de funciones desempeña el papel de la multiplicación en el álgebra de un grupo finito, permitiendo por tanto definir la representación regular y, al mismo tiempo, encontrar por regularización el equivalente del elemento unidad del grupo finito..." [Weyl, 1927].
Por lo que se sabe de la definición básica habitual de multiplicación como suma de sumandos iguales, el concepto de medida le es inherente: en efecto, cada sumando representa la "referencia" que se toma para medir y el número de estas referencias representa la medida. Al efectuar la multiplicación, el resultado representa también la misma medida pero en el caso particular de una referencia igual al elemento unitario. Por tanto, el postulado de Weyl apunta hacia la posibilidad de entender también la convolución en términos de una medida.
Por otra parte, Weyl utiliza la regularización para establecer equivalencias entre estructuras y deducir de éstas correspondencias útiles entre elementos y operaciones de cada una de ellas.

En este apartado se desarrollan las nociones de la teoría de la medida [Cohn, 1980], y de la teoría de la representación de grupos que fundamentan el planteamiento del CBRM. Por una parte, el CBRM propone la convolución como forma de evaluar funciones, considerando que la convolución entre dos funciones representa la medida de una de ellas tomando la otra como unidad, relación equivalente a la que existe entre los dos elementos de un producto habitual. Por otra parte, el CBRM establece una regularización de la convolución sobre una operación recursiva paramétrica, que calcula por iteraciones sucesivas y que permite satisfacer los objetivos señalados en cuanto al coste computacional.

3.1 Teoría de la medida

El concepto de medida tiene una larga historia de más de 5000 años, que surge del manejo de longitudes, áreas y volúmenes y de la necesidad de su cálculo [Van Dalen, 1972], [Boyer, 1986]. Intuitivamente, calcular una medida consiste en asociar un valor a una característica de un objeto. El valor no es una cualidad intrínseca del objeto evaluado, ya que, la acción de medir requiere habitualmente la comparación con un elemento de referencia, de naturaleza idéntica al objeto a evaluar, cuya elección es arbitraria. Por tanto, la existencia del valor va ligada a la existencia de una estructura formal previa que permita las comparaciones y la cuantificación de éstas.

Concepto de medida

Desde el punto de vista formal, el problema de la medida se plantea en el marco de un conjunto E del que se quiere medir alguno de sus subconjuntos. La estructura mínima que permite establecer una medida es la σ–álgebra [Nielsen, 1997].

Una σ–álgebra en E es una colección A de subconjuntos de E con las siguientes propiedades

- $E \in A$
- Si $X \in A$ entonces $X^c \in A$
- Si $X_1, X_2, ..., X_n, ... \in A$ entonces $\bigcup_{n=1}^{\infty} X_n \in A$

Si la tercera condición sólo se verifica para la uniones finitas, entonces A es un álgebra.

El par (E, A) donde E es un conjunto y A es una σ–álgebra sobre E, es un espacio medible y conjuntos medibles son los elementos de A. Existe gran

cantidad de ejemplos de espacios medibles, de los cuales destacamos $(E, \wp(E))$, siendo $\wp(E)$ la mayor σ–álgebra de E.

La medida entendida como una función de conjuntos. Ejemplos.

En un espacio medible, (E, A), se define una medida μ como:

$\mu: A \longrightarrow [0, \infty[$

que satisface

- $\mu(\varnothing) = 0$
- μ es numerablemente aditiva, esto es, la medida de una unión numerable de elementos disjuntos de A es igual a la suma de sus medidas individuales. Se requiere que la unión sea también elemento de A.

La terna (E, A, μ) se denomina espacio de medida.

Existen muchos ejemplos de medidas de las cuales pueden destacarse algunas:

- La medida de contar:

se define en $(E, \wp(E), \mu)$, por $\mu(X)$ = número de elementos que tiene X, siendo X un elemento de $\wp(E)$.

- La probabilidad:

se define en $(N, \wp(N), \mu)$, por $\mu(X) = \Sigma p_n$ siendo X un elemento de $\wp(N)$ y p_n una sucesión de números reales no negativos. Si $\Sigma p_n = 1$, entonces la medida es una probabilidad y si $p_n = 1$ para cada n, entonces se trata de la medida de contar de los naturales.

- La medida de Lebesgue:

representa una extensión de las nociones de longitud y área. Dado un conjunto abierto conteniendo intervalos disjuntos, $X = \sum_k (a_k, b_k)$, la medida de Lebesgue de X es $\mu_L(X) = \sum_k (b_k - a_k)$.

Si X' es un conjunto cerrado $X' = [a,b] - \sum_k (a_k, b_k)$, la medida de Lebesgue es $\mu_L(X') = (b-a) - \sum_k (b_k - a_k)$.

- La medida de Haar:

es una manera de asignar un volumen invariante a los subconjuntos de grupos topológicos localmente compactos. Si G es un grupo topológico localmente compacto podemos considerar la σ-álgebra X generada por todos los subconjuntos compactos de G. Si a es un elemento de G y S es un conjunto en X, entonces el conjunto $aS = \{as / s \in S\}$ (donde la multiplicación es la operación en G) está también en X. Una medida μ en X se llama invariante por traslación izquierda si $\mu(aS) = \mu(S)$ para todo a y S.

Se verifica que hay, salvo una constante multiplicativa, sólo una medida invariante por traslación izquierda en X, que sea finita en todos los conjuntos compactos. Ésta es la medida de Haar en G. (hay también una medida invariante por traslación derecha, esencialmente única en X, pero las dos medidas no necesitan coincidir). La medida de Haar en el grupo topológico (R,+) que toma el valor 1 en el intervalo [0, 1] es igual a la medida de Borel. Esto puede generalizarse para (R^n, +).

La medida expresada en términos de integral. Ejemplos.

Si A = C_K , siendo C_K el conjunto de las funciones φ: R ⟶ R que cumplen dos condiciones fundamentales, que son, anularse fuera de un intervalo que

depende de la función particular y ser continuas en R, entonces la función φ es no negativa si $\varphi(t) \geq 0$, $\forall t \in R$.

En este caso, la aplicación lineal $\mu : C_K \longrightarrow R$, del espacio de las funciones continuas con soporte compacto en el cuerpo R de los números reales, tal que $\mu(t) \geq 0$ para φ no negativa, es una medida.

Puesto que toda función continua de soporte compacto es integrable, una forma de concretar el funcional asociado con la medida μ es por medio de una integral, [Wheeden, 1977], en esta integral, la función φ representa la "densidad" de la medida.

$$\langle \mu, \varphi \rangle = \int_R \varphi d\mu$$

Se citan algunos ejemplos de medidas definidas en términos de integrales:

- La medida de Lebesgue [Bartle, 1995]

se construye a partir del funcional aplicando la integral de Riemann

$$\langle \mu, \varphi \rangle = \int_R \varphi d\mu = \int_R \varphi dx$$

- La medida delta de Dirac

Sea *r* un número real. La fórmula

$$\langle \delta, \varphi \rangle = \varphi(r)$$

define una medida $T=\delta(r)$, que se denomina delta de Dirac y que sirve para establecer un modelo matemático para un impulso.

- La medida de Haar

Usando el enfoque general de la integración de Lebesgue, se puede definir una integral para todas las funciones medibles f: $G \rightarrow R$ (o C), llamada la integral de Haar. Esta definición es el primer paso del análisis armónico.

Generalización

Si f es una función localmente medible en el sentido de Lebesgue sobre un compacto de R, se le puede asociar la medida μ_f definida por

$$\langle \mu_f, \varphi \rangle = \int_R f(x)\varphi(x)dx \tag{2.1}$$

Esta relación establece una correspondencia biunívoca entre las clases de funciones medibles sobre todo compacto y sus medidas asociadas, circunstancia que se aprovecha para efectuar una identificación formal de ambos objetos. Cabe subrayar que, en este contexto, el concepto de medida no es una simple generalización del concepto de función, sino una generalización de las clases de funciones medibles sobre compactos.

Ampliación del concepto de medida: la medida producto

Si se quiere construir una σ–álgebra y una medida μ en el producto $E = \prod E_i$ de una familia de espacios de medida (E_i, A_i, μ_i) para i ∈N, es necesario considerar la integración del espacio de medida producto en relación con la integración en los espacios factores. El teorema de la medida producto resuelve la cuestión según dos planteamientos, uno de carácter geométrico y otro de carácter probabilístico [Parthasaraty, 1980]. A continuación se presentan los dos planteamientos para i=2.

La versión geométrica del teorema enuncia que dados dos espacios de medida (E_1, A_1, μ_1) y (E_1, A_2, μ_2), existe una única medida μ en A_1 x A_2 tal que para cada $A \in A_1$ y $B \in A_2$

$$\mu(A \times B) = \mu_1(A) \times \mu_2(B) \tag{2.2}$$

La aplicación más familiar e inmediata de esta idea es el cálculo del área de un rectángulo. Se considera el plano real RxR como producto de dos rectas reales R y el cálculo de áreas de figuras planas se obtiene a partir del cálculo de longitudes de segmentos m_2. Así:

$m_2[a, b]x[c, d]) = m_1([a, b]) \times m_1([c, d])$

de modo que la medida de Lebesgue del plano, m_2, es en cierto sentido el producto de dos copias de m_1, siendo esta última la medida de Lebesgue de la recta.

La segunda construcción es de naturaleza probabilística. En este caso, el teorema enuncia que partiendo de un espacio de medida (E_1, A_1, μ_1), un espacio medible (E_2, A_2) y una medida de transición Λ definida de $E_1 \times A_2$ en $[0, \infty[$, se construye una medida μ en $A_1 \times A_2$ tal que para cada $A \in A_1$ y $B \in A_2$.

$$\mu = \int_A \Lambda_B d\mu_1$$

Y además, para cada $C \in A_1 \times A_2$

$$\mu(C) = \int \Lambda(x, C_x) d\mu_1$$

La formalización sería dual para (E_2, A_2, μ_2) como espacio de medida, (E_1, A_1) espacio medible y Λ medida de transición definida de $E_2 \times A_1$ en $[0, \infty[$.

La idea es la del experimento compuesto. Suponiendo un experimento en el que se realizan dos observaciones, la primera x_1 está en E_1 y la segunda x_2 en E_2. La probabilidad de que la primera caiga en un conjunto A de A_1 es $\mu_1(A)$ y una vez hecha la observación x_1, la probabilidad de que la segunda caiga en un conjunto B de A_2 es $\mu_2(x_1, B)$. Entonces, la probabilidad de que la observación (x_1, x_2) caiga en $A \times B$ es:

$$\mu(A \times B) = \int_A \mu_2(x_1, B) d\mu_1 \qquad (2.3)$$

Convolución

Si G es un grupo dotado de una medida m (por ejemplo, un grupo topológico localmente compacto Hausdorff con la medida de Haar) y si f y g son funciones real- o complejo-valuadas y m- integrables de G, entonces se puede definir su convolución como

$$f * g(x) = \int_G f(y)g(xy^{-1})dm(y) \tag{2.4}$$

Comparando (2.1), (2.3) y (2.4), la convolución así definida aparece como una integral de medida en la cual una de las funciones actúa como densidad de medida mientras que la otra es la función a medir. La funciones f y g son intercambiables.

Para las funciones discretas hay que formular la convolución como sigue:

$$f * g(m) = \sum_n f(n)g(m-n) \tag{2.5}$$

En (2.5) los papeles desempeñados por las funciones f y g siguen siendo idénticos e intercambiables pero la interpretación de elemento unitario sustituye a la de densidad de medida.

3.2 Teoría de representación de grupos

La teoría de la representación de grupos tiene por objeto la construcción de una aplicación entre un grupo y alguna otra estructura que permita establecer correspondencias útiles entre elementos y operaciones, con el fin de trasladar problemas hacia la estructura mejor entendida para resolverlos con más

facilidad [Fulton, 1991]. Esta teoría no es uniforme ya que depende del tipo de grupo elegido (finito, compactos o localmente compactos, de Lie, ...) [Brocker, 1985] y de la estructura "blanco" (grupos de permutaciones, grupos de matrices sobre algunos cuerpos o, más generalmente, grupos de transformaciones lineales invertibles de un espacio vectorial,...). Hay que destacar que muchos de los resultados de la teoría de representación de grupos finitos son probados para grupos topológicos compactos o localmente compactos, haciendo un promedio sobre el grupo [Simon, 1996]. Estas pruebas pueden transportarse a los grupos infinitos si el promedio es sustituido por una integral conveniente. Un ejemplo conocido lo constituyen los grupos localmente compactos, usando la medida de Haar. La teoría que resulta es una parte central del análisis armónico que generaliza las nociones de series de Fourier y transformadas de Fourier a una transformación de funciones definidas sobre grupos localmente compactos. Para grupos compactos, se pueden conseguir armónicos extrayendo una representación irreducible de cada clase de equivalencia de representaciones. Esta elección de armónicos goza de algunas de las propiedades útiles de la transformada de Fourier clásica, como por ejemplo la de cambiar convoluciones por productos escalares.

Representación lineal: definición y ejemplos.

Una representación de un grupo finito G es un homomorfismo de grupo, ρ, de G en el grupo lineal $GL(V)$ donde V es un espacio vectorial de dimensión n definido por:

$$\forall(g, v) \in G \times V,\ g.v = \rho(g)(v)$$

Algunos ejemplos de representaciones lineales habituales son:

- Representación trivial

$\forall s \in G, \rho(s) = Id_V$

- Representación regular

Dado un espacio vectorial de dimensión $|G|$ se considera una base $B = \{e_h\}$ con $h \in G$. Para $s \in G$ se define $\rho(s) \in GL(V)$ por $\rho(s)(e_h) = e_{sh}$ que corresponde a una permutación de las coordenadas

- Representación suma

Dadas dos representaciones ρ_1 y ρ_2 sobre V y W respectivamente, se define la representación $\rho_1 \oplus \rho_2$ sobre *GL(V, W)* por

$$\forall (v, w) \in GL(V, W), \rho_1 \oplus \rho_2(g)(v + w) = \rho_1(v) + \rho_2(w)$$

- Representación producto

Dadas dos representaciones ρ_1 y ρ_2 sobre V y W respectivamente, se define la representación $\rho_{V \otimes W}$ sobre *GL(V, W)* por

$$\forall f \in GL(V, W), \rho_{V \otimes W}(g)(f) = \rho^2(g) \circ f \circ \rho^1(g^{-1})$$

Considerando la representación regular, aparece que el número total de las permutaciones de los vectores de la base *B* es igual al número de elementos del grupo *G*. Así, cada elemento del grupo *G* se regulariza sobre un elemento de *GL(V)*, esto es, sobre una permutación.

3.3 Planteamiento del CBRM

Después de poner de manifiesto que la convolución es una operación entre dos funciones que representa la evaluación de una de ellas tomando la otra como unidad, se conviene en dar a esta evaluación el calificativo de estructural debido a que se realiza por comparación con una referencia de la misma naturaleza que la función evaluada. La posibilidad de regularizar la convolución sobre otra operación, como sugiere la teoría de representación, permite sustituir la evaluación estructural por esta segunda operación, que recibirá el nombre de evaluación no estructural, porque en este caso no se necesita comparación con ninguna referencia.

La razón de este planteamiento es crear el marco formal adecuado para evaluar por convolución pero con menor coste computacional del que ésta ocasiona. La evaluación estructural de funciones del tipo $h(k) = h(k)*\delta(k)$ se caracteriza generalmente por un alto nivel de derivación, debido a que el número de operaciones primitivas, suma y multiplicación, que hay que realizar crece con la complejidad de la función $h(k)$. Además, la determinación de la cascada de operaciones a realizar requiere cierto empirismo ya que depende del método de cálculo empleado. La diversidad de métodos de cálculo puede ocasionar, además, gran diversidad en los mecanismos de derivación. Este tipo de evaluación aparece, por tanto, como poco sistemático, casi con cierto carácter artesanal. La evaluación estructural del tipo $h(k) = f(k)*g(k)$ tiene una complejidad $O(n^2)$ con respecto al número n de puntos a calcular porque sólo tiene el coste de combinación, aunque requiere el conocimiento previo de las funciones que intervienen, $f(k)$ y $g(k)$. Sin embargo el coste sigue siendo alto debido a que la secuencia de operaciones a realizar crece a medida que aumenta n. De estas consideraciones se desprende que la mejora puede venir por la vía de rebajar el coste de combinación, reutilizando los resultados previos de la convolución en los

cálculos posteriores. Estas consideraciones orientan la búsqueda hacia una operatoria recursiva, con sólo dos recursos de derivación que son inicializar e iterar. Si, además, esta operatoria es única y se parametriza, el método de evaluación es totalmente sistemático y el empirismo está en la búsqueda de los parámetros caracterizadores de la función cuya evaluación se está realizando, no en la secuencia de mecanismos de derivación a utilizar. Dado el ahorro de recursos, se prevé ganancia de rendimiento computacional así como mayor robustez, por el carácter sistemático del planteamiento.

En lo que sigue se van a sentar las bases de la teoría que sustenta la evaluación no estructural buscada.

4 Desarrollo formal del CBRM

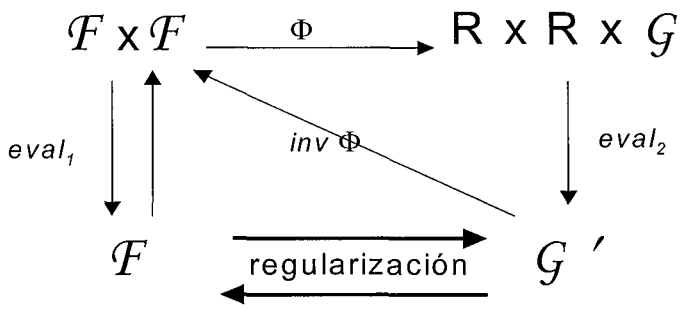

Fig 2.1 Planteamiento formal del CBRM

Sea $\mathcal{F}$ el conjunto de funciones discretas de variable discreta definidas en R. Sea la aplicación $eval_1$:

$$eval_1 : \mathcal{F} \times \mathcal{F} \longrightarrow \mathcal{F}$$

$$(f, g) \longrightarrow eval_1(f, g) = eval_1(g, f) = f * g = g * f \qquad (2.6)$$

$eval_1$ define la convolución como una operación entre dos funciones discretas, que representa la evaluación de una de ellas tomando la otra como unidad (expresión (2.5)).

Se puede demostrar que $(\mathcal{F}, *)$ es un grupo abeliano.

Sea $\mathcal{G}$ el conjunto de funciones definidas de N en R

$$F: \mathsf{N} \longrightarrow \mathsf{R}$$
$$i \longrightarrow F_i$$

Se define $\mathcal{G}' \subset \mathcal{G}$ de la manera siguiente:

$F \in \mathcal{G}' \Leftrightarrow \exists\, (\alpha, \beta) \in \mathsf{R}, \exists\, G \in \mathcal{G}$ tales que

$F_i = \alpha F_{i-1} + \beta G_{i-1}$

$(\mathcal{G}', +, \mathsf{x})$ es un espacio vectorial sobre R para cada par (α, β)

Se define la aplicación $eval_2$:

$$eval_2 : \mathsf{R} \times \mathsf{R} \times \mathcal{G} \longrightarrow \mathcal{G}'$$

$$(\alpha, \beta, G) \longrightarrow eval_2(\alpha, \beta, G) = F \qquad (2.7)$$

$eval_2$ define la función F como una función recursiva evaluada iterativamente por la suma ponderada de F y G, siendo α, β los parámetros de ponderación que indican la contribución de F y G en F, en cada iteración.

Se define ahora la aplicación ϕ de $\mathcal{F} \times \mathcal{F}$ en $\mathsf{R} \times \mathsf{R} \times \mathcal{G}$ de la manera siguiente:

$$\phi: \mathcal{F} \times \mathcal{F} \longrightarrow \mathsf{R} \times \mathsf{R} \times \mathcal{G}$$

$$(f, g) \longrightarrow \phi(f, g) = (\alpha, \beta, G) \qquad (2.8)$$

tal que, $\forall y \in [x, \infty[\subset \mathsf{R}, \exists\, i \in \mathsf{N} / f * g\,(y) \equiv F_i = \alpha F_{i-1} + \beta G_i$

Condición necesaria para la regularización de la convolución sobre la fórmula recursiva

La restricción expresada en (2.8) de que todos los valores de una convolución encuentren igualación con valores sucesivos de una función evaluada iterativamente, F, es la condición necesaria para regularizar la convolución y, de esta manera, poder contar con una evaluación de tipo no estructural, basada en la evaluación estructural que proporciona la convolución. Las condiciones bajo las cuales se establece dicha igualdad son relevantes en cuanto al alcance de la evaluación no estructural inducida. A continuación se detalla el desarrollo de estas condiciones.

Sea el desarrollo formal de la convolución de dos funciones, f y g, pertenecientes a $\mathcal{F}$. Se realiza a partir de un punto inicial, x, del intervalo $[x, \infty[\subset \mathsf{R}$, con un paso $h \in \mathsf{R}$.

Los k primeros valores de la convolución $f*g$ son:

$$f*g(x)=f(x)g(x)$$
$$f*g(x+h)=f(x)g(x+h)+f(x+h)g(h)$$
....
$$f*g(x+kh)=f(x)g(x+kh)+f(x+h)g(x+(k-1)h)+.......$$
$$...+f(x+(k-1)h)g(x+h)+f(x+kh)g(x) \qquad (2.9)$$
....

$$f*g(x+kh)=\sum_{p=0}^{p=k} f(x+ph)g(x+(k-p)h)$$
....

Se puede modificar este desarrollo haciendo que aparezca la derivada de alguna de las dos funciones. En (2.10) aparece la derivada de la función f entre paréntesis. Como las dos funciones son intercambiables en relación con el papel que desempeñan, se puede llegar a una expresión análoga a (2.10) en la que aparece la derivada de la función g.

$$f*g(x)=f(x)g(x)$$
$$f*g(x+h)=f*g(x)+hg(x)\frac{f(x+h)-f(x)}{h}+f(x)g(x+h)$$
$$f*g(x+2h)=f*g(x+h)+hg(x)\frac{f(x+2h)-f(x+h)}{h}$$
$$+hg(x+h)\frac{f(x+h)-f(x)}{h}+f(x)g(x+2h)$$
....
$$.f*g(x+kh)=f*g(x+(k-1)h)+hg(x)\frac{f(x+kh)-f(x+(k-1)h}{h}.............$$
$$....................+hg(x+(k-1)h)\frac{f(x+h)-f(x)}{h}+f(x)g(x+kh) \qquad (2.10)$$
....

$$f*g(x+kh)=f*g(x+(k-1)h)+h\sum_{p=0}^{p=k-1} g(x+ph)f'(x+(k-p)h)$$
$$+f(x)g(x+kh)$$

....

La expresión obtenida en (2.10) ha de igualarse con la función recursiva dada por la ecuación (2.8), $F_i = \alpha F_{i-1} + \beta G_{i-1}$

Para ello se propone:

- hacer $x \equiv 0$
- hacer $h = 1$
- hacer $i \equiv k$

Así, se asimilan

- $f*g(x + kh)$ con F_i
- βG_{i-1} con $h\sum_{p=0}^{p=k-1} g(x+ph)f'(x+(k-p)h) + f(x)g(x+kh)$

La interpretación de la regularización propuesta es la siguiente:

- El argumento inicial, $x \in \mathsf{R}$, para el cual se evalúa la convolución se regulariza sobre el índice inicial de la fórmula recursiva, F, que es el cero.
- Elegido el paso $h \in \mathsf{R}$ con el cual se evalúa la convolución, se regulariza éste sobre el paso de la fórmula recursiva, F, que es 1.
- Con estas dos regularizaciones, el número de iteración de la convolución y de la fórmula recursiva coinciden, permitiendo confundir en todo lo que sigue las expresiones $f*g(k)$ y F_k para simplificar la escritura.

El parámetro α que interviene explícitamente en (2.8) ha sido añadido a fin de recoger la posibilidad de que, entre los factores que intervienen en la parte del sumatorio, alguno pueda contener de forma implícita el término $f*g(x + kh)$ y dar lugar a factorización (ver Tablas 2.1 y 2.2).

Condición suficiente para la regularización de la convolución sobre la fórmula recursiva

La condición suficiente para la regularización de la convolución está relacionada con la posibilidad de encontrar la inversa de la función ϕ. Ello significa que, dada una forma recursiva del tipo (2.8), se pueda encontrar un par de funciones f y g cuya convolución tenga la forma (2.8).

Sea una función recursiva Y_i, del tipo (2.9).
Entonces Y_i cumple:

$$\begin{aligned} &Y_0 \\ &Y_1 = \alpha Y_0 + \beta Z_0 \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ &Y_{i-1} = \alpha Y_{i-2} + \beta Z_{i-2} \\ &Y_i = \alpha Y_{i-1} + \beta Z_{i-1} \\ &Y_{i+1} = \alpha Y_i + \beta Z_i \end{aligned} \qquad (2.11)$$

Siendo α,β diferentes de cero y Z_i una función auxiliar

Sustituyendo el valor de Y_i en la última ecuación de (2.11) por su valor, que expresa la penúltima ecuación y remontando hasta la primera ecuación aparece el desarrollo de la convolución de dos funciones f y g sin más que identificar:

$$\begin{aligned} &f(i) = \alpha^i \\ &g(i) = \beta Z_{i-1} \\ &g(0) = Y_0 \end{aligned} \qquad (2.12)$$

Si α ó β son iguales a cero, las ecuaciones (2.11) se simplifican. En estos casos, las expresiones de f y g son:

$$\begin{array}{ll} \alpha = 0 & \beta = 0 \\ f(i) = \delta(i) & f(i) = \delta(i) \\ g(i) = \beta Z_{i-1} & g(i) = \alpha^i g(0) \\ g(0) = Y_0 & g(0) = Y_0 \end{array} \qquad (2.13)$$

Por tanto, cualquier función desarrollada bajo la forma recursiva (2.11) puede representar la convolución de dos funciones *f* y *g*. En los casos descritos por las ecuaciones (2.13), se tiene una convolución trivial, ya que una de las funciones que intervienen es la delta de Dirac.

Esta propuesta de regularización no es única, como tampoco lo es la forma de modificar el desarrollo (2.9), que conduce a la expresión (2.10). Otras modificaciones pueden conducir a formas recursivas de la convolución en las que aparecen no sólo el valor en la iteración anterior sino otros valores anteriores o incluso todos los valores anteriores a la iteración considerada. No se considera en este trabajo de investigación más que una de las posibles regularizaciones de la convolución, quedando las demás como líneas de investigación futuras.

5 Aplicación del CBRM

Las bases establecidas en el apartado anterior permiten desarrollar un método de evaluación no estructural, CBRM, válido para un conjunto amplio de funciones, que ha de culminar con la definición de unas primitivas. Importa, para ello, concretar la correspondencia entre las características de las funciones *f*, *g* y los parámetros de regularización de la convolución, α, β, *G*. El desarrollo de la convolución dado por la ecuación (2.10) puede simplificarse notablemente para

algunos tipos de funciones usuales, obteniéndose un patrón de construcción de la evaluación no estructural muy sencillo, a la medida de esas funciones. Es interesante considerar estos casos que, aunque correspondan a particularizaciones del caso general, tienen gran utilidad por ser casos muy frecuentes.

- Cuando una de las dos funciones de la convolución es del tipo potencial, por ejemplo, $f(x + kh) = K^{x+kh}$, la expresión recursiva (2.10) se transforma en:

$f*g(x + kh) = K^h . f*g(x +(k-1)h) + K^x . g(x+kh)$

que es la fórmula recursiva (2.7) con $\alpha = K^h$, $\beta = K^x$, $G = g(x+kh)$

- Cuando una de las dos funciones es constante, por ejemplo $f(x + kh) = M$, $f'(x + kh) = 0$ y los términos del sumatorio de la expresión (2.10) se anulan; la fórmula resultante es:

$f*g(x + kh) = . f*g(x +(k-1)h) + M\, g(x+kh)$

que corresponde a $\alpha = 1$, $\beta = M$, $G = g(x+kh)$

- Cuando una de las dos funciones es lineal, por ejemplo $f(x+kh) = p(x+kh)$, los términos del sumatorio de la expresión (2.10) son todos iguales a la pendiente de la recta, esto es, a ph. La expresión resultante es entonces:

$f*g(x + kh) = . f*g(x +(k-1)h) + ph\, [g(x)+g(x+h)+.....+ g(x+(k-1)h)]$

que corresponde a $\alpha = 1$, $\beta = ph$, $G = \sum_{j=0}^{k-1} g(x + jh)$

Estos tres casos permiten abordar de manera muy sencilla la evaluación recursiva de un gran número de funciones. De la aplicación de las simplificaciones anteriores se deriva la posibilidad de sistematizar la evaluación, construyendo una "tabla de equivalencia" que facilita automáticamente la fórmula recursiva que corresponde a una evaluación por convolución dada. Al consultar dicha tabla, se encuentran para cada par (f, g) los valores (α, β, G) que corresponden. La Tabla 2.1 muestra estas correspondencias para algunas funciones usuales. Por motivos de claridad y sin pérdida de generalidad, se han expresado las funciones considerando x=0 y h=1. Se han elegido casos en los que, al menos una de las funciones que interviene en la convolución, es de las tres que proporcionan simplificación (función constante, lineal o potencial). De no ser así, se realiza la correspondencia basándose en el caso general dado por la ecuación (2.10).

$f^*g(k)$	$g(k) = \delta(k)$	$g(k) = M$	$g(k) = pk$	$g(k) = K^k$
$f(k) = \delta(k)$	$\delta(k)$	M	pk	K^k
$f(k) = N$	N	$\alpha=1\ \beta=M\ G_k=N$ $G_0=MN$	$\alpha=1\ \beta=p\ G_k=N\,k$ $G_0=0$	$\alpha=K\ \beta=1\ G_k=N$ $G_0=N$
$f(k) = qk$	qk	$\alpha=1\ \beta=M\ G_k=qk$ $G_0=0$	$\alpha=1\ \beta=p$ $G_k=\frac{qk(k-1)}{2}$ $G_0=0$	$\alpha=K\ \beta=1\ G_k=qk$ $G_0=0$
$f(k) = a^k$	a^k	$\alpha=a\ \beta=1\ G_k=M$ $G_0=M$	$\alpha=\alpha\ \beta=1\ G_k=pk$ $G_0=0$	$\alpha=K\ \beta=1\ G_k=a^k$ $G_0=1$
$g(k) = \cos k$	$\cos k$	$\alpha=1\ \beta=M$ $G_k=\cos k$ $G_0=M$	$\alpha=1\ \beta=p$ $G_k=\Sigma \cos i$ $(0 \le i < k)$ $G_0=0$	$\alpha=K\ \beta=1$ $G_k=\cos k$ $G_0=1$
$g(k)=\text{sen } k$	$\text{sen } k$	$\alpha=1\ \beta=M$ $G_k=\text{sen } k$ $G_0=0$	$\alpha=1\ \beta=p$ $G_k=\Sigma \text{sen } i$ $(0 \le i < k)$ $G_0=0$	$\alpha=K\ \beta=1$ $G_k=\text{sen } k$ $G_0=0$
$g(k)=\text{Log } k$	$\text{Log } k$	$\alpha=1\ \beta=M$ $G_k=\text{Log } k$ $G_0=0$	$\alpha=p\ \beta=p$ $G_k=\text{Log } k$ $G_0=0$	$\alpha=K\ \beta=K$ $G_k=\text{Log } k$ $G_0=0$
$g(k)=1/k$	$1/k$	$\alpha=1\ \beta=M$ $G_k=1/(k-1)$ $G_0=M$	$\alpha=1\ \beta=p$ $G_k=\Sigma\, 1/i$ $(0 < i < k)$ $G_0=p$	$\alpha=K\ \beta=K$ $G_k=1/(k-1)$ $G_0=K$

Tabla 2.1 Tabla de equivalencia de algunas funciones usuales

Recíprocamente, se sistematiza la descomposición de una función dada en convolución de otras dos. En la Tabla 2.2 se muestra la descomposición de algunas funciones usuales a partir de la forma recursiva que se les puede asociar. Para dos de ellas, la función constante y la función potencial, la única descomposición alcanzable es la descomposición trivial.

Funciones usuales $F(x+kh)$	**Parámetros de la fórmula recursiva** $F_k = \alpha F_{k-1} + \beta G_k$			**Descomposición en convolución de dos funciones** $F_k \equiv f * g(x+kh)$	
	α	β	G_k	$f(x+kh)$	$g(x+kh)$
constante M	0	1	M	$\delta(x+kh)$	$g(x) = M$ $g(x+kh) = M$
lineal $m(x+kh)$	1	h	m	1	$g(x) = mx$ $g(x+kh) = mh$
exponencial a^{x+kh}	a^h	0	0	$\delta(x+kh)$	$g(x) = a^x$ $g(x+kh) = a^{x+kh}$
inversa $1/(x+kh)$	1	-1	$G_k = \sum_{k=1}^{\infty}(*)$	1	$g(x+kh) = -\sum_{k=1}^{\infty}(*)$
Raíz-m-ésima $\sqrt[m]{x+kh}$	1	1	$G_k = \sum_{k=1}^{\infty}(**)$	1	$g(x+kh) = \sum_{k=1}^{\infty}(**)$
logaritmo $\log(1+kh)$	1	1	$G_k = \sum_{k=1}^{\infty}(***)$	1	$g(x+kh) = \sum_{k=1}^{\infty}(***)$
trigonométrica $\cos(x+kh)$	$\alpha = \cos(h)$	$\beta = -\operatorname{sen}(h)$	$G_0 = \cos x/\beta$ $G_k = -\operatorname{sen}(x+(k-1)h)$	α^{x+kh}	$g(x) = \cos x$ $g(x+kh)) = -\beta \operatorname{sen}(x+(k-1)h)$
$\operatorname{sen}(x+kh)$	$\alpha = \operatorname{sen}(h)$	$\beta = \cos(h)$	$G_0 = \operatorname{sen} x/\beta$ $G_k = \cos(x+(k-1)h)$	α^{x+kh}	$g(x) = \operatorname{sen} x$ $g(x+kh) = \beta \cos(x+(k-1)h)$
hiperbólica $\cosh(x+kh)$	$\alpha = \cosh(h)$	$\beta = \operatorname{senh}(h)$	$G_0 = \cosh x/\beta$ $G_k = \operatorname{senh}(x+(k-1)h)$	α^{x+kh}	$g(x) = \cosh x$ $g(x+kh) = \beta \operatorname{senh}(x+(k-1)h)$
$\operatorname{senh}(x+kh)$	$\alpha = \operatorname{senh}(h)$	$\beta = \cosh(h)$	$G_0 = \operatorname{senh} x/\beta$ $G_k = \cosh(x+(k-1)h)$	α^{x+kh}	$g(x) = \operatorname{senh} x$ $g(x+kh) = \beta \cosh(x+(k-1)h)$

$\Sigma(*) = \Sigma[(k+1)^n - k^n] \cdot h^n / x^{n+1}$

$\Sigma(**) = \Sigma(1-m)(1-2m).....(1-nm)x^{(1-nm)/m}[(k+1)^n - k^n)\, h^n)]/\, n!\, m^{n+1}$

$\Sigma(***) = \Sigma(-1)^n[k^n - (k-1)^n]\, h^n / n$

Tabla 2.2 Descomposición de algunas funciones usuales.

En las expresiones de F_k de la Tabla 2.2, x representa el primer valor para el cual se calcula el valor de la función, h representa el paso de iteración y k el número de iteración.

6 Primitivas y derivadas

En una metodología de evaluación, la propuesta de primitivas es una etapa crucial. Conceptualmente, las primitivas son las operaciones básicas de referencia del proceso de evaluación sobre las cuales se construyen los niveles de derivación.

En el contexto de la evaluación no estructural, una primitiva $p_{\alpha\beta}$ es una expresión paramétrica del tipo

$p_{\alpha\beta} \equiv \alpha f + \beta g$ donde α, f, β, g son números reales.

Se define un nivel de derivación fijando un valor inicial $f = F_0$ e iterando la primitiva $p_{\alpha\beta}$ cuantas veces se quiera, de manera que en cada iteración se obtiene un valor F_i tal que

$F_i = \alpha F_{i-1} + \beta G_{i-1}$

En esta derivación α y β se mantienen constantes y cada nuevo valor $g = G_i$ incorporado determina un nuevo valor F_i. Por tanto, se tiene

$i = 0 \quad F_0$

$i = 1 \quad F_1 = \alpha F_0 + \beta G_0$

$i = 2 \quad F_2 = \alpha F_1 + \beta G_1$

.....

El conjunto de valores $\{F_i\}$ obtenidos por cálculo iterativo son puntos de la función F evaluada, ordenados en el sentido creciente de las iteraciones i. Asimismo, los valores incorporados $\{G_i\}$ son puntos de una función G llamada función auxiliar, también ordenados en el sentido creciente de i.

Las funciones F son clasificables en familias. Una familia de funciones es un conjunto de funciones que muestra un mismo comportamiento. Éste viene definido por los intervalos de pertenencia de α, β y por G.

Los niveles de derivación sucesivos se relacionan entre sí por medio de la función auxiliar. Suponiendo que en un nivel D_n de derivación se ha evaluado una función F, con los parámetros α, β y la función auxiliar G, cualquier función H que se pueda evaluar con los parámetros α', β' y la función auxiliar F se considerará de nivel de derivación D_{n+1}. Por tanto, al nivel de derivación D_{n+1} le corresponden funciones auxiliares calculadas en el nivel de derivación anterior. Por convenio, se puede proponer como nivel primero de derivación D_1 aquel en el cual las funciones evaluadas son del tipo $h(k) = h(k)*\delta(k)$ como, por ejemplo, la función constante y la función potencial.

A continuación se presenta un ejemplo de evaluación no estructural que corresponde al caso asociado a la convolución de una función constante por una función potencial ($f(k) = a^k$, $g(k) = M$). Para este caso, la tabla de equivalencia da los valores $\alpha = a$, $\beta = 1$ y $G = M$. Estudiando los intervalos de variación de los parámetros se ponen en evidencia distintos comportamientos que definen familias de funciones. Posteriormente se realiza un mapa de localización de éstas en el cual los valores de α, β y G actúan como coordenadas (Fig.2.8). La Tabla 2.3 recoge la descripción del comportamiento definitorio de cada familia.

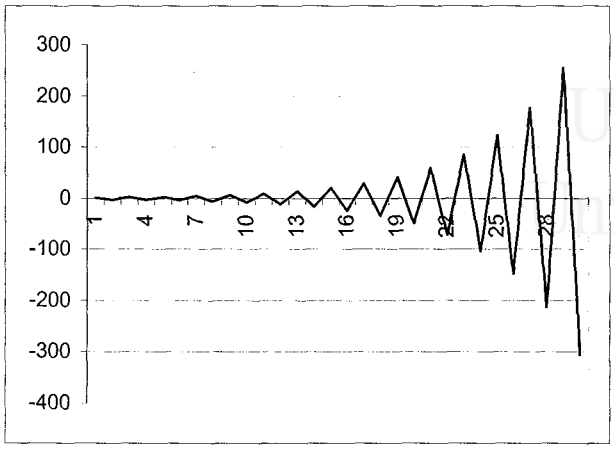

Fig.2.2 Familia 1 ($\alpha = -1.2$, $\beta G = -1.2$)

Familia 1:

$\alpha \leq -1$

Se observa una oscilación divergente, más rápida para valores más negativos de α. La amplitud es también mayor para valores más negativos de βG, pero la influencia de βG es más moderada que la de α. (Fig.2.2)

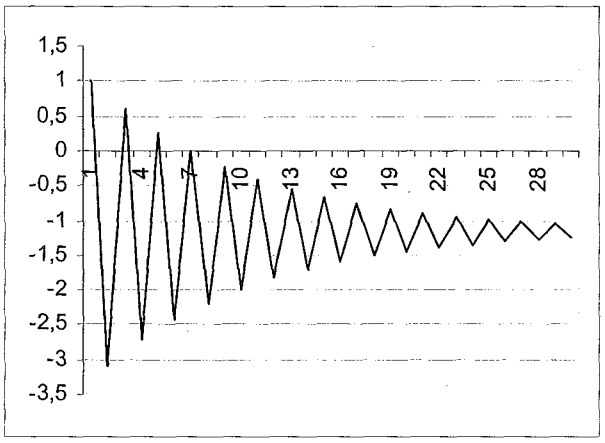

Fig.2.3 Familia 2 ($\alpha = -0.9$, $\beta G = -2.2$)

Familia 2

$-1 < \alpha \leq 0$

Se observa una oscilación convergente, más rápida para valores de α más cerca de cero. (Fig.2.3). El signo de βG fija la convergencia en torno a un valor que es de su mismo signo. El valor absoluto de βG influye en la amplitud de la oscilación y, por tanto, en el valor de convergencia.

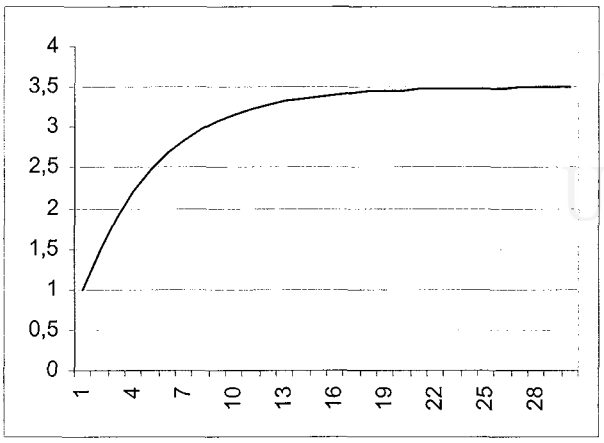

Fig.2.4 Familia 3 ($\alpha = 0.8$, $\beta G = 0.7$)

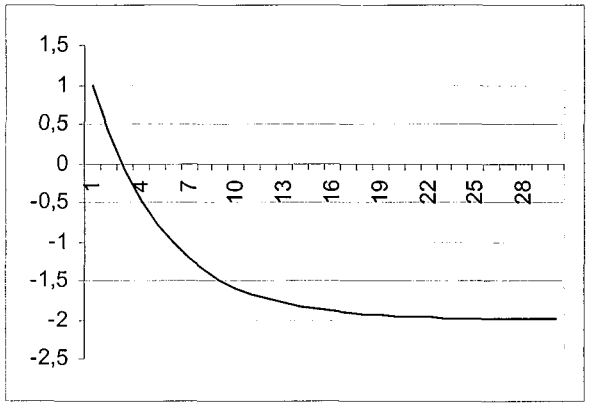

Fig.2.5 Familia 4 ($\alpha = 0.8$ $\beta G = -0.4$)

Familias 3 y 4:

$0 < \alpha \leq 1$

Se observa una evolución continua convergente con asíntota horizontal, más abrupta cuanto más se acerca α a cero. βG fija la dirección de la evolución, crecimiento o decrecimiento, así como el valor de la convergencia. A valores más negativos le corresponde un valor más negativo del valor de convergencia y a valores más positivos un valor más positivo. El cambio cualitativo entre el crecimiento y el decrecimiento tiene lugar cuando se cumple $\beta G + \alpha = 1$ (Fig.2.4 y 2.5)

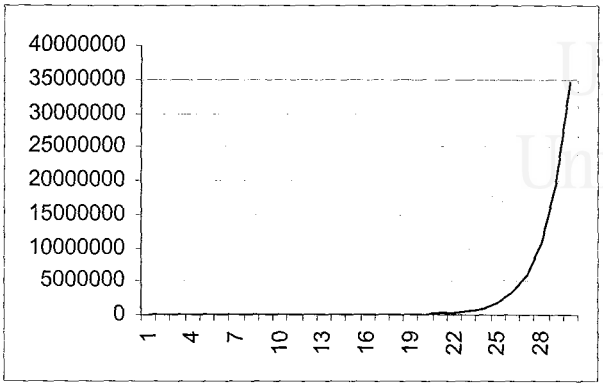

Fig 2.6 Familia 5 (α=1.8, βG= 0.3)

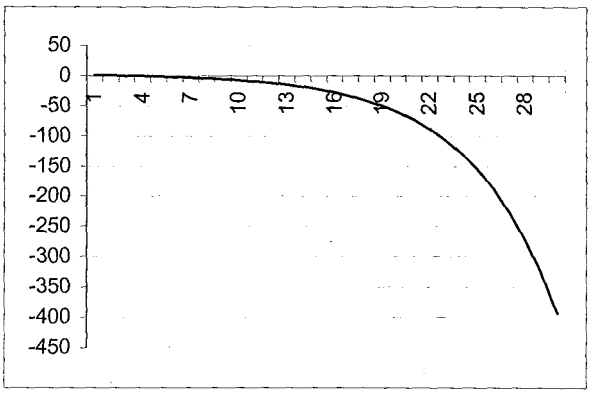

Fig.2.7 Familia 6 (α=1.2, βG = −1.6)

Familias 5 y 6:

$\alpha > 1$

Se observa una evolución continua divergente con asíntota vertical, más abrupta cuanto mayor es α y cuanto mayor es $|\beta G|$, aunque éste influye de forma más moderada. El parámetro βG fija la dirección de la evolución, crecimiento o decrecimiento El cambio cualitativo de uno a otro tiene lugar para $\beta G + \alpha = 1$ (Fig.2.6 y 2.7)

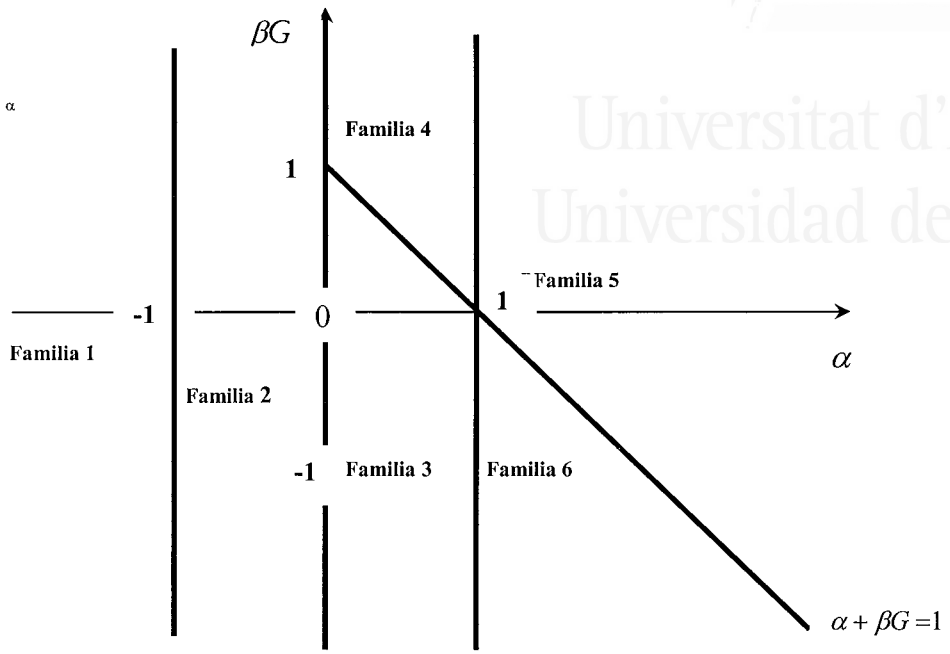

Fig 2.8 Mapa bidimensional de localización de las familias de funciones asociadas a la convolución de constante por potencial

Familias asociadas a la convolución de una función constante por una función potencial		
Familia	**Intervalos de pertenencia de los parámetros**	**Comportamiento cualitativo**
1	$\alpha \in]-\infty, -1]$ $\beta G \in \mathcal{R}_+$	Oscilación divergente
2	$I\alpha \in]-1, 0]$ $\beta G \in \mathcal{R}_+$	Oscilación convergente
3	$\alpha \in]0, 1]$ $\alpha+\beta G > 1$	Crecimiento con saturación
4	$\alpha \in]0, 1]$ $\alpha+\beta G < 1$	Decrecimiento con saturación
5	$\alpha \in]1, +\infty]$ $\alpha+\beta G > 1$	Crecimiento con asíntota vertical
6	$\alpha \in]1, +\infty]$ $\alpha+\beta G < 1$	Decrecimiento con asíntota vertical

Tabla 2.3 Comportamiento correspondiente a las familias de funciones asociadas a la convolución de constante por potencial

Con este ejemplo, se ha puesto de manifiesto la relación que existe entre los dos tipos de evaluación, estructural (convolución) y no estructural (recursiva). Cada evaluación por convolución se traduce a la evaluación recursiva, por un conjunto finito de familias de funciones. Cada familia representa un comportamiento diferenciado y puede instanciarlo un número infinito de veces por concreción de los parámetros dentro del intervalo que corresponde. Al ejemplo bidimensional que se acaba de tratar, le corresponden seis familias. Se puede realizar un estudio similar para cualquier evaluación por convolución.

7 Conclusión

En este capítulo se sientan las bases teóricas de la resolución del problema formulado en la introducción, que es la obtención de primitivas capaces de realizar a bajo nivel el cálculo de funciones que habitualmente necesitan más niveles de derivación. Como estrategia de evaluación, se establece la convolución de funciones a la cual se asocia, por un proceso de regularización, una operatoria recursiva que cumple el requerimiento de la disminución de niveles para un gran número de funciones. Las primitivas propuestas tienen una forma paramétrica estándar muy sencilla y los rasgos diferenciadores de las funciones que se calculan se deben a los valores concretos de los parámetros. Las primitivas son, por tanto, valores numéricos que actúan como semilla en el cálculo iterativo de una función. Los niveles de derivación sucesivos se relacionan entre sí por medio de la función auxiliar.

Capítulo 3

ARQUITECTURAS CBRM

1 Introducción

Este capítulo aborda el diseño y evaluación de arquitecturas que instrumentan el método recursivo de evaluación basado en convolución, CBRM. Consiste en trasladar al plano de realización física la operación recursiva que se ha propuesto. Para ello, se diseña un prototipo de procesador, facilitando una descripción funcional de sus módulos principales. Se realiza la evaluación del prototipo con resultados o estimaciones concretas de las magnitudes más significativas desde el punto de vista arquitectural, como el tiempo de cálculo y el área utilizada. Para realizar estimaciones de tiempo y área fácilmente comparables con las de otras propuestas se ha utilizado como unidad de medida un dispositivo elemental de cálculo, formado por una puerta XOR y una AND, que implementa un sumador completo de un bit. Posteriormente, se han efectuado también mediciones en dispositivos de lógica reconfigurable FPGA (*Field Programmable Gate Array*). Descritos en lenguaje VHDL (*Hardware Description Language*), los circuitos han sido simulados en la tarjeta. xcv300e-6bg352-XST de Xilinx, proporcionando una estimación de los tiempos de cálculo y de los recursos hardware. Abordar los aspectos de disipación de potencia sería procedente en versiones de mayor carácter aplicado, lo que excede el alcance de esta memoria. En su caso se

plantearía una realización ASIC (*Application Specific Integrated Circuit*) del procesador básico.

El procesador CBRM tiene su techo de velocidad establecido por la operatoria recursiva que realiza, que sólo es capaz de proporcionar un punto por iteración. Esta limitación es un inconveniente para aplicaciones que requieran mayor rapidez y sugiere buscar mejoras. Las prestaciones aumentan incorporando distintos grados de paralelismo; en el módulo mismo y en las arquitecturas que involucran varios módulos interconectados. En todos los casos, se establecen comparaciones de tiempo y área y se opta por una solución combinada y flexible de cálculo en serie y en paralelo, como solución intermedia y adaptable a los requerimientos del problema a tratar.

2 Arquitectura del procesador CBRM

En este apartado se va a proponer la arquitectura del procesador CBRM que corresponde a la modalidad básica de operatoria. Esta arquitectura consta de tres módulos principales: el de cálculo, donde se realiza la operación recursiva, el de control, que se ocupa de gestionar la ejecución del cálculo y la memoria que almacena los datos. Estas funciones pueden implementarse de formas muy diversas. Puede resultar interesante concentrar todos los datos en una sola memoria o bien, alternativamente, dotar también de memoria a los módulos de cálculo y de control. Por otra parte, algunos datos pueden existir almacenados permanentemente o calcularse, bien en línea bien ex profeso en una etapa previa de cálculo. En la descripción de los módulos se mencionan las distintas posibilidades, teniendo en cuenta que la evaluación de la arquitectura depende finalmente de la implementación elegida.

2.1 Descripción funcional de los módulos

La figura 3.1 representa la estructura general del procesador CBRM que consta de tres módulos: el de cálculo, el de control y la memoria.

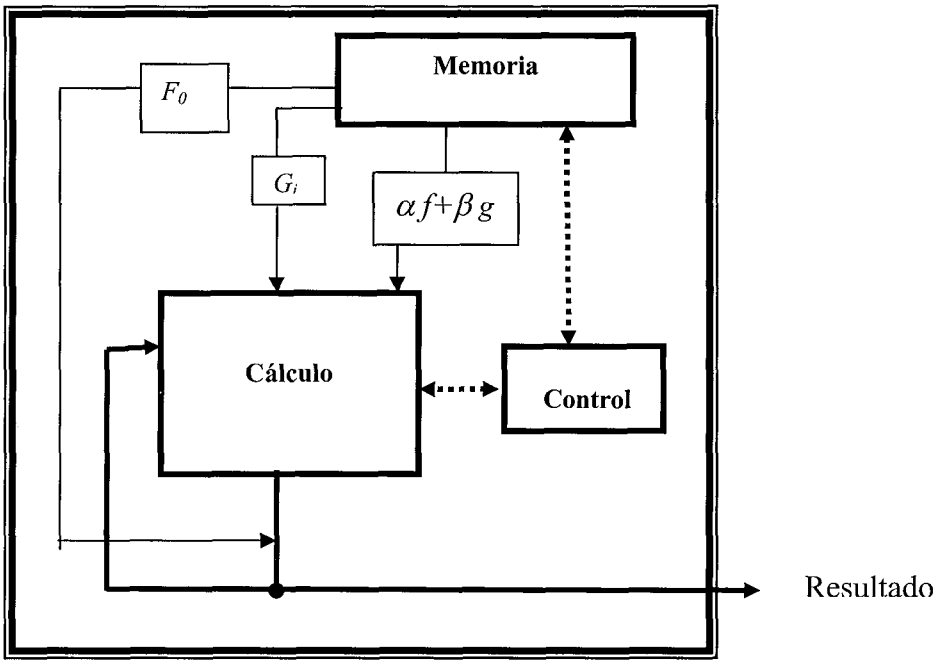

Fig 3.1. Estructura general del procesador CBRM que presenta tres módulos principales

Módulo de cálculo

El módulo de cálculo proporciona el resultado de la operación recursiva. Al inicio de la etapa de cálculo de una función F, están disponibles el valor inicial F_0 de la función, así como los valores de los parámetros α, β y la función auxiliar G que caracterizan la primitiva que hay que usar. El resultado puede alcanzarse mediante el cálculo, que involucra dos multiplicaciones y una suma, o

permanecer precalculado. La figura 3.2 representa el esquema funcional del módulo de cálculo que efectúa la operación recursiva.

$F_i = \alpha F_{i-1} + \beta G_{i-1}$ donde

F_i es el valor, en la iteración i, de la función que se calcula

F_{i-1} es el valor que tenía la función en la iteración anterior, $i-1$

α, β son los parámetros propios de la primitiva que calcula la función F

G_{i-1} es el valor de la función auxiliar que se incorpora en la iteración i

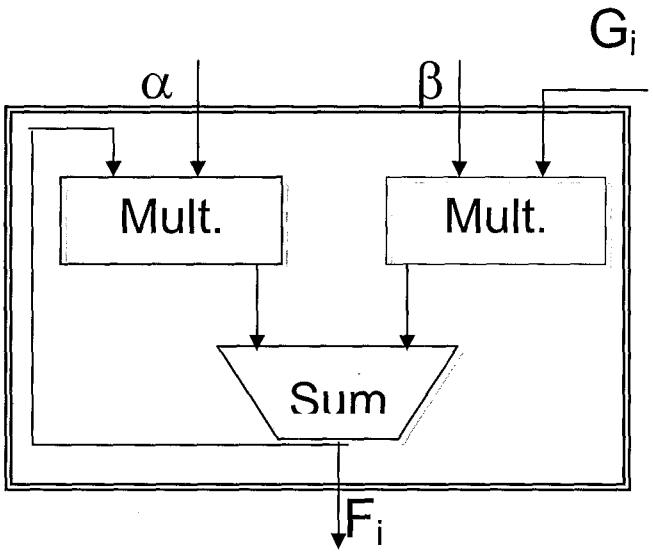

Fig 3.2 Esquema funcional del módulo de cálculo

Módulo de control

El módulo de control gestiona la ejecución del cálculo mediante señales de control. El control consiste en asegurar la correcta selección de los parámetros, la captación y el paso de argumentos al módulo de cálculo, en guiar la salida de resultados y contar el número de iteraciones para marcar el final de la operación.

Memorias

En caso de querer implementar memorias en los módulos de cálculo y de control se establece el desglose siguiente para los datos que han de contener:

- Memoria de control

Contiene los datos necesarios para definir y controlar la ejecución del cálculo. El número de iteraciones, k, que coincide con el número de puntos a calcular, está relacionado con el intervalo a recorrer I y con el paso de iteración h, por la ecuación $I = k.h$.

- Memoria de cálculo

Contiene los parámetros α y β, que están generalmente relacionados con el paso h, así como los puntos de la función auxiliar G. La primitiva $\alpha f+\beta g$ que hay que emplear puede calcularse en línea, por medio de dos multiplicaciones y una suma o también permanecer precalculada. En este caso la operación consiste en direccionar la tabla por algunos bits particulares de los valores de F (que es la función calculada) y de G que es la función auxiliar. Si $F \equiv f$ y $G \equiv g$, sólo hará falta un acceso para captar el valor, en cambio si f y g representan fragmentos de F y G harán falta más accesos para completar el valor del punto $\alpha F+\beta G$.

Si se opta por una memoria única, ésta contendrá todos los datos mencionados.

El camino seguido por los datos previo a la ejecución del cálculo es el siguiente (Fig 3.3):

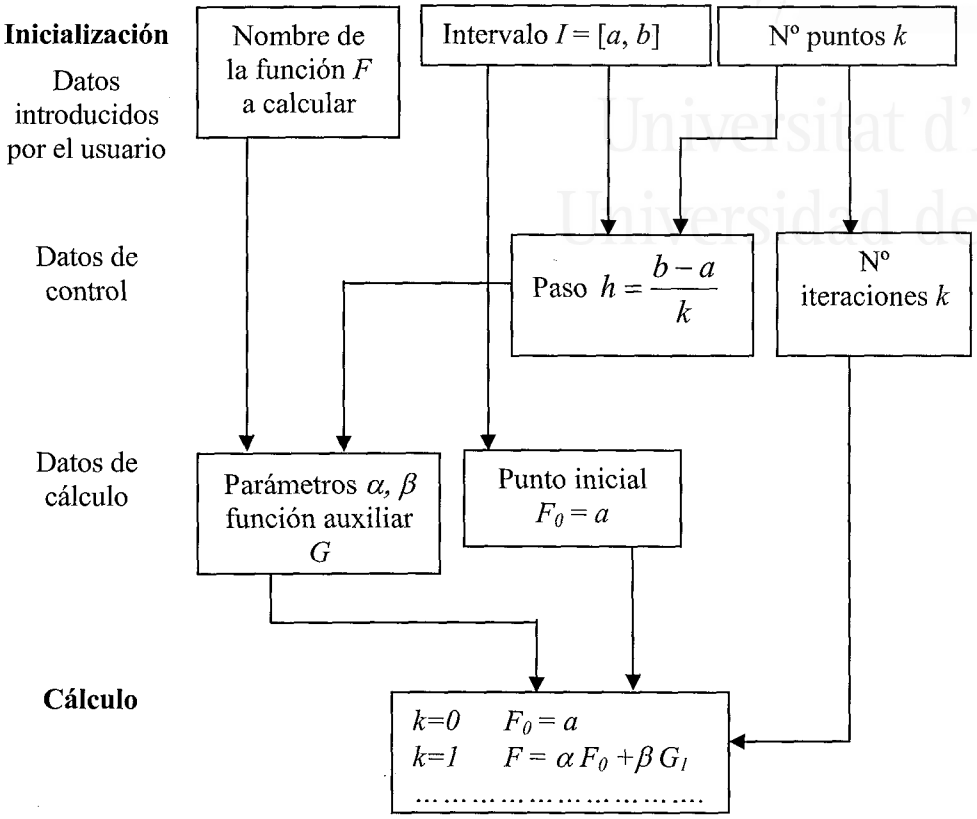

Fig 3.3 Camino seguido por los datos

2.2 Implementación

En esta investigación, el propósito de la implementación es la realización de un prototipo que sea capaz de mostrar el método CBRM. Por tanto, no se plantea en ningún momento que la implementación elegida tenga que ser óptima y, menos aún, que el método expuesto dependa de implementación alguna. Por ello, sólo se han considerado los aspectos de diseño que conciernen a la posibilidad de reutilización, al coste de desarrollo hardware, a la flexibilidad y a la limitación en el número y diversidad de los módulos necesarios; cuestiones que importan en la

etapa de verificación del prototipo. Además, se ha centrado toda la atención en la implementación del módulo de cálculo, considerado como la parte principal del prototipo.

Se ha optado finalmente por no efectuar las dos multiplicaciones y la suma para calcular los puntos de la función. En su lugar se tendrán los resultados disponibles en una tabla LUT (*Look-up table*), de acceso paralelo, utilizando aritmética distribuida para su almacenamiento. Los resultados que figuran en la LUT son instanciaciones de la primitiva $\alpha f+\beta g$, donde f y g son grupos de bits que representan fragmentos de las funciones F y G, de manera que para obtener cualquier punto $\alpha F+\beta G$, hay que realizar tantos accesos a la LUT como fragmentos del tamaño de f haya en F (igual al número de fragmentos del tamaño de g que hay en G) y recomponer posteriormente el valor $\alpha F + \beta G$ mediante una suma. Suponiendo que F y G tienen un tamaño de n bits, si se considera una fragmentación en p partes, los fragmentos tendrán un tamaño de $t = n/p$ bits. Por tanto, el número de celdas de la LUT es 2^{2t}. La tabla 3.1 muestra un ejemplo de tabla LUT en la cual los fragmentos f y g son de dos bits ($f = (f_1, f_2)$ y $g = (g_1, g_2)$). Cabe mencionar que uno de los bits de cada fragmento es el bit de signo que corresponde al operando.

	$(f_2, g_2) = (00)$	$(f_2, g_2) = (01)$	$(f_2, g_2) = (10)$	$(f_2, g_2) = (11)$
$(f_1, g_1) = (00)$	0	0	0	0
$(f_1, g_1) = (01)$	$-\beta$	β	$-\beta$	β
$(f_1, g_1) = (10)$	α	α	$-\alpha$	$-\alpha$
$(f_1, g_1) = (11)$	$\alpha-\beta$	$\alpha+\beta$	$-\alpha-\beta$	$-\alpha+\beta$

Tabla 3.1. Estructura de una tabla LUT con fragmentos de 2 bits

La suma de las instanciaciones $\alpha f + \beta g$ extraídas de la tabla puede realizarse de forma secuencial. Por tanto, el módulo de cálculo consta de una tabla LUT y de un sumador.

En la implementación propuesta, para los valores de los argumentos, se ha optado por una representación en coma fija con la partición siguiente:

F y G tienen n bits, de los cuales:

Signo = 1 bit, parte entera $n/2$ bits, parte fraccionaria $n/2 - 1$ bits. El rango abarcado por esta representación es:

$$[-2^{-(\frac{n}{2}-1)}, -2^{\frac{n}{2}} + 2^{-(\frac{n}{2}-1)}] \cup [2^{-(\frac{n}{2}-1)}, 2^{\frac{n}{2}} - 2^{-(\frac{n}{2}-1)}]$$

Los valores de α y β que intervienen en los ejemplos presentados en esta investigación están comprendidos entre 0 y 2. De ahí que no haya bit de signo y que sólo haya 1 bit dedicado a la parte entera en la partición propuesta. En otros supuestos, la partición sufriría las modificaciones pertinentes.

Parte entera = 1 bit, parte fraccionaria = n'-1 bits

El rango abarcado es $[2^{-(n'-1)}, 2 - 2^{-(n'-1)}]$.

El tamaño de las instanciaciones de las primitivas almacenadas en la LUT es n''; éstas se representan en complemento a dos para evitar las restas en el momento de recomponer el resultado $\alpha F + \beta G$. Cabe precisar que, después de obtener el resultado, hay que pasarlo a una representación signo-magnitud necesaria para direccionar de nuevo la LUT.

Sin pérdida de generalidad se plantea $n = n' = n''$.

3 Evaluación de la arquitectura CBRM

En un dispositivo, los tiempos reales de cálculo y el espacio ocupado dependen de la tecnología utilizada en las implementaciones. No obstante, se puede lograr una primera aproximación de los tiempos de ejecución y del área ocupada independiente de la tecnología haciendo una estimación del circuito en términos de puertas lógicas. Así, en un circuito, el número de puertas utilizadas constituye una medida del área ocupada y el retardo de una puerta representa la unidad con la que se mide el tiempo de respuesta del circuito, entendido como camino crítico. En esta investigación, se considera como unidad de medida el conjunto formado por dos puertas lógicas (una XOR y una AND) porque este conjunto implementa una operación elemental que es la suma completa de un bit. τ_t, y τ_a son las unidades de tiempo y espacio, respectivamente, que corresponden a este dispositivo de cálculo elemental y que intervienen en la estimación de los bloques lógicos habituales. Una explicación más detallada de este modelo se facilita en [Wong, 1994], [Ercegovac, 2000] y [Piñeiro, 2002b].

Se presentan a continuación estimaciones de espacio y tiempo realizadas para bloques lógicos usuales, algunos de los cuales se encuentran en el prototipo propuesto.En estas estimaciones, además de τ_t, y τ_a, suelen intervenir otras características como el tamaño en bits y el número de entradas del bloque.

- Tiempos

Tablas LUT: se estima un retardo de 3 τ_t para una tabla con entrada de 7 bits, 3.5 τ_t para entradas de 8 bits, 4 τ_t para entradas de 9 bits y 4.5 τ_t para entradas de 10-11 bits, 5 τ_t para entradas con más de 11 bits.

Multiplexores: los multiplexores 2:1 y 3:1 tienen un retardo de 0,5 τ_t

Registros: tienen un retardo de $1\tau_t$

Registros de desplazamiento: hasta 4 desplazamientos $0{,}5\cdot\tau_t$; $1{,}5\cdot n\cdot\tau_t$ si el número de desplazamientos está comprendido entre 5 y 16, y $3\cdot\tau_t$ si está comprendido entre 17 y 64.

Estructura de reducción: para un contador 3:2 se asume un retardo de $2\tau_t$ y para un contador 4:2 el retardo es de $3\ \tau_t$

- Áreas

Tablas LUT: 40 τ_a/Kbit para tablas direccionadas por 6 bits como máximo y 35 τ_a/Kbit para direccionamientos entre 7 y 11 bits.

Multiplexores: $0{,}25\cdot k\cdot n\cdot\tau_a$ siendo k el número de vectores de entrada y n el tamaño de la palabra

Registro de n bits: $0{,}5\cdot n\cdot\tau_a$

Registros de desplazamiento: hasta 4 desplazamientos, $0{,}5\cdot n\cdot\tau_a$ siendo n el tamaño de la palabra; $2{,}5\cdot n\cdot\tau_a$ si el número de desplazamientos está comprendido entre 5 y 16, y $10{,}5\cdot n\cdot\tau_a$ si está comprendido entre 17 y 64.

Estructura de reducción: para un contador 3:2 se asume un retardo de $2\tau_a$ y para un contador 4:2, $4\ \tau_a$

3.1 Complejidad

Espacio ocupado

En la Tabla 3.2 se ha realizado una estimación del área que ocupa una LUT, expresada en función de τ_a para distintos valores de *t* y de *n*. El espacio varía con

el número de celdas de la LUT que es 2^{2t+2} y con el tamaño de los datos que encierra, que se tomará igual a n. El número de bits de direccionamiento es $2t+2$

n	t	LUT	
		Bytes	τ_a
16	**1**	32 B	10 τ_a
	2	128 B	40 τ_a
	4	2 KB	560 τ_a
	8	512 KB	102400 τ_a
32	**1**	64 B	20 τ_a
	2	256 B	80 τ_a
	4	4 KB	1120 τ_a
	8	1 MB	204800 τ_a
64	**1**	128B	40 τ_a
	2	512B	160 τ_a
	4	8KB	2240 τ_a
	8	2MB	409600 τ_a

Tabla 3.2 Estimación de la memoria ocupada por una LUT

En base a los datos reflejados en la Tabla 3.2 se establece una estimación del área total ocupada por el módulo de cálculo CBRM, expresada en términos de τ_a para distintos valores de n y t. Para ello se suman las áreas ocupadas por la LUT, por el sumador y por el registro de desplazamiento (Tabla 3.3).

	$n = 16$	$n = 32$	$n = 64$
t=1	$10\,\tau_a+16\cdot1\tau_a+8\tau_a$ $= 34\,\tau_a$	$20\,\tau_a+32\cdot1\cdot\tau_a+16\tau_a$ $= 68\,\tau_a$	$40\,\tau_a+64\cdot1\cdot\tau_a+32\tau_a$ $= 136\,\tau_a$
t=2	$40\,\tau_a + 8\cdot2\cdot\tau_a +8\tau_a$ $= 64\,\tau_a$	$80\,\tau_a+16\cdot2\cdot\tau_a+16\tau_a$ $= 128\,\tau_a$	$160\,\tau_a+32\cdot2\cdot\tau_a +32\tau_a$ $= 256\,\tau_a$
t=4	$560\,\tau_a + 4\cdot4\tau_a+8\tau_a$ $= 584\,\tau_a$	$1120\,\tau_a + 8\cdot4\tau_a+16\tau_a$ $= 1168\,\tau_a$	$2240\,\tau_a + 16\cdot4\tau_a+32\tau_a$ $= 2336\,\tau_a$
t=8	$102400\,\tau_a + 2\cdot8\cdot\tau_a +8\tau_a$ $= 102424\,\tau_a$	$204800\,\tau_a + 4\cdot8\cdot\tau_a +16\tau_a$ $= 204848\,\tau_a$	$409600\,\tau_a + 8\cdot8\cdot\tau_a+32\tau_a$ $= 409696\,\tau_a$

Tabla 3.3 Estimación del área total ocupada por el módulo de cálculo del CBRM

Se observa que el área total ocupada crece exponencialmente con t y linealmente con n. El coste de área viene motivado esencialmente por el tipo de crecimiento exponencial de la LUT. A efectos de ocupación de área, interesa que los fragmentos sean lo más pequeños posible, cualquiera que sea el valor de n.

Tiempo de cálculo

La Tabla 3.4 muestra una estimación del tiempo de cálculo expresado en función de τ_t para distintos valores de n y t. El tiempo de cálculo es la suma del tiempo de acceso a la LUT y del tiempo invertido por la suma secuencial.

	$n = 16$	$n = 32$	$n = 64$
$t=1$	$3\tau_t+16\tau_t \lg 16+7{,}5\tau_t = 74{,}5\ \tau_t$	$3\tau_t+32\tau_t \lg 32+15{,}5\tau_t = 178{,}5\ \tau_t$	$3\tau_t+64\tau_t \lg 64+31{,}5\tau_t = 418{,}5\ \tau_t$
$t=2$	$3\tau_t+8\tau_t \lg 8+7{,}5\tau_t = 34{,}5\ \tau_t$	$3\tau_t+16\tau_t \lg 16+15{,}5\tau_t = 82{,}5\ \tau_t$	$3\tau_t+32\tau_t \lg 32+31{,}5\tau_t = 194{,}5\ \tau_t$
$t=4$	$4.5\tau_t+4\tau_t \lg 4+7{,}5\tau_t = 20\tau_t$	$4.5\tau_t+8\tau_t \lg 8+15{,}5\tau_t = 44\ \tau_t$	$4.5\tau_t+16\tau_t \lg 16+31{,}5\tau_t = 100\ \tau_t$
$t=8$	$5\tau_t+2\tau_t \lg 2+7{,}5\tau_t = 14{,}5\ \tau_t$	$5\tau_t+4\tau_t \lg 4+15{,}5\tau_t = 28{,}5\ \tau_t$	$5\tau_t+8\tau_t \lg 8+31{,}5\tau_t = 60{,}5\ \tau_t$

Tabla 3.4 Estimación del tiempo de cálculo del CBRM

Se observa una disminución clara del tiempo de cálculo cuando crece el tamaño t de los fragmentos. La disminución es más acusada cuanto más altos son los valores de n. Ello se debe a la importancia relativa del segundo término de la suma, correspondiente al tiempo del sumador, que es $n/t \cdot \log_2 n/t$ y que decrece marcadamente cuando t aumenta. Los tiempos de acceso a la LUT son poco relevantes; no dependen de n, sólo de t pero son poco variables con éste.

Resumiendo: las estimaciones del área ocupada y del tiempo de cálculo invertido ponen de manifiesto que el tamaño creciente de los fragmentos

aumenta considerablemente el área ocupada para cualquier valor de n, pero también disminuye de forma dramática el tiempo de cálculo. La fragmentación de los operandos es, pues, un factor determinante en las prestaciones del módulo de cálculo a tener en cuenta en el momento de proponer mejoras.

3.2 Simulación del módulo de cálculo CBRM

Se ha sintetizado el circuito relativo al módulo de cálculo del prototipo CBRM en la plataforma xcv300e-6bg352-XST de Xilinx, con el propósito de verificar el funcionamiento del método.

El módulo de cálculo descrito en VHDL consta de dos subcircuitos principales, MEM y MAC cuyo cometido es, respectivamente, seleccionar las primitivas parciales y sumarlas secuencialmente, siguiendo la Implementación1. A su vez, el módulo MEM se desglosa en otros tres, Tlu, Puntos y Counter. Tlu representa la LUT en la cual se almacenan las primitivas que actúan como productos parciales de una multiplicación, Puntos selecciona los cuatro bits de direccionamiento extraídos de las entradas F y G y Counter marca el paso de extracción. El módulo MAC, que realiza la suma secuencial y los desplazamientos de los productos parciales, se compone de dos módulos embebidos el uno en el otro: Prueba y Elemento recursivo. Prueba es un sumador de 1 bit, Elemento recursivo es un sumador de n bits construido por repetición de Prueba. Para terminar, MAC repite n veces la estructura de Elemento recursivo. Sincronizado con el reloj del sistema, el contador Counter marca los accesos a LUT de 0 a 15. Cada acceso es recogido por el módulo Puntos para proporcionar los cuatro bits y que direccionan la LUT. A su salida cada producto parcial extraído va al MAC donde es sumado y desplazado, hasta completar las 15 sumas de que consta una iteración. El resultado obtenido, que está en complemento a dos, deberá ser convertido a representación signo

magnitud antes de realimentarse en el módulo Puntos para iniciar la siguiente iteración, después de poner el contador Counter a cero

En las figuras 3.4 a-l, se presenta un esquema del circuito de cálculo CBRM desglosado por módulos. Para cada módulo se muestran el símbolo y su instanciación.

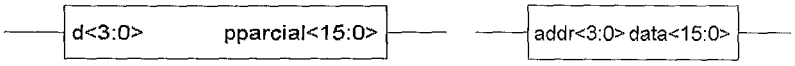

Fig. 3.4a Módulo Tlu representado simbólicamente e instanciado por una ROM

Fig. 3.4b Módulo Puntos representado simbólicamente

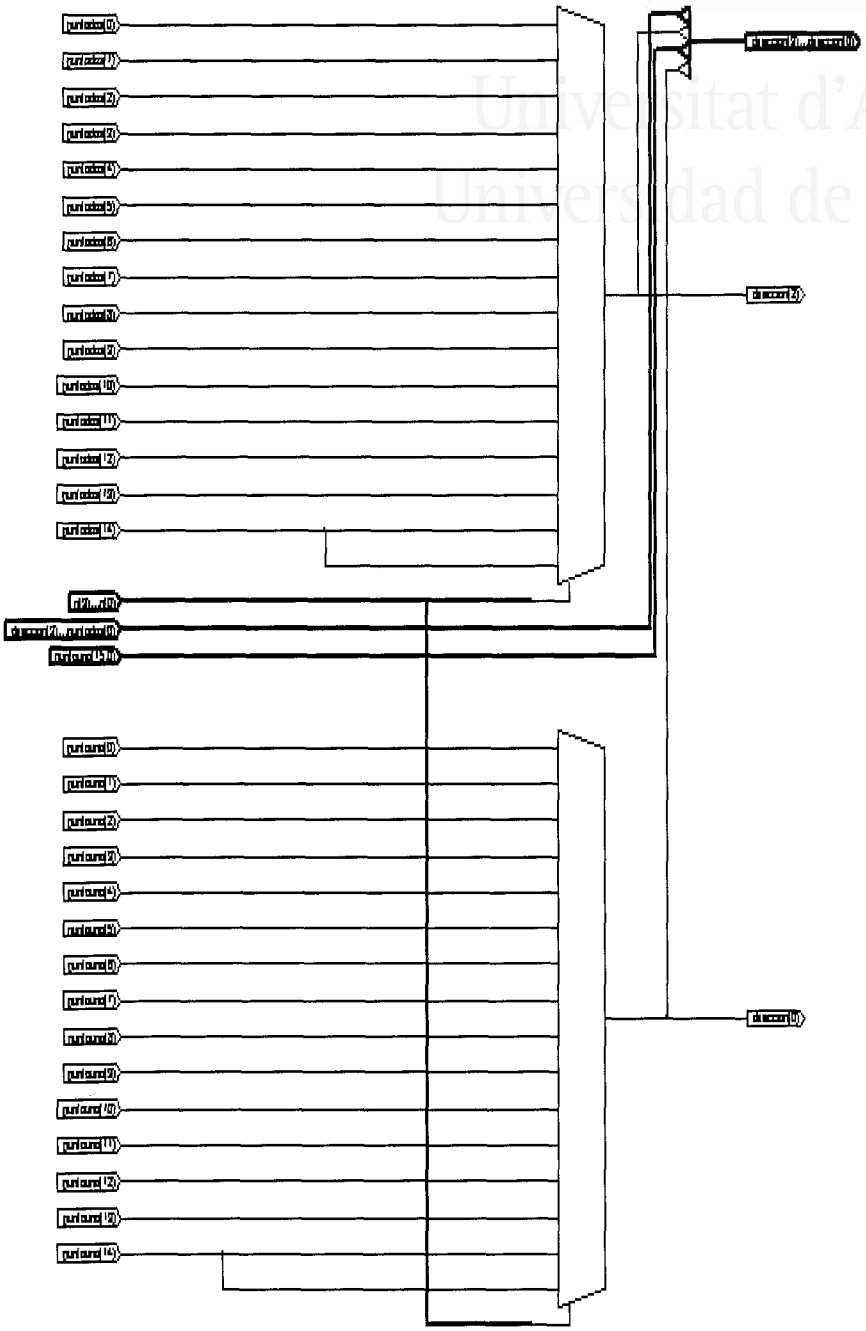

Fig.3.4c Instanciación del módulo Puntos por dos multiplexores

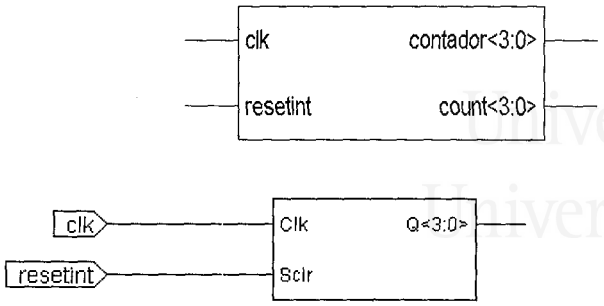

Fig.3.4d Módulo Counter representado simbólicamente e instanciado por un contador

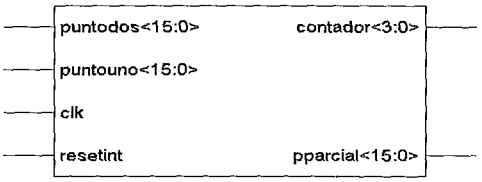

Fig.3.4e Módulo MEM representado simbólicamente

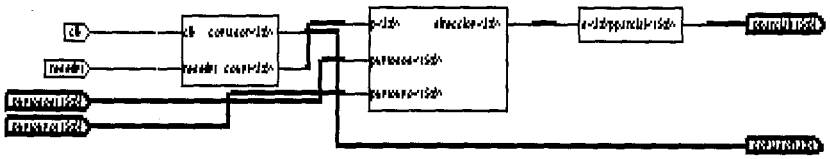

Fig.3.4f Estructura del módulo MEM

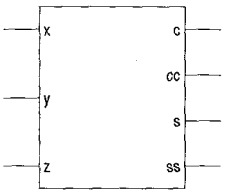

Fig.3.4g Módulo Prueba representado simbólicamente

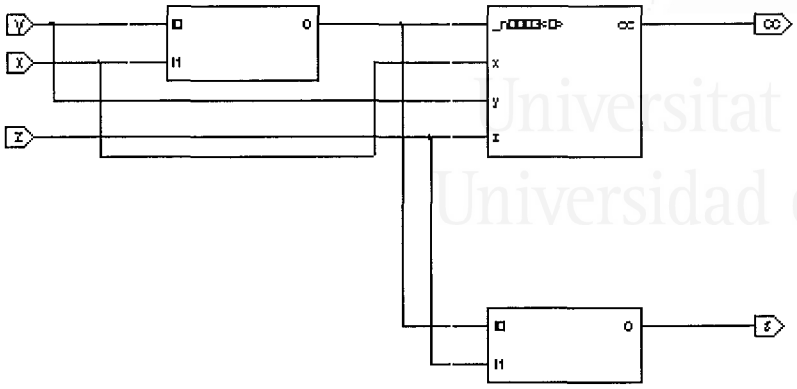

Fig.3.4h Módulo Prueba instanciado por puertas XOR

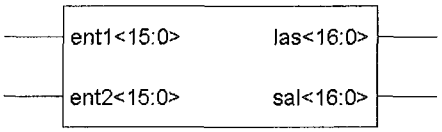

Fig.3.4i Módulo Elemento recursivo representado simbólicamente

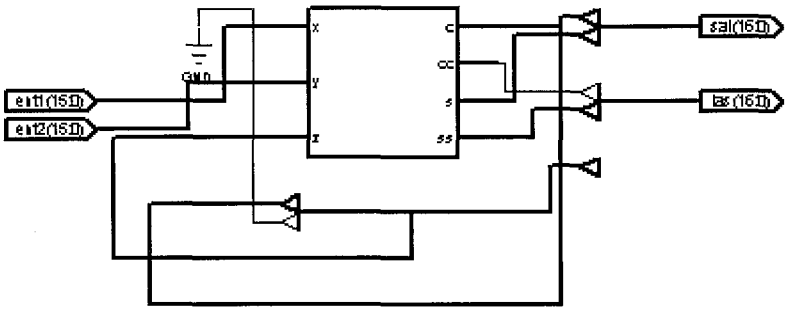

Fig.3.4j Módulo Elemento recursivo instanciado por puertas XOR

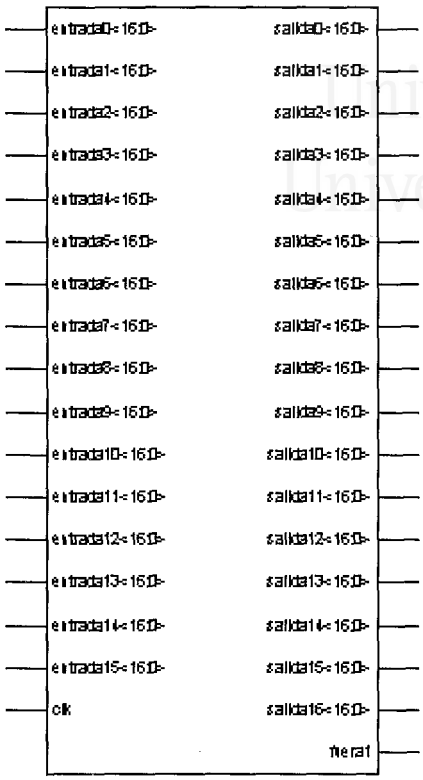

Fig.3.4k Módulo MAC representado simbólicamente

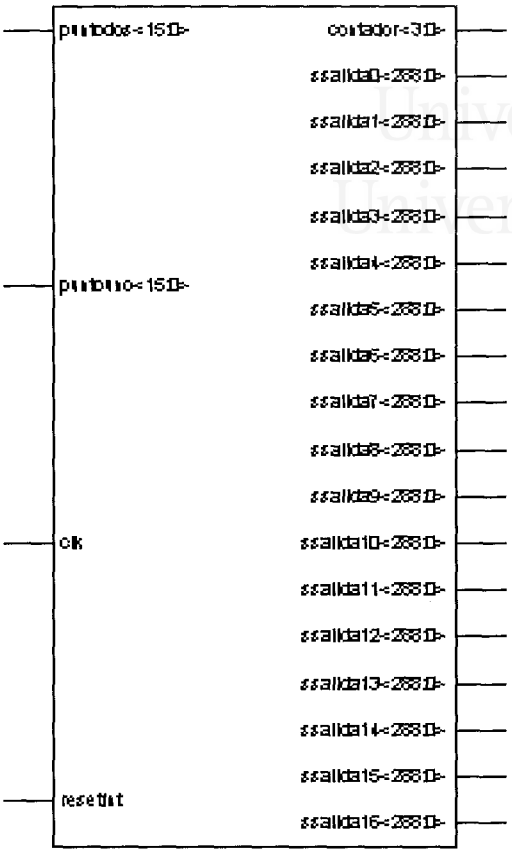

Fig.3.4l Módulo de cálculo CBRM representado simbólicamente

Las estimaciones de tiempo se han realizado con datos de precisión igual a 16 bits, arrojando los resultados que aparecen en la Tabla 3.5. Cada estimación se realiza midiendo el tiempo entre que la señal entra al módulo por un pin de entrada hasta que sale por un pin de salida; por consiguiente, ninguna estimación tiene porqué ser única. El tiempo consta de la suma de dos contribuciones, la de la lógica combinacional involucrada más la de la lógica de encaminamiento, que figuran en este orden en la tabla. El contador tiene un tiempo de offset que influye en los módulos MEM y CBRM y que se ha tenido en cuenta en los resultados.

Módulo MEM		Módulo MAC	
Counter	6,778 ns = 5,594 ns + 1,184 ns	Prueba	8,011 ns = 5,867 ns + 2,144 ns
Puntos	10,897 ns = 6,977 ns + 3,920 ns	El.recursivo	29,683 ns = 12,419 ns + 17, 264 ns
Tlu	8,939 ns = 5,867 ns + 3,072 ns		
Total MEM = 26,229 ns = 15,085 ns + 11,144 ns		Total MAC = 77,827 ns = 26,459 ns + 51,368 ns	
Total CBRM –parcial = 51,951 ns = 18,655 ns + 33, 296 ns			
Total CBRM = 104,056 ns = 26,229 ns + 77,827 ns			

Tabla 3.5 Estimaciones de tiempo del módulo de cálculo CBRM implementado en la FPGA xcv300e-6bg352-XST de Xilinx.

Los resultados que aparecen en la Tabla 3.5 ponen de manifiesto que:

- El tiempo total empleado por el módulo MEM depende linealmente del tiempo correspondiente a los módulos que lo integran. Así es también la dependencia de los tiempos que corresponden a la lógica combinacional y al encaminamiento.
- El tiempo empleado por la lógica combinacional del módulo MAC crece linealmente desde el nivel más interior, Prueba, hasta el más exterior MAC. El tiempo de encaminamiento crece en el mismo sentido, pero con un crecimiento rápido primero que tiende a moderarse.
- Generalmente, se observa que el tiempo invertido por el encaminamiento es inferior al que necesita la lógica combinacional, salvo que se trate de estructuras embebidas.
- El tiempo total de cálculo del CBRM-parcial que aparece en la Tabla 3.5 debe interpretarse como la suma de MEM más El. Recursivo. Ello se debe a que el dato extraído de la memoria tiene 16 bits obligando a una suma de 16 bits,

como mínimo. El tiempo de cálculo CBRM representa el tiempo necesario para calcular un punto de la función, que se obtiene sumando los tiempos MEM y MAC.

- La contribución del módulo MAC influye de forma notable en el tiempo total. La implementación 1 del CBRM no es óptima en cuanto al tiempo.

4 Mejora de la arquitectura CBRM

La arquitectura CBRM es paralelizable en algunas de sus etapas, sugiriendo mejoras en sus prestaciones. Se consideran en este apartado la incorporación de distintos grados de paralelismo. Primero, se plantea la paralelización de la suma en el módulo de cálculo. En un segundo tiempo, aprovechando la partición inicial realizada por el tipo de representación de los datos elegido, se propone la paralelización de la ejecución del cálculo de un punto. Para finalizar, se considera la paralelización del cálculo de N puntos de una función.

4.1 Incorporación de paralelismo en la suma, en el módulo de cálculo

En el módulo de cálculo descrito, la suma de los productos parciales se realiza secuencialmente, como aparece en la Tabla 3.5. Esta operación de n sumas de n bits tiene una complejidad temporal proporcional a $O(n^2)$. La realización de una suma con estructura de reducción rebaja la complejidad a $O(n\log_2 n)$ con el consiguiente aumento de hardware. En este sentido existen abundantes referencias en la literatura que proponen diversos algoritmos que utilizan circuitos contadores [Wallace, 1964], [Dadda, 1965] o compresores [Weinberger, 1981], [Song y de Michelli, 1991], [Known et al., 2000] basados en elementos de suma sin acarreo

[Bewick y Flynn, 1992], [Omondi, 1994], [Oklobdzija et al, 1996], [Choi et al., 1997]. Estos elementos son conectados entre sí formando una topología determinada, por ejemplo en árbol [Takagi et al, 1985]. Si los sumandos están divididos en fragmentos de tamaño mayor que 1 bit, cabe también la posibilidad de realizar la suma por bloques [Mora, 2001].

Las Tablas 3.6 a, b y c muestran la diferencia entre el área ocupada por implementaciones que emplean un sumador que opera secuencialmente o un sumador con reductores. En esta última implementación no intervienen los registros de desplazamiento.

n = 16	*Implementación con sumador secuencial*	*Implementación con reductores 4:2*
t=1	$10\ \tau_a+16\cdot1\tau_a+8\tau_a = 34\ \tau_a$	$10\ \tau_a+7\ \tau_a+16\cdot1\tau_a = 33\ \tau_a$
t=2	$40\ \tau_a+8\cdot2\cdot\tau_a+8\tau_a = 64\ \tau_a$	$40\ \tau_a+7\ \tau_a+8\cdot2\tau_a = 63\ \tau_a$
t=4	$560\ \tau_a+4\cdot4\tau_a+8\tau_a = 584\ \tau_a$	$560\ \tau_a+7\ \tau_a+4\cdot4\tau_a = 583\ \tau_a$
t=8	$102400\ \tau_a+2\cdot8\cdot\tau_a+8\tau_a = 102424\ \tau_a$	$102400\ \tau_a+7\ \tau_a+2\cdot8\tau_a = 102423\ \tau_a$

Tabla 3.6a Estimación del área total ocupada por las implementaciones con sumador secuencial y con reductores para $n = 16$ bits

n = 32	*Implementación con sumador secuencial*	*Implementación con reductores 4:2*
t=1	$20\ \tau_a + 32 \cdot 1 \cdot \tau_a + 16\tau_a = 68\ \tau_a$	$20\ \tau_a + 15\ \tau_a + 32 \cdot 1\tau_a = 67\ \tau_a$
t=2	$80\ \tau_a + 16 \cdot 2 \cdot \tau_a + 16\tau_a = 128\ \tau_a$	$80\ \tau_a + 15\ \tau_a + 16 \cdot 2\tau_a = 127\ \tau_a$
t=4	$1120\ \tau_a + 8 \cdot 4\tau_a + 16\tau_a = 1168\ \tau_a$	$1120\ \tau_a + 15\ \tau_a + 8 \cdot 4\tau_a = 1167\ \tau_a$
t=8	$204800\ \tau_a + 4 \cdot 8 \cdot \tau_a + 16\tau_a = 204848\ \tau_a$	$204800\ \tau_a + 15\ \tau_a + 4 \cdot 8\tau_a = 204847\ \tau_a$

Tabla 3.6b Estimación del área total ocupada por las implementaciones con sumador secuencial y con reductores para $n = 32$ bits

n = 64	*Implementación con sumador secuencial*	*Implementación con reductores 4:2*
t=1	$40\ \tau_a + 64 \cdot 1 \cdot \tau_a + 32\tau_a = 136\ \tau_a$	$40\ \tau_a + 31\ \tau_a + 64 \cdot 1\tau_a = 135\ \tau_a$
t=2	$160\ \tau_a + 32 \cdot 2 \cdot \tau_a + 32\tau_a = 256\ \tau_a$	$160\ \tau_a + 31\ \tau_a + 32 \cdot 2\tau_a = 255\ \tau_a$
t=4	$2240\ \tau_a + 16 \cdot 4\tau_a + 32\tau_a = 2336\ \tau_a$	$2240\ \tau_a + 31\ \tau_a + 16 \cdot 4\tau_a = 2335\ \tau_a$
t=8	$409600\ \tau_a + 8 \cdot 8 \cdot \tau_a + 32\tau_a = 409696\ \tau_a$	$409600\ \tau_a + 31\ \tau_a + 8 \cdot 8\tau_a = 409695\ \tau_a$

Tabla 3.6c Estimación del área total ocupada por las implementaciones con sumador secuencial y con reductores para $n = 64$ bits

Estas tablas ponen de manifiesto que la implementación con reductores presenta una ocupación de área equivalente a la de la implementación con suma secuencial.

Las Tablas 3.7 a, b y c muestran la diferencia entre los tiempos de cálculo en implementaciones que emplean un sumador que opera secuencialmente o un sumador con elementos de reducción.

n =16	*Implementación con sumador secuencial*	*Implementación con reductores 4:2*
t = 1	$3\tau_t+16\tau_t \lg 16+7{,}5\tau_t = 74{,}5\ \tau_t$	$3\tau_t+3.3\tau_t+\tau_t \lg 16 = 16\ \tau_t$
t = 2	$3\tau_t+8\tau_t \lg 8+7{,}5\tau_t = 34{,}5\ \tau_t$	$3\tau_t+3.3\tau_t+\tau_t \lg 8 = 15\ \tau_t$
t = 4	$4.5\tau_t+4\tau_t \lg 4+7{,}5\tau_t = 20\tau_t$	$4.5\tau_t+.3\ 3\tau_t+\tau_t \lg 4 = 15{,}5\ \tau_t$
t = 8	$5\tau_t+2\tau_t \lg 2+7{,}5\tau_t = 14{,}5\ \tau_t$	$5\tau_t+3.3\tau_t+\tau_t \lg 2 = 15\ \tau_t$

Tabla 3.7a Estimación del tiempo de cálculo en las implementaciones con sumador secuencial y con reductor para n = 16 bits

n =32	*Implementación con sumador secuencial*	*Implementación conreductor 4:2*
t =1	$3\tau_t+32\tau_t \lg 32+15{,}5\tau_t = 178{,}5\ \tau_t$	$3\tau_t+3.4\ \tau_t+\tau_t \lg 32 = 20\ \tau_t$
t = 2	$3\tau_t+16\tau_t \lg 16+15{,}5\tau_t = 82{,}5\ \tau_t$	$3\tau_t+3.4\ \tau_t+\tau_t \lg 16 = 19\ \tau_t$
t = 4	$4.5\tau_t+8\tau_t \lg 8+15{,}5\tau_t = 44\ \tau_t$	$4.5\tau_t+3.4\ \tau_t+\tau_t \lg 8 = 19{,}5\ \tau_t$
t = 8	$5\tau_t+4\tau_t \lg 4+15{,}5\tau_t = 28{,}5\ \tau_t$	$5\tau_t+3.4\ \tau_t+\tau_t \lg 4 = 19\ \tau_t$

Tabla 3.7b Estimación del tiempo de cálculo en las implementaciones con sumador secuencial y con reductor para n = 32 bits

n =64	*Implementación con sumador secuencial*	*Implementación con reductor 4:2*
t =1	$3\tau_t+64\tau_t \lg 64+31{,}5\tau_t = 418{,}5\ \tau_t$	$3\tau_t+3.5\tau_t+\tau_t \lg 64 = 24\ \tau_t$
t = 2	$3\tau_t+32\tau_t \lg 32+31{,}5\tau_t = 194{,}5\ \tau_t$	$3\tau_t+3.5\ \tau_t+\tau_t \lg 32 = 23\ \tau_t$
t = 4	$4.5\tau_t+16\tau_t \lg 16+31{,}5\tau_t = 100\ \tau_t$	$4.5\tau_t+3.5\ \tau_t+\tau_t \lg 16 = 23{,}5\ \tau_t$
t = 8	$5\tau_t+8\tau_t \lg 8+31{,}5\tau_t = 60{,}5\ \tau_t$	$5\tau_t+3.5\ \tau_t+\tau_t \lg 8 = 23\ \tau_t$

Tabla 3.7c Estimación del tiempo de cálculo en las implementaciones con sumador secuencial y con reductor para n = 64 bits

Estas tablas ponen de manifiesto que la implementación que utiliza reductores invierte un tiempo de cálculo prácticamente constante para cualquier tamaño de los fragmentos de un operando. Además, el tiempo de cálculo crece bastante poco con el tamaño del operando. Cabe observar que los tiempos en la implementación con reductores son mucho mejores que los de la implementación con sumador secuencial, sobre todo a medida que el tamaño de los fragmentos disminuye y a medida que aumenta el tamaño de los operandos.

4.2 Incorporación de paralelismo en el cálculo de un punto

La partición inicial de los operandos puede utilizarse para incorporar paralelización a nivel de la ejecución del cálculo:

a) obteniendo el resultado del producto que corresponde a la parte entera de F por la parte fraccionaria de α más la parte entera de G por la parte fraccionaria de β (mediante extracciones de la LUT)

b) sumándole al resultado anterior F (si la parte entera de α es igual a 1), G (si la parte entera de β es igual a 1), $F+G$ (si las partes enteras de α y β son iguales a 1)

c) obteniendo el resultado del producto que corresponde a la parte fraccionaria de F por la parte fraccionaria de α más la parte fraccionaria de G por la parte fraccionaria de β (extracciones de la LUT)

Este planteamiento permite mayor flexibilidad, permitiendo tomar en consideración la parte a) solamente, para aplicaciones en las que pueda asumirse el error debido a la eliminación de la parte fraccionaria de F y G. Puede también paralelizarse el cálculo de las etapas a) y c). Por otra parte, la etapa b) no

interviene cuando α y β están comprendidos entre cero y uno, por tanto sólo se tomará en consideración cuando sea necesario.

Las mejoras en área ocupada y en tiempo de cálculo dependen del supuesto en el que se trabaja. En el supuesto a), la mejora puede cuantificarse a partir de lo establecido por el estudio anterior relativo a datos no particionados, con la corrección del número de bits que corresponda. Como ejemplo, se trata el caso del cálculo paralelo de las partes a) y c), estudiando también la influencia del tipo de partición de los operandos.

Cálculo paralelo con operandos particionados

Se sustituye un operando de *n* bits por dos operandos de $n/2$ bits y $n/2 - 1$ bits. El primero representa la parte entera y el segundo la parte fraccionaria. La implementación utiliza una tabla LUT compartida y duplica el sumador y los elementos de reducción que corresponden. Los resultados se muestran en las Tablas 3.8 a-b.

	Implementación con reductores 4:2 $n = 32$ *particionado en* 16 *y* 15	*Implementación con reductores 4:2* $n = 64$ *particionado en* 32 *y* 31
t=1	$10\ \tau_a + 2(7\ \tau_a + 16 \cdot 1\tau_a) = 56\ \tau_a$	$20\ \tau_a + 2(15\ \tau_a + 32 \cdot 1\tau_a) = 114\ \tau_a$
t=2	$40\ \tau_a + 2(7\ \tau_a + 8 \cdot 2\tau_a) = 86\ \tau_a$	$80\ \tau_a + 2(15\ \tau_a + 16 \cdot 2\tau_a) = 174\ \tau_a$
t=4	$560\ \tau_a + 2(7\ \tau_a + 4 \cdot 4\tau_a) = 606\ \tau_a$	$1120\ \tau_a + 2(15\ \tau_a + 8 \cdot 4\tau_a) = 1214\ \tau_a$
t=8	$102400\ \tau_a + 2(7\ \tau_a + 2 \cdot 8\tau_a) = 102446\ \tau_a$	$204800\ \tau_a + 2(15\ \tau_a + 4 \cdot 8\tau_a) = 204894\ \tau_a$

Tabla 3.8a Estimación del área ocupada por la implementación con datos particionados, para $n = 32$ y $n = 64$

	Implementación con reductores 4:2 n =32 *particionado en* 16 *y* 15	*Implementación conreductor 4:2* n =64 *particionado en* 32 *y* 31
$t = 1$	$3\tau_t + 3.3\tau_t + \tau_t \lg 16 = 16\ \tau_t$	$3\tau_t + 3.4\ \tau_t + \tau_t \lg 32 = 20\ \tau_t$
$t = 2$	$3\tau_t + 3.3\tau_t + \tau_t \lg 8 = 15\ \tau_t$	$3\tau_t + 3.4\ \tau_t + \tau_t \lg 16 = 19\ \tau_t$
$t = 4$	$4.5\tau_t + .3\ 3\tau_t + \tau_t \lg 4 = 15{,}5\ \tau_t$	$4.5\tau_t + 3.4\ \tau_t + \tau_t \lg 8 = 19{,}5\ \tau_t$
$t = 8$	$5\tau_t + 3.3\tau_t + \tau_t \lg 2 = 15\ \tau_t$	$5\tau_t + 3.4\ \tau_t + \tau_t \lg 4 = 19\ \tau_t$

Tabla 3.8b Estimación del tiempo de cálculo la implementación con datos particionados, para $n = 32$ y $n = 64$

Comparando la Tabla 3.8 a con las estimaciones homólogas de la Tablas 3.6b-c, se pone de manifiesto una disminución apreciable del área ocupada en la implementación que particiona los operandos; la disminución es tanto mayor cuanto mayor es el tamaño del fragmento. La comparación de la Tabla 3.7b con las Tablas 3.8b-c muestra también una disminución de tiempo, pero no tan apreciable ya que, como se ha mencionado anteriormente, la implementación con elementos de reducción no presenta variaciones de tiempo espectaculares cuando el tamaño del operando varía, ni tampoco cuando varía el tamaño del fragmento.

La partición propuesta para los operandos es aproximadamente de mitad y mitad para las partes entera y fraccionaria del operando. Si una de las partes es de tamaño mayor que la otra, no se prevén cambios en el área ocupada, ya que ésta se calcula como una suma de contribuciones de dos dispositivos, independiente de su distribución. El tiempo de cálculo experimenta un ligero aumento, debido a la parte del operando que presenta mayor número de bits, pero, tal y como ya se ha observado, las variaciones del tamaño del operando tienen poca repercusión en los tiempos.

4.3 Incorporación de paralelismo en el cálculo de N puntos de una función

El cálculo de N puntos de una función puede efectuarse a través del mismo módulo de cálculo, adoptando la partición de datos que se estime oportuna. Si T es el tiempo de cálculo de un punto, la utilización de un mismo módulo para el cálculo de los N puntos tiene como efecto la multiplicación por un factor N del tiempo T. Para paliar este aumento lineal del tiempo se pueden proponer arquitecturas que, aunque aumenten un poco el área ocupada, son beneficiosas para la disminución del tiempo de cálculo, de manera que permiten establecer un compromiso según los requerimientos del problema a resolver.

El problema es el siguiente: dada una función, calcular N puntos en el intervalo [a, b]. La introducción de paralelismo a este caso consiste en establecer distintas etapas que representan la profundidad de un árbol. Cada etapa de cálculo divide los subintervalos de [a, b] definidos en la etapa anterior en subintervalos más pequeños. Por tanto, al inicio de cada etapa, el número de módulos que han de actuar es igual al número de subintervalos definidos en la etapa anterior. En cada etapa el cálculo de los módulos se realiza en paralelo.

Si $N = N_1 \times N_2 \times .... \times N_p$

1ª etapa: 1 mó dulo calcula N_1 puntos

2ª etapa: N_1 módulos calculan $N_1 \times N_2$ puntos (N_2 puntos por módulo)

.....

p-ésima etapa: $N_1 x N_2 x .... x N_{p-1}$ módulos calculan $N = N_1 x N_2 x .. x N_p$ puntos (N_p puntos por módulo)

En la Tabla 3.9 se presentan los resultados en cuanto a tiempo de cálculo y número de módulos necesarios para el cálculo secuencial realizado por un solo módulo y el cálculo paralelizado.

	Tiempo de cálculo de N puntos	*Número de módulos necesarios*
Cálculo secuencial	$TN = TN_1N_2$ .....N_p	1
Cálculo paralelizado	$=TN_1+TN_2+.....+ TN_p$ $= T (N_1+N_2+.....+ N_p)$	$1+ N_1 + N_1N_2+ N_1 N_2N_3.+...+.N_1 N_2N_{3...} N_{p-1}$

Tabla 3.9 Estimación del tiempo de cálculo y del número de módulos en el cálculo secuencial y paralelo

Se observa que en el cálculo paralelizado, el tiempo depende de la descomposición N_1, N_2,....., N_p que se ha hecho de N. El valor de la suma $N_1+N_2+.....+ N_p$ disminuye cuando aumenta p, que es el número de factores en los que se descompone N. Por tanto, dado N, para mejorar el tiempo, hay que realizar el mayor número posible de etapas. Por el contrario, el número de módulos necesarios decrece cuando disminuye p, es decir, cuando disminuye el número de etapas. Además, si se ordenan los números N_i de forma que el mayor sea N_p, disminuye también el número de módulos, dado que N_p, no interviene en la expresión.

Comparado con los resultados proporcionados por un módulo único, el cálculo paralelizado presenta la ganancia de velocidad, la productividad y la eficiencia siguientes:

- ganancia en velocidad:

$$G = \frac{TN}{T(N_1 + N_2 .... + N_p)} = \frac{\prod_1^p N_i}{\sum_1^p N_i}$$

- productividad:

$$P = \frac{N}{1 + N_1 + N_1 N_2 + ... + N_1 N_2 ..... N_{p-1}}$$

- eficiencia:

En la primera etapa:

$$E_1 = \frac{TN_1}{T(N_1 + N_2 + ...N_p)} = \frac{N_1}{\sum_1^p N_i}$$

En la 2ª etapa:

$$E_2 = \frac{TN_2}{T(N_1 + N_2 + ...N_p)} = \frac{N_2}{\sum_1^p N_i}$$

En la p-ésima etapa

$$E_{p-1} = \frac{TN_{p-1}}{T(N_1 + N_2 + ...N_p)} = \frac{N_{p-1}}{\sum_1^p N_i}$$

La eficiencia media es:

$$E = \frac{\sum_{1}^{p-1} E_i}{p-1}$$

La Tabla 3.10 recoge las medidas de ganancia en velocidad, productividad y eficiencia del cálculo realizado por un módulo frente al cálculo paralelizado.

	Ganancia en velocidad	*Productividad*	*Eficiencia*
Cálculo secuencial	1	N	100%
Cálculo paralelizado	$G = \frac{\prod_{1}^{p} N_i}{\sum_{1}^{p} N_i}$	$P = \frac{N}{1 + N_1 + N_1 N_2 + \ldots + N_1 N_2 \ldots N_{p-1}}$	$E = \frac{\sum_{1}^{p-1} E_i}{p-1}$

Tabla 3.10 Comparación de ganancia en velocidad, productividad y eficiencia entre el cálculo secuencial y paralelizado

En la Tabla 3.10 se observa que el cálculo de N puntos llevado a cabo por un solo módulo tiene unas medidas de productividad y eficiencia óptimas. Sin embargo, el cálculo paralelizado obtiene una ganancia en velocidad superior a la del cálculo secuencial, modulable por la descomposición de N. La productividad y eficiencia son, en cambio, inferiores.

5 Conclusión

El presente capítulo se ha dedicado al estudio de la arquitectura CBRM, con la propuesta y evaluación de un prototipo de procesador, del cual se han estudiado las características del módulo de cálculo. Se han realizado estimaciones del tiempo de cálculo y del área ocupada utilizando una medida de los recursos hardware independiente de tecnología, facilitando posteriormente medidas con referencia explícita a una plataforma determinada. Se han realizado igualmente algunas propuestas de mejora de prestaciones del módulo de cálculo con incorporación de paralelismo. Además de considerar las mejoras obtenidas por la paralelización de la suma, se ha planteado también el cálculo pralelo El cálculo de un gran número de puntos tiene una limitación inherente al CBRM que es la de proporcionar un punto por iteración. Se propone una arquitectura que consta de varios módulos interconectados, que incorpora etapas de cálculo paralelo. Esta propuesta presenta flexibilidad debido a la posibilidad de variar el número de etapas y el número de módulos por etapa. Se establecen las comparaciones en cuanto a ganancia en velocidad, productividad y eficiencia de los dos tipos de cálculo en función de estos parámetros.

Capítulo 4

CÁLCULO DE ROTACIONES MEDIANTE CBRM

1 Introducción

El presente capítulo y el siguiente están dedicados a las aplicaciones. Debido a su presencia en numerosos y muy diversos ámbitos, las funciones trigonométricas e hiperbólicas constituyen un caso de enorme interés aplicado. Así, es previsible que cualquier mejora en la operatoria de estas funciones repercuta favorablemente en el rendimiento global de los cálculos en los que dichas funciones intervienen. La rotación es una transformación geométrica, formalizable a partir de la combinación de las funciones trigonométricas seno y coseno, que sirve de base a algunas transformaciones muy utilizadas en tratamiento digital de la señal y de la imagen como la transformada de Hough [Bruguera, 1996], [García et al, 2003a], [García et al, 2003b], de Fourier [Frigo, 1998], [Frigo, 2000], [Randi, 2000] y la transformada del coseno [Sánchez, 1997], [Dick, 1998]. En estos casos, el CBRM presenta una idoneidad particular como método de cálculo debido a la posibilidad de evaluar simultáneamente las dos coordenadas características del punto que rota. El presente capítulo está dedicado a la clasificación de algunas de estas transformadas con el fin de

proponer un patrón común de cálculo para todas ellas, asumible por el CBRM. Posteriormente, se efectúa un análisis comparativo de los resultados de cálculo de las transformadas de Hough y Fourier entre el CBRM y otros métodos conocidos.

2 Rotaciones en el plano

Seguidamente se analizan las ecuaciones de la rotación bidimensional y se expresan con el formalismo del CBRM

2.1 Ecuaciones

En el plano definido por un par de ejes ortogonales, (Ox, Oy), sea $P(x_i, y_i)$ un punto de una circunferencia de radio R y θ_i el ángulo que forman OP y el eje Ox

$$x_i = R \cos \theta_i \qquad (4.1)$$
$$y_i = R \operatorname{sen} \theta_i$$

Sea $\Delta\theta$ el incremento de cada rotación:

$\theta_i = \theta_{i-1} + \Delta\theta, \ \forall i$

Así,

$$\cos \theta_i = \cos (\theta_{i-1} + \Delta\theta) \qquad (4.2)$$
$$\operatorname{sen} \theta_i = \operatorname{sen} (\theta_{i-1} + \Delta\theta)$$

planteando $\cos \Delta\theta = \alpha$, $\operatorname{sen} \Delta\theta = \beta$

desarrollando y sustituyendo (4.2) en (4.1) y reagrupando los términos, se obtiene:

$$x_i = \alpha x_{i-1} - \beta y_{i-1}$$
$$y_i = \alpha y_{i-1} + \beta x_{i-1} \qquad (4.3)$$
con la restricción adicional $\alpha^2 + \beta^2 = 1$

Las ecuaciones (4.3) expresan el valor de las coordenadas del punto rotado después de cada rotación, en función de las coordenadas del punto anterior.
Si se definen las coordenadas del punto P como dos funciones F y G, de forma que cada función represente la proyección del punto sobre uno de los ejes:

$$F(x_i, y_i) = x_i \qquad (4.4)$$
$$G(x_i, y_i) = y_i$$

Sustituyendo (4.4) en (4.3), resulta:

$$F(x_i y_i) = \alpha F(x_{i-1}, y_{i-1}) - \beta G(x_{i-1}, y_{i-1})$$
$$G(x_i y_i) = \alpha G(x_{i-1}, y_{i-1}) + \beta F(x_{i-1}, y_{i-1}) \qquad (4.5)$$
con $\alpha^2 + \beta^2 = 1$

La aplicación del CBRM no hace intervenir explícitamente las coordenadas; en su lugar se le asocia, como consecuencia de la regularización establecida, el número de la iteración en curso. El cálculo de F y G se expresa entonces como

$$F_i = \alpha F_{i-1} - \beta G_{i-1}$$
$$G_i = \alpha G_{i-1} + \beta F_{i-1} \qquad (4.6)$$
con $\alpha^2 + \beta^2 = 1$

Las ecuaciones (4.6) son evaluables a partir de las primitivas encontradas, por tanto, bajo el formato CBRM, la rotación bidimensional corresponde a la

evaluación cruzada de dos funciones, F y G, con una restricción sobre sus parámetros, $\alpha^2 + \beta^2 = 1$

2.2 Funciones configurables a partir de rotaciones

El movimiento de rotación es común a muchas transformaciones matemáticas; por ello, la evaluación de éstas por el CBRM se basa en el esquema propuesto en el apartado anterior.

Añadiendo a las ecuaciones (4.1) la coordenada z para pasar de una representación bidimensional a otra tridimensional, se describe el movimiento helicoidal. Haciendo $R = A$ en la primera ecuación y $R = B$ en la segunda, se describe el movimiento elipsoidal, siendo A y B los semi-ejes menor y mayor de la elipse. Asimismo, la transformada de Hough, utilizada en el proceso de segmentación de la imagen, transforma un punto del plano cartesiano $P(x,y)$ en una curva en el dominio parametrizado (ρ,θ) de Hough, mediante la ecuación $\rho = x \cdot cos\theta + y \cdot sen\theta$, para ángulos pertenecientes al intervalo $[0,\Pi[$. Las transformadas ortogonales pertenecen a una clase más general de transformaciones discretas para las que resulta menos inmediata la configuración como rotación y, por tanto, menos evidente la aplicabilidad del CBRM.

$$F(u) = \sum_{m=0}^{N-1} T(x,u) f(x) \qquad (4.7)$$

La ecuación (4.7) expresa una transformación ortogonal genérica en la cual $f(x)$, $0 \leq x <$ N y $F(u)$, $0 \leq u <$ N son respectivamente la secuencia original y

transformada, ambas de longitud $N = 2^n$. $T(x, u)$ es el núcleo de transformación. En general, la naturaleza de una transformación de este tipo está determinada por las propiedades de su núcleo. De manera similar se puede definir la transformación inversa. En la Tabla 4.1 se mencionan algunas transformaciones ortogonales importantes, facilitando la expresión del núcleo.

Transformada	**Núcleo $T(x,u)$**	**Observaciones**
Fourier	$\frac{1}{N}\exp(\frac{-2j\pi ux}{N})$	Núcleo trigonométrico
Hartley	$\cos(\frac{2\pi ux}{N}) + sen(\frac{2\pi ux}{N})$	Núcleo trigonométrico
Coseno	$e(k)\cos\frac{(2x+1)\pi u}{2N}$	Núcleo trigonométrico $e(0) = 1/\sqrt{2}$ $e(k) = 1 \quad 0 < k < N$
Walsh	$\frac{1}{N}\prod_{i=0}^{n-1}(-1)^{bi(x)bn-i-1(u)}$	Núcleo no trigonométrico b_i representa el i-ésimo bit del entero b
Hadamard	$\frac{1}{N}\prod_{i=0}^{n-1}(-1)^{bi(x)bi(u)}$	Núcleo no trigonométrico b_i representa el i-ésimo bit del entero b

Tabla 4.1 Algunas transformaciones ortogonales

El resto del capítulo se dedica a desarrollar la aplicación del CBRM al cálculo de algunas transformadas, realizando las modificaciones formales necesarias cuando la aplicabilidad del CBRM no puede deducirse de forma sencilla. En todos los casos se presentan otras propuestas de cálculo a fin de comparar los resultados.

3 Aplicación del CBRM a la transformada de Hough

En este apartado se realiza un análisis formal de la transformada de Hough para la cual se propone la aplicación del CBRM, como método idóneo de cálculo. También se han seleccionado dos ejemplos en los cuales se incluyen comparaciones entre los resultados proporcionados por el CBRM y por el CORDIC. La primera propuesta es una implementación segmentada reconfigurable del CORDIC [Deng, 2001]. La segunda considera una aplicación paralela del mismo algoritmo [Bruguera, 1996]. Estas implementaciones, que se discuten en [García et al, 2003b], comparan el CBRM y el CORDIC en cuanto a área ocupada y tiempo de cálculo.

3.1 Transformada de Hough

Propuesta en 1962, la transformada de Hough (HT) es una herramienta importante en la segmentación de imágenes [Hough, 59]. Detección de curvas [Muamar, 1991], reconocimiento de objetos [Haule, 1989], vectorización de fotografías aéreas [da Silva, 1990], reconstrucción de imágenes 3D [Yamazava, 2000], inspección de la calidad industrial [Bariani, 1997], aplicaciones biomédicas [Dong, 2001], [Tezmol, 2002], reconocimiento de quásares [Huang, 2000] y OCR [Sural, 2001], son algunos ejemplos de las múltiples utilidades de esta transformada. La HT es enormemente atractiva debido a su robustez aunque la gran cantidad de recursos espaciales y temporales que requiere la han mantenido alejada de las aplicaciones de tiempo real. Los esfuerzos realizados en investigación han ido en la línea de diseñar algoritmos rápidos o arquitecturas ad-hoc. Los distintos algoritmos, basados en el cálculo de funciones son: lineal [Koshimizu, 1990], combinatorio [Ben-Tzvi, 1990],

binario [da Fontura, 1989], adaptativo [Walther, 1971] y rápido [Li, 1992], cuyo inconveniente es la falta de regularidad y de posibilidades de paralelización, comparados con el algoritmo tradicional. Éste, en cambio, permite implementar arquitecturas con memoria compartida o distribuida (matriz lineal, mesh, hipercubo) [Shankar, 1990] y en árbol binario, así como arquitecturas sistólicas específicas [Chuang, 1995].

El proceso de detección de primitivas geométricas utilizando la HT puede descomponerse en tres etapas: la creación del contorno de la imagen utilizando un detector de bordes, la aplicación de la HT a cada punto de la imagen y la votación en el dominio de Hough por medio de la cual se realiza la extracción de las primitivas.

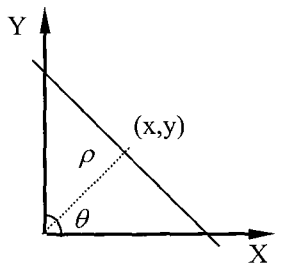

Fig 4.1. Parametrización de rectas para la HT.

Si la primitiva a detectar es la recta, desde el punto de vista geométrico, la HT transforma un punto del plano cartesiano $P(x,y)$ en una curva en el dominio (ρ,θ) de Hough (ver Fig.4.1). Recíprocamente, a un punto del dominio de Hough le corresponde una primitiva geométrica del plano. El dominio de Hough puede interpretarse como una rejilla de votación en la que cada punto $P(x,y)$ del plano "vota" por el conjunto de líneas que intersectan en él y que

son representadas por puntos en la rejilla (ρ,θ). Si disponemos de una imagen digitalizada de tamaño NxN y definimos el espacio parametrizado (ρ,θ), cada $P(x,y)$ se transformará en $\rho = x\cdot cos\theta + y\cdot sen\theta$. El espacio parametrizado se discretiza en N_θ niveles, entre 0 y Π y en N_ρ niveles, entre ρ_{min} y ρ_{max} y la transformada de Hough consiste en calcular los valores ρ para todos los ángulos del intervalo $[0,\Pi[$ y para cada píxel de la imagen digitalizada. El cálculo directo tiene una complejidad de $O(N^2)$ y el total de operaciones a realizar es de $N^2\cdot N_\theta$ Si se considera el intervalo $[0,\Pi[$ como la unión de dos subintervalos $[0,\Pi/2[\cup[\Pi/2,\Pi[$, para cada pixel (x_i,y_j) de la imagen, la transformada de Hough puede escribirse como:

$$\begin{aligned} \rho_I(k) &= x_i\cdot cos\theta_k + y_j\cdot sen\theta_k \qquad 0 \le \theta_k < \Pi/2 \\ \rho_{II}(k) &= y_j\cdot cos\theta_k - x_i\cdot sen\theta_k \qquad \Pi/2 \le \theta_k < \Pi \end{aligned} \tag{4.8}$$

Haciendo que

$$\begin{aligned} &\theta_k = \theta_{k-1} + \Delta\theta \\ &cos\,\theta_k = cos(\theta_{k-1} + \Delta\theta) \\ &sen\,\theta_k = sen(\theta_{k-1} + \Delta\theta) \\ &cos\,\Delta\theta = \alpha, sen\Delta\theta = \beta \end{aligned} \tag{4.9}$$

Al sustituir (4.9) en (4.8) tenemos que:

$$\begin{aligned} &\rho_I(k) = \alpha\cdot\rho_I(k-1) + \beta\cdot\rho_{II}(k-1) \\ &\rho_{II}(k) = \alpha\cdot\rho_{II}(k) + \beta\cdot\rho_I(k-1) \\ &con \quad \alpha^2 + \beta^2 = 1 \end{aligned} \tag{4.10}$$

Según aparece en (4.10), las funciones $(\rho_I)_k$ y $(\rho_{II})_k$ son evaluables por el CBRM, con la peculiaridad de que lafunción $G = (\rho_I)_k$ cuando se evalúa $(\rho_{II})_k$ y $G = (\rho_{II})_k$ cuando se evalúa $(\rho_I)_k$; inicializando $(\rho_I)_0$ y $(\rho_{II})_0$ con el valor de las coordenadas de cada píxel de la imagen.

3.2 Comparación CBRM - CORDIC segmentado

En este apartado se comparan los resultados de cálculo de la HT obtenidos por el CBRM y por una propuesta basada en el algoritmo CORDIC. En esta última [Deng, 2001], el cálculo se realiza con una representación de los datos de 16 bits en punto fijo y con un CORDIC de 12 iteraciones, implementado en una plataforma Xilinx XS4010XL-PC84 FPGA para prototipado rápido.
La tarjeta Xilinx XS4010XL-PC84 FPGA es un dispositivo de capacidad media que funciona con una velocidad moderada. Posee 400 CLBs arreglados en una matriz de 20 x 20, que equivale aproximadamente a 10000 puertas.

Para efectuar la comparación, se han tenido en cuenta por una parte, la evaluación de la arquitectura CBRM descrita en el capítulo 4 y por otra, las características de la tarjeta Xilinx utilizada, teniendo encuenta que un CLB (Bloque Lógico) consta de una LUT-3, dos LUT- 4 y dos cerrojos.

Comparación

La implementación CORDIC ocupa un 83% de área, esto es, 333 CLBs de los 400 que cuenta la tarjeta XS4010XL A una frecuencia de reloj de 40 MHz, con una complejidad computacional $O(N^2)$ para una imagen N x N, una imagen de 128x128 pixels con 128 ángulos discretos ($\Delta\theta$ = 1.40625°) tarda 0.0262 s en transformarse. La Tabla 4.2 presenta una estimación del área ocupada por el CORDIC y la Tabla 4.3 presenta la estimación del área ocupada por el CBRM.

Estas estimaciones se han hecho en función de τ_a. La precisión de los datos es de 16 bits. Se han considerado las dos implementaciones del CBRM, recordando que en la implementación 1 interviene un sumador secuencial y en la implementación 2 se sustituye éste por una estructura de reducción de 4:2 antes de realizar la suma final. En ambas implementaciones el área ocupada se duplica teniendo en cuenta que se efectúan dos evaluaciones cruzadas.

CORDIC Segmentado	***Nº CLBs =333***	*área ocupada*
LUT-3	$1 \cdot 333 = 333$	$333 \cdot 2^3 \cdot 2^4 \cdot 40\ \tau_a$ / Kbit $= 1665\tau_a$
LUT-4	$2 \cdot 333 = 666$	$2 \cdot 333 \cdot 2^4 \cdot 2^4 \cdot 40\ \tau_a$ / Kbit $= 6660\text{-}\tau_a$
Cerrojos	$2 \cdot 333 = 666$	$2 \cdot 333 \cdot 0.5 \cdot 2^4 \cdot\ \tau_a = 5328\text{-}\tau_a$
Total		$13653\ \tau_a$

Tabla 4.2 Estimación del área ocupada por la implementación CORDIC

CBRM	*Implementation 1*	*Implementation 2*
t=1	$2(10\ \tau_a + 16 \cdot 1\tau_a + 8\tau_a) = 68\ \tau_a$	$2(10\ \tau_a + 7\ \tau_a + 16 \cdot 1\tau_a) = 66\ \tau_a$
t=2	$2(40\ \tau_a + 8 \cdot 2 \cdot \tau_a + 8\tau_a) = 128\ \tau_a$	$2(40\ \tau_a + 7\ \tau_a + 8 \cdot 2\tau_a) = 126\ \tau_a$
t=4	$2(560\ \tau_a + 4 \cdot 4\tau_a + 8\tau_a) = 1168\ \tau_a$	$2(560\ \tau_a + 7\ \tau_a + 4 \cdot 4\tau_a) = 1166\ \tau_a$
t=8	$(102400\ \tau_a + 2 \cdot 8 \cdot \tau_a + 8\tau_a) = 204848\ \tau_a$	$2(102400\ \tau_a + 7\ \tau_a + 2 \cdot 8\tau_a = 204846\tau_a$

Tabla 4.3 Estimación del área ocupada por la implementación CBRM

A partir de la observación de las tablas 4.2 y 4.3 aparece que el CBRM ocupa un área creciente con el tamaño del fragmento t, pero su implementación resulta mucho más favorable que el CORDIC, hasta t=8

En cuanto al tiempo de cálculo, el CBRM involucra 64 x 128 x 128 iteraciones para calcular un número de puntos igual al número de píxels por la mitad del número de ángulos. Cada una de las iteraciones dura $16/t$ ciclos. Se asume que $\tau_t \approx 1$ ns para la tarjeta XS4010XL-PC84.

CBRM	*Implementation 1*	*Implementation 2*
t=1	$64 \cdot 128 \cdot 128 \cdot (3\tau_t + 16\tau_t \lg 16 + 7{,}5\tau_t)$ =78,119 ms	$64 \cdot 128 \cdot 128 \cdot (3\tau_t + 3.3\tau_t + \tau_t \lg 16)$ =16,777 ms
t=2	$64 \cdot 128 \cdot 128 \cdot (3\tau_t + 8\tau_t \lg 8 + 7{,}5\tau_t)$ =36,176 ms	$64 \cdot 128 \cdot 128 \cdot (3\tau_t + 3.3\tau_t + \tau_t \lg 8)$ =15,728 ms
t=4	$64 \cdot 128 \cdot 128 \cdot (4.5\tau_t + 4\tau_t \lg 4 + 7{,}5\tau_t)$ =20,962 ms	$64 \cdot 128 \cdot 128 \cdot (3.5\tau_t + 3.3\tau_t + \tau_t \lg 4)$ =16,253 ms
t=8	$64.128 \cdot 128 \cdot (5\tau_t + 2\tau_t \lg 2 + 7{,}5\tau_t)$ =15,204 ms	$64 \cdot 128 \cdot 128 \cdot (5\tau_t + 3.3\tau_t + \tau_t \lg 2)$ =15,728 ms

Tabla 4.4 Tiempo de cálculo del CBRM

La Tabla 4.4 pone de manifiesto que la implementación 2 del CBRM siempre proporciona mejores resultados que los del CORDIC (0,0262 s).Para t =4 y t =8, incluso la implementación 1 del CBRM es mejor que el CORDIC.

3.3 Comparación CBRM - CORDIC paralelo

En este apartado se considera el cálculo de la HT realizado por una implementación paralela del CORDIC [Bruguera, 1996]. El cálculo de la HT de una imagen N x N con un procesador CORDIC único requiere $N^3/2$ ciclos,

asumiendo que en cada evaluación se obtienen dos valores de ρ. El tiempo puede reducirse por medio de la paralelización. Para ello existen tres aproximaciones distintas: paralelizar los píxeles de la imagen, paralelizar el ángulo θ, o paralelizar los dos simultáneamente. La paralelización de los píxeles requiere $N^3/2$ procesadores, a razón de uno por píxel y por ángulo y la transformación se efectúa en el tiempo de una operación CORDIC (n ciclos para radix-2, $n/2$ +$n/4$ para radix mixto 2-4 y $n/2$ para radix-4, si n es la precisión de los datos). Además, de la cantidad de recursos hardware que supone, esta aproximación acarrea también conflictos en el proceso de votación ya que, los resultados obtenidos por los procesadores que trabajan con el mismo ángulo θ pueden votar el mismo elemento en el espacio de Hough. La introducción del paralelismo sólo en los píxeles requiere N^2 procesadores, uno por píxel. El número de operaciones CORDIC es entonces de $N/2$ y también puede haber conflictos en el proceso de votación. La solución que no produce conflictos es la paralelización de los ángulos. En este caso, se necesita un procesador por ángulo en el cual son procesados secuencialmente todos los píxeles de la imagen. El número total de procesadores es de $N/2$ y el número de ciclos para la evaluación de la transformada es N^2 + la latencia; se procesa un píxel por ciclo.

Comparación

La implementación considerada en [Bruguera, 1996] utiliza una precisión de 12 bits y un procesador CORDIC de 10 etapas (de estas 10 etapas, 6 son las etapas estándar de las iteraciones, 1 es la de la compensación del factor de escala y 3 para la realización del escalado). Cada etapa consta de dos registros, dos multiplexores y dos sumadores/restadores. La etapa estándar necesita 24 bits para cada ángulo almacenado en la ROM.

Para transformar los 128 x 128 puntos de una imagen con 128 rotaciones, en el caso de la paralelización de los ángulos, se necesitan 64 procesadores y se ejecutan 128 x 128 ciclos. El área estimada para este cálculo aparece en la Tabla 4.5.

CORDIC paralelo	*Cantidad.*	*Área ocupada*
Registros	20·64	$20 \cdot 64 \cdot 0.5 \cdot 12\ \tau_a = 7880 \cdot \tau_a$
Multiplexores	20·64	$20 \cdot 64 \cdot 0.25 \cdot 2 \cdot 12\ \tau_a = 7880 \cdot \tau_a$
Sumadores/restadores	20·64	$20 \cdot 64 \cdot 12 \cdot \tau_a =$ $15760\ \tau_a$
Tablas LUT	64·24 bits	$10.64 \cdot 24 \cdot 40\ \tau_a$/Kbit = $60\ \tau_a$
Total	-	**31580 τ_a**

Tabla 4.5 Estimación del área ocupada por el CORDIC paralelo

La estimación de tiempos es la que muestra la Tabla 4.6, asumiendo τ_t=1ns.

CORDIC paralelo	*Cantidad*	*Tiempo*
Registros	10	$128 \cdot 128 \cdot 10 \cdot 1\tau_t = 0{,}164$ ms
Multiplexores	10	$128 \cdot 128 \cdot 10 \cdot 0.5\ \tau_t = 0{,}082$ ms
Sumadores /restadores	10	$128 \cdot 128 \cdot 10 \cdot \lg 12\ \tau_t \approx 0{,}573$ ms
Tablas LUT	64	$64 \cdot 3\ \tau_t = 192$ ns
Total	-	**0.819 ms**

Tabla 4.6 Estimación del tiempo de cálculo en el CORDIC paralelo

La paralelización por medio del CBRM consiste, según la propuesta del capítulo 3, en realizar el cálculo con N módulos organizados en estructura de árbol. En este caso, N = 64 representa el número de incrementos $\Delta\theta$ dentro del intervalo [0, Π/2[, que es también el número de ángulos para los que hay que calcular ρ, para cada uno de los 128 x128 píxeles.

	Tiempo de cálculo	*Número de módulos necesarios*
Cálculo paralelizado	$=TN_1+TN_2+.....+TN_p$ $= T(N_1+N_2+.....+N_p)$	$1+N_1+N_1N_2+N_1N_2N_3.+...+.N_1N_2N_{3...}N_{p-1}$

La descomposición de N en un producto $N_1N_2.....N_p$ permite definir la profundidad del árbol y, además, la distribución de los módulos por etapa incide en las prestaciones de la arquitectura Para realizar la comparación con la propuesta CORDIC paralelo, se plantean dos descomposiciones de N, una que maximiza el tiempo de cálculo y la otra que maximiza el número de módulos del árbol. Así, efectuará la comparación con las peores condiciones del CBRM.

- Descomposición que maximiza el tiempo de cálculo

$N = N_1 \times N_2$ con $N_1 = 2$ y $N_2 = 32$

Tiempo de cálculo = (2 + 32)T = 34 T

Número de módulos de cálculo del árbol = 1+2 = 3

- Descomposición que maximiza el número de módulos

$N = N_1 \times N_2 \times N_3 \times N_4 \times N_5 \times N_6$ con $N_i = 2$ para $i \in [1, 6]$

Tiempo de cálculo = (2+2+2+2+2+2)T = 12 T , siendo T el tiempo de cálculo de un valor.

Número de módulos de cálculo del árbol = 1+2+4+8+16+32 = 63

Las Tablas 4.7a y b representan el área total ocupada por el CBRM en los dos supuestos, el que maximiza el tiempo, $N_1 = 2$, $N_2 = 32$ y el que maximiza el área ocupada, $N_i = 2$ para $i \in [1, 6]$.

n = 16	*Implementación con sumador secuencial*	*Implementación con reductores 4:2*
t=1	2 x 3 x 34 τ_a = 408 τ_a	2 x 3 x 33 τ_a = 198 τ_a
t=2	2 x 3 x 64 τ_a = 384 τ_a	2 x 3 x 63 τ_a = 378 τ_a
t=4	2 x 3 x 584 τ_a = 3504 τ_a	2 x 3 x 583 τ_a = 3498 τ_a
t=8	2 x 3x 102424 τ_a = 614544 τ_a	2 x 3 x 102423 τ_a = 614538 τ_a

Tabla 4.7a Estimación del área ocupada por el CBRM paralelo para N = 64
($N_1 = 2$ y $N_2 = 32$, tiempo de cálculo máximo)

n = 16	*Implementación con sumador secuencial*	*Implementación con reductores 4:2*
t=1	2 x 63 x 34 τ_a = 4284 τ_a	2 x 63 x 33 τ_a = 4158 τ_a
t=2	2 x 63 x 64 τ_a = 8064 τ_a	2 x 63 x 63 τ_a = 7938 τ_a
t=4	2 x 63 x 584 τ_a = 73584τ_a	2 x 63 x 583 τ_a = 73458 τ_a
t=8	2 x 63 x 102424 τ_a = 12905424 τ_a	2 x 63 x 102423 τ_a = 12905298 τ_a

Tabla 4.7b Estimación del área ocupada por el CBRM paralelo para N = 64
($N_i = 2$ para $i \in [1, 6]$, área ocupada máxima)

Las Tablas 4.8a y b representan el tiempo de cálculo de los 128 x 128 puntos, en los dos supuestos: el que maximiza el tiempo, $N_1 = 2$, $N_2 = 32$ y el que maximiza el área ocupada, $N_i = 2$ para $i \in [1, 6]$, asumiendo que τ_t =1ns.

n =16	*Implementación con sumador secuencial*	*Implementación con reductores 4:2*
t = 1	557056 x 74,5 = 0,041 ms	557056 x 16 = 0,009 ms
t = 2	557056 x 34,5 = 0,019 ms	557056 x 15 = 0,008 τ_t ms
t = 4	557056 x 20 = 0,011 ms	557056 x 15,5 = 0,009 τ_t ms
t = 8	557056 x 14,5 = 0,008 ms	557056 x 15 = 0,008 τ_t ms

Tabla 4.8a Estimación del tiempo de cálculo del CBRM paralelo para N = 64
(N_1 = 2 y N_2 = 32, tiempo de cálculo máximo)

n =16	*Implementación con sumador secuencial*	*Implementación con reductores 4:2*
t = 1	196608 X 74,5 = 0,015 ms	196608 X 16 = 0,003 ms
t = 2	196608 X 34,5 = 0,007 ms	196608 X 15 = 0,003 ms
t = 4	196608 X 20 = 0,004 ms	196608 X 15,5 = 0,003 ms
t = 8	196608 X 14,5 = 0,003 ms	196608 X 15 = 0,003 ms

Tabla 4.8b Estimación del tiempo de cálculo del CBRM paralelo en función de m para N = 64
(N_i = 2 para $i \in [1, 6]$, área ocupada máxima)

Comparando la Tabla 4.7b, que representa los peores resultados en cuanto a área ocupada del CBRM paralelo, con la Tabla 4.5, aparece que para valores de t<4, la implementación paralela del CBRM ocupa menos espacio que el procesador CORDIC. El tiempo de cálculo correspondiente que aparece en la Tabla 4.8 b es entre 50 y 100 veces menor para el CBRM que para el CORDIC.

Comparando la Tabla 4.8 a, que representa los peores resultados en cuanto a tiempo del CBRM paralelo, con la Tabla 4.6, aparece que en el peor caso del CBRM, que se da para la implementación con sumador secuencial y t =1, el tiempo es 20 veces menor que en el CORDIC. La ocupación de área correspondiente que aparece en la Tabla 4.7a es unas 10 veces menor en el CBRM para t<8.

En resumen: es destacable el buen comportamiento del CBRM en cuanto al tiempo de cálculo. En cuanto al área ocupada, a partir de $t > 4$, los resultados pueden llegar a ser menos buenos que los del procesador CORDIC.

Por otra parte, cabe mencionar que los datos manejados en las tablas 4.7a y b y 4.8a y b tienen una precisión de 16 bits frente a los 12 bits de la implementación CORDIC, lo que refuerza aún más las conclusiones mencionadas.

El trabajo referenciado no proporciona resultados del error cometido por la arquitectura CORDIC estudiada.

Conviene decir que la implementación del CBRM descrita aquí puede proporcionar tiempos de cálculo todavía más cortos si se proponen cuatro funciones en lugar de dos para la evaluación, $\rho_I(k)$ $\rho_{II}(k)$ $\rho_{III}(k)$ $\rho_{IV}(k)$. El número de incrementos es entonces N/2. Este cambio repercute la descomposición, ocasionando menor tiempo de cálculo y menor número de módulos en el árbol, pero hay que duplicar el área total porque se duplica el número de procesadores.

4 Aplicación del CBRM a la transformada de Fourier

En este apartado se realiza un análisis formal de la transformada de Fourier para la cual se propone la aplicación del CBRM, como método idóneo de cálculo. El CBRM se aplica después de organizar los cálculos siguiendo el esquema de doblamientos sucesivos. Se han seleccionado dos propuestas de cálculo de la transformada de Fourier a fin de compararlas con la implementación por el CBRM. La primera considera una implementación basada en aritmética distribuida, con replicación segmentada de las etapas, que se prueba en una FPGA [Mintzer, 1996]. La segunda propuesta combina la convolución cíclica

con la aritmética distribuida por bloques y utiliza la descomposición de Cooley y Tuckey [Chien, 2002].

4.1 Transformada de Fourier

A pesar de su evolución hacia el dominio espacial, las técnicas de mejora de la imagen en el dominio frecuencial siguen motivando gran cantidad de investigación. En este sentido, la transformada de Fourier constituye una herramienta de referencia obligada, útil en procesos de filtrado [Chamberlain, 2002] [Peiming, 2001] y de reconstrucción de imágenes [Mozón, 1996]. Recientemente, se ha utilizado la transformada rápida de Fourier (FFT) en modulación (OFDM: Orthogonal Frequency División Multiplexing), resultando así una herramienta valiosa en el ámbito de las comunicaciones [Chang, 2000] [Hsiao, 2000].

El algoritmo más general desarrollado para el cálculo de la FFT es el de Cooley y Tukey, de 1965, basado en el método del doblamiento sucesivo [Cooley, 1965]. El proceso, que consiste en la descomposición de la transformada de una secuencia en múltiples transformadas sobre secuencias de menor tamaño, es la base de todos los algoritmos de la categoría FFT y su principal aportación radica en la reducción drástica de la complejidad computacional. Así, la computación de una secuencia de N elementos, que ocasiona, según la definición, una complejidad algorítmica de $O(N^2)$, se reduce a $O(N \log_2 N)$ con el método FFT.

Las variantes del algoritmo difieren entre sí por las computaciones realizadas y por la forma en que se almacenan los resultados intermedios [Swarztrauber, 1987]. Estas diferencias originan mejoras sustanciales de alguna prestación, como el ahorro de almacenamiento en memoria, en el caso de los algoritmos *in-place;* la velocidad elevada en el caso de los algoritmos *self-sorting,*[Temperton

1991] o la regularidad de la arquitectura de implementación en los algoritmos *geometría constante* [Pease, 1968]. Existen también modos de extender las mejoras combinando algunos de estos esquemas, como computar más de una etapa a la vez, o combinar algunas de las estrategias mencionadas según la etapa en curso [Englewood, 1975].

Cada variante del algoritmo consigue un rendimiento óptimo con una determinada organización del hardware, ya que el carácter óptimo en cuanto a complejidad computacional suele obtenerse sacrificando la regularidad del algoritmo. Por ello, los esquemas *in-place* y *geometría constante* marcan dos tendencias distintas del hardware. Los primeros se implementan generalmente en arquitecturas segmentadas puesto que así se minimiza la latencia entre etapas y la longitud de las memorias utilizadas [Chang, 2001], mientras que los segundos, al presentar una forma de indexar constante a través de todas sus etapas, dan lugar a una estructura más regular y a un control más simple. Ello permite un procesamiento basado en una columna de procesadores que computan en paralelo sobre los datos y, usando una red de interconexión fija, los recirculan sobre la misma, optimizando tanto los elementos de procesamiento como las comunicaciones [Fang, 1997], [Chan, 1993].

Sea $F(u)$ la transformada de Fourier discreta (DFT) de una función real unidimensional $f(x)$ de N puntos. La transformada de Fourier es compleja y se expresa en (4.11).

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)\exp\frac{-2j\pi ux}{N} \qquad (4.11)$$

Desarrollo de la DFT basado en el esquema de doblamientos sucesivos

Por conveniencia se expresará la ecuación (4.11) como sigue

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x) W_{2M}^{ux} \qquad (4.12)$$

donde $W_N = \exp\frac{-2j\pi}{N}$

Se supone que $N = 2^n = 2M$, donde n es un entero positivo. La ecuación (4.12) puede expresarse entonces como

$$F(u) = \frac{1}{2}\left(\frac{1}{M}\sum_{x=0}^{M-1} f(2x) W_M^{u(2x)} + \frac{1}{M}\sum f(2x+1) W_M^{u(2x+)}\right) \qquad (4.13)$$

como $W_{2M}^{2ux} = W_{2M}^{ux}$ la ecuación queda en

$$F(u) = \frac{1}{2}\left(\frac{1}{M}\sum_{x=0}^{M-1} f(2x) W_M^{u(2x)} + \frac{1}{M}\sum f(2x+1) W_M^{ux} W_{2M}^{u}\right) \qquad (4.14)$$

definiendo $F_{par}(u) = \frac{1}{M}\sum_{x=0}^{M-1} f(2x) W_M^{ux}$

$$F_{impar}(u) = \frac{1}{M}\sum_{x=0}^{M-1} f(2x+1) W_M^{ux}$$

para u = 0, 1, 2,.....M-1, la ecuación se reduce a

$$F(u) = \frac{1}{2}\left(F_{par}(u) + F_{impar}(u) w_{2M}^{u}\right) \qquad (4.15)$$

Asimismo, como $W_M^{u+M} = W_M^u \; y W_{2M}^{u+M} = -W_{2M}^u$ se deduce que.

$$F(u+M) = \frac{1}{2}\left(F_{par}(u) - F_{impar}(u) W_{2M}^{u}\right) \qquad (4.16)$$

Del análisis de estas ecuaciones se deducen algunas propiedades interesantes. Una transformada de N puntos puede calcularse dividiendo la expresión original en dos partes como indican las ecuaciones (4.15) y (4.16). El cálculo de la primera mitad necesita la evaluación de las dos transformadas de $N/2$ puntos, F_{par} y F_{impar} y la otra mitad se deduce entonces directamente, sin necesidad de más cálculos. Repitiendo el razonamiento, se remonta a la transformada de tamaño mínimo que es 2. Así, en lugar de calcular una transformada de $N = 2^n$ puntos, se puede calcular 2^{n-1} transformadas de 2 puntos que se van componiendo por parejas de tipo F_{par} ,F_{impar}, a lo largo de n etapas para terminar dando la transformada entera. Las transformadas iniciales de 2 puntos se forman a partir de dos valores de $f(x)$, con $x = 0, 1, \ldots, N$, emparejando los valores según el procedimiento de inversión de bits. Si x representa el argumento de $f(x)$, el correspondiente argumento de la distribución ordenada se obtiene expresando x en forma binaria e invirtiendo el orden de sus bits. La estructura de una pareja inicial es $f(x) + f(\overline{x})W_M^{ux}$ donde $\overline{x}$ representa el argumento con los bits invertidos con respecto a x, no el complementario.

Aplicación del CBRM

Se sigue con la transformada de Fourier de $N = 2^n = 2M$ puntos. Se procede a plantear el cálculo simultáneo pero separado de la parte real e imaginaria de la transformada.

El número de parejas iniciales será entonces de $2M$ y su estructura

$$f(x) + f(\overline{x})\cos 2^{n-1}\frac{\pi u}{M} \text{ (parte real)}$$

$$f(x) + f(\overline{x})sen2^{n-1}\frac{\pi u}{M} \text{ (imaginaria).}$$

Sean $R_{ed\ par}(u)$, $R_{ed\ impar}(u)$, $I_{ed\ par}(u)$, $I_{ed\ impar}(u)$ las partes, real par e impar, imaginaria par e impar, respectivamente, de los resultados parciales de la transformada de Fourier, donde *e* es un indicador de la etapa de cálculo y *d* el número de orden dentro de la etapa.

Se tomará $\alpha_e = \cos(2^{n-e}\frac{\pi u}{M})$ y $\beta_e = sen(2^{n-e}\frac{\pi u}{M})$

- 1ª etapa: Calcular las *M* transformadas iniciales de 2 puntos, en su parte real e imaginaria.

$R_{00\ par}(u) = f(0) + \alpha_1(u) f(2^{n-1})$	$I_{00\ par}(u) = -\beta_1(u) f(2^{n-1})$
$R_{01im\ par}(u) = f(2^{n-2}) + \alpha_1(u) f(2^{n-2} + 2^{n-1})$	$I_{01\ im\ par}(u) = -\beta_1(u) f(2^{n-2} + 2^{n-1})$
.....	.....
$R_{0M\text{-}1\ im\ par}(u) = f(2 + 2^2 ... + 2^{n-2}) + \alpha_1(u) f(2 + 2^2 ... + 2^{n-2} + 2^{n-1})$	$I_{0M\text{-}1\ impar}(u) = -\beta_1(u) f(2 + 2^2 + ... 2^{n-2} + 2^{n-1})$

(4.17)

En lo que sigue, en aras a la claridad, se omite el argumento *u* en las expresiones del cálculo.

- 2ª etapa: Pasar de las *M* transformadas de 2 puntos a *M*/2 transformadas de 4 puntos

$R_{1,0\ par} = R_{0,0\ par} + \alpha_2 R_{0,1\ impar} - \beta_2 I_{0,1\ impar}$	$I_{1,0\ par} = I_{0,0\ par} + \beta_2 R_{0,1\ impar} + \alpha_2 I_{0,1\ impar}$
$R_{1,1\ impar} = R_{0,2\ par} + \alpha_2 R_{0,3\ impar} - \beta_2 I_{0,3\ impar}$	$I_{1,1impar} = I_{0,2\ par} + \beta_2 R_{0,3\ impar} + \alpha_2 I_{0\ 3\ impar}$

$$
\begin{array}{ll}
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
R_{1,M/2\text{-}1impar}=R_{0,M/2\ par}+\alpha_2\ R_{0,M/2+1\ impar} - \beta_2\, I_{0,M/2+1\ impar} & I_{1,M/2\text{-}1impar}=I_{0,M/2\ par}+\beta_2\, R_{0,M/2+1\ impar} + \alpha_2\, I_{0,M/2+1\ impar}
\end{array}
\qquad (4.18)
$$

- 3° etapa: Pasar de las *M/2* transformadas de 4 puntos a *M/4* transformadas de 8 puntos

$$
\begin{array}{ll}
R_{2,0\ par}=R_{1,0\ par}+\alpha_3\ R_{1,1\ impar} - \beta_3\, I_{1,1impar} & I_{2,0\ par}=I_{1,0\ par}+\beta_3\ R_{1,1\ impar}+\alpha_3\, I_{1,1impar} \\
R_{2,1\ impar}=R_{1,2\ par}+\alpha_3\ R_{1,3\ impar} - \beta_3\, I_{1,3\ impar} & I_{2,1\ impar}=R_{1,2\ par}+\beta_3\ R_{1,3\ impar}+\alpha_3\, I_{1,3\ impar} \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
R_{2,M/4\text{-}1\ impar}=R_{1,M/4\ par}+\alpha_3\ R_{1,M/4+1\ impar} - \beta_3\, I_{1,M/4+1\ impar} & I_{2M/4\text{-}1im\ par}=R_{1,M/4\ par}+\beta_3\ R_{1,M/4+1\ impar}+\alpha_3\, I_{1,M/4+1\ impar}
\end{array}
\qquad (4.19)
$$

- *n*-ésima etapa: Pasar de las 2 transformadas de *M* puntos a una transformada de *N*=2*M* puntos

$$
\begin{array}{ll}
\boldsymbol{R}=R_{n\text{-}1,0}=R_{n\text{-}2,0\ par}+\alpha_n\ R_{n\text{-}2,1\ impar} - \beta_n\, I_{n\text{-}2,1\ impar} & \boldsymbol{I}=I_{n\text{-}1,0}=I_{n\text{-}2,0\ par}+\beta_n\ R_{n\text{-}2,1impar}+\alpha_n\, I_{n\text{-}2,\ 1\ impar}
\end{array}
\qquad (4.20)
$$

En realidad, estos cálculos sólo se llevan a cabo para la primera mitad de los valores de u, u = 0, 1, ...M-1, ya que la segunda mitad precisa de los mismos

cálculos salvo la última etapa en la cual hay una modificación en los signos. Después de una demostración sencilla se llega a la expresión resultante (4.21):

$$\boldsymbol{R} = R_{n-1,0} = R_{n-2,0\ par} - \alpha_n\ R_{n-2,1\ impar} + \beta_n\ I_{n-2,1\ impar} \qquad \boldsymbol{I} = I_{n-1,0} = I_{n-2,0\ par} - \beta_n\ R_{n-2,1 impar} - \alpha_n\ I_{n-2,\ 1\ impar} \qquad (4.21)$$

para $u = M, M+1, \ldots\ldots, N-1$

La aplicación del CBRM se pone de manifiesto reordenando las operaciones a partir de la segunda etapa

- 1ª etapa: ecuaciones (4.17)
- 2ª etapa: 2 x $M/2$ operaciones

$$R_{1,0\ par} = R_{0,0\ par} + \alpha_2\ R_{0,1\ impar} - \beta_2\ I_{0,1\ impar}$$
$$I_{1,0\ par} = I_{0,0\ par} + \beta_2\ R_{0,1\ impar} + \alpha_2\ I_{0,1\ impar}$$
$$R_{1,1\ impar} = R_{0,2\ par} + \alpha_2\ R_{0,3\ impar} - \beta_2\ I_{0,3\ impar}$$
$$I_{1,1impar} = I_{0,2\ par} + \beta_2\ R_{0,3\ impar} + \alpha_2\ I_{0\ 3\ impar}$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$R_{1,M/2-1impar} = R_{0,M2\ par} + \alpha_2\ R_{0,M/2+1\ impar} - \beta_2\ I_{0,M/2+1\ impar}$$
$$I_{1,M/2-1impar} = I_{0,M/2\ par} + \beta_2\ R_{0,M/2+1\ impar} + \alpha_2\ I_{0,M/2+1\ impar}$$

- 3ª etapa: 2 x M/4 operaciones

$$R_{2,0\ par} = R_{1,0\ par} + \alpha_3\ R_{1,1\ impar} - \beta_3\ I_{1,1\ impar}$$
$$I_{2,0\ par} = I_{1,0\ par} + \beta_3\ R_{1,1\ impar} + \alpha_3\ I_{1,1\ impar}$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$R_{2,M/4-1\ impar} = R_{1,M/4\ par} + \alpha_3\ R_{1,M/4+1\ impar} - \beta_3\ I_{1,M/4+1\ impar}$$

$I_{2M/4\text{-}1im\ par} = R_{1,M/4\ par} + \beta_3\ R_{1,M/4+1\ impar} + \alpha_3 I_{1,M/4+1\ impar}$

.....

- n-ésima etapa: 2 x 2 operaciones

$$\boldsymbol{R} = R_{n\text{-}1,0} = R_{n\text{-}2,0\ par} + \alpha_n\ R_{n\text{-}2,1\ impar} - \beta_n I_{n\text{-}2,1\ impar}$$
$$\boldsymbol{I} = I_{n\text{-}1,0} = I_{n\text{-}2,0\ par} + \beta_n\ R_{n\text{-}2,1impar} + \alpha_n I_{n\text{-}2,\ 1\ impar}$$

$$\boldsymbol{R} = R_{n\text{-}1,0} = R_{n\text{-}2,0\ par} - \alpha_n\ R_{n\text{-}2,1\ impar} + \beta_n I_{n\text{-}2,1\ impar}$$
$$\boldsymbol{I} = I_{n\text{-}1,0} = I_{n\text{-}2,0\ par} - \beta_n\ R_{n\text{-}2,1impar} - \alpha_n I_{n\text{-}2,\ 1\ impar}$$

Se observa que, a partir de la 2ª etapa, cualquier par R_{ed}, I_{ed} se calcula como una rotación, definida por los parámetros α_e, β_e, seguida de una traslación, definida por el parámetro 1. En cada etapa, las rotaciones se realizan con un ángulo distinto, propio de la etapa. Queda patente que los resultados de dos rotaciones consecutivas (cuatro ecuaciones) en una etapa e, intervienen como argumentos en una misma rotación (dos ecuaciones) en la etapa siguiente e+1, por lo que se produce, en cada etapa, una división por dos del número de ecuaciones, hasta llegar al resultado final.

La operación básica del CBRM aplicada a este caso se realiza de la manera descrita en el capítulo 3, esto es, accediendo a una tabla LUT y efectuando la suma de los productos parciales extraídos. En este caso, las casillas de la tabla LUT contienen combinaciones de los tres parámetros (α, β, 1).

Coste computacional

El número de operaciones CBRM a realizar se establece por inducción. Se cuentan tanto las operaciones que calculan las partes real como las que calculan la

parte imaginaria, para cada valor de *u,* considerando calculadas las parejas iniciales

$N = 4, n = 2, M = 2$

F(0): 1 suma
F(1): 2 x 3 = 6 operaciones CBRM
F(2): deducido de F(0), 1 suma
F(3): deducido de F(1), 2 x 1 = 2 operaciones CBRM (cambio de signo)
Total: 8 operaciones CBRM y 2 sumas.

$N = 8, n = 3, M = 4$

F(0): 3 sumas
F(1), F(2) y F(3) = 14 operaciones CBRM
F(4):3 sumas
F(5), F(6) y F(7) = 2 x 3 = 6 operaciones CBRM (cambio de signo)
Total: 20 operaciones CBRM y 6 sumas

$N = 16, n = 4, M = 8$

F(0): 7 sumas
F(1), F(2), F(3).....F(7) = 30 operaciones CBRM
F(8): 7 sumas
F(9),.....,F(15) = 2 x 7 = 14 operaciones CBRM (cambio de signo)
Total: 44 operaciones CBRM y 14 sumas

.....

Continuando este razonamiento, se plantea la hipótesis siguiente:

operaciones CBRM(n) = 2x operaciones CBRM(n-1) + 4

sumas(n) = 2 x sumas (n-1) + 2

Demostración por inducción

Suponiendo que:

Operaciones CBRM (1) = 0

Sumas (1) = 1

para $n>1$

el número de operaciones CBRM es

operaciones CBRM(n) = $2(2^n-1) + (2^n-2) = 2^{n+1} + 2^n - 4$

y el número de sumas es

sumas(n)= $2^n - 2$

entonces, para $n+1$ se tiene

operaciones CBRM($n+1$) = $2^{n+2} + 2^{n+1} - 4$

sumas($n+1$) = $2^{n+1} - 2$

A partir de estas expresiones, se comprueba que para $n>1$

Operaciones CBRM($n+1$) = 2 x operacionesCBRM(n) + 4

sumas(n) = 2 x sumas (n-1) + 2

con lo que queda demostrada la hipótesis planteada.

4.2 Comparación CBRM- diseño con segmentación de etapas

Descripción de la arquitectura segmentada

En esta propuesta [Mintzer, 1996], el cálculo de las parejas iniciales se obtiene utilizando aritmética distribuida. La aritmética distribuida es una modificación a nivel de bit de la multiplicación por acumulación escalada [White, 1989], cuyo objetivo es ocultar dicha multiplicación.

La arquitectura mínima se construye para una transformada de 16 puntos que puede incrementarse hasta 8192 puntos replicando la etapa. Cada etapa consta de cuatro registros paralelo-serie (P/S-R), dos acumuladores escalados (MAC) y dos LUTs de 32x16 bits. Los valores de los senos y cosenos figuran en las tablas. En cada etapa los valores de la LUT cubren un intervalo igual al incremento angular de la etapa precedente que se divide a su vez en incrementos menores. Este diseño con segmentación de las etapas evita el crecimiento exponencial de la memoria que es el mayor inconveniente para una implementación en FPGA. Para una transformada de 8192 puntos se requieren cuatro etapas. Se prueba este diseño en una plataforma Xilinx XC 4025. El cálculo de las parejas iniciales se realiza en 320 ns para datos de 16 bits, a una frecuencia de reloj de 50 Hz, Sabiendo que el número de mariposas es de $N/2\ \log_2 N$, el cálculo total de la transformada de 16 puntos tarda 10,34 μs. Después de una replicación por cuatro, la transformada de 8192 puntos se realiza en 17 ms.

A fin de facilitar una comparación con los resultados que proporciona el CBRM, he realizado unas estimaciones del espacio ocupado y del tiempo de cálculo invertidos por la arquitectura segmentada para transformadas de 16, 8192 y 61K puntos, con datos de 16 bits.

Estimación del área ocupada y del tiempo de cálculo en la arquitectura segmentada

En este apartado la estimación se realiza en base a las unidades τ_a y τ_t

- Transformada de 16 puntos (una etapa)

Estimación del área ocupada

4 (P/S-R) = 4 x 0,5·16· τ_a = 32· τ_a
2 (S/P-R) = 2 x 0,5·16· τ_a = 16· τ_a
2 (MAC) = 2 x16· τ_a =· 32τ_a
2 LUTs 32x16 bits =· 40τ_a

Total $120\tau_a$

Estimación del tiempo de cálculo (camino crítico a través de la etapa)

1 (P/S-R) = $1\tau_t$
1 (MAC) = lg 8 τ_t =· $3\tau_t$
1 LUTs 32x16 bits = $3.5\tau_t$ x 16 accesos = 56 τ_t
Total: N/2 $\log_2$N x ($1\tau_t + 3\tau_t + 56\tau_t$) = 12 x 60 τ_t **=·720 τ_t**

- Transformada de 8192 puntos (4 etapas)

Estimación del área ocupada

1ª etapa $120\tau_a$

2ª 3ª etapas
2 (P/S-R) = 2 x 0,5·16· τ_a = 16· τ_a
2 shift registers (S-R) = 2 x 0,5·16· τ_a =·$16\tau_a$
2 (MAC) = 2 x16· τ_a =· $32\tau_a$
2 LUTs 32x16 bits = $40\tau_a$

4ª etapa
2 (P/S-R) = 2 x 0,5·16· τ_a = 16· τ_a
2 (S/P-R) = 2 x 0,5·16· τ_a = 16· τ_a
2 (S-R) = 2 x 0,5·16· τ_a =·$16\tau_a$
2 (MAC) = 2 x16· τ_a =· $32\tau_a$
2 LUTs 32x16 bits =40 τ_a
2 registros =2 x 0,5·16· τ_a =· $16\tau_a$
Total: $120\tau_a$ + 2 x 104 τ_a + 136 τ_a **= 464 τ_a**

Estimación del tiempo de cálculo (camino crítico a través de 4 etapas)

1ª etapa
1 (P/S-R) = $1\tau_t$
1 (MAC) = lg 8 τ_t =· $3\tau_t$
1 LUTs 32x16 bits = $3.5\tau_t$ x 16 accesos = 56 τ_t
56 τ_t + $4\tau_t$ + $1\tau_t$ = 61 τ_t

2ª etapa
1 (P/S-R) = $1\tau_t$
1 (MAC) = lg 64 τ_t =· $6\tau_t$
1 LUTs 32x16 bits = $3.5\tau_t$ x 16 accesos = 56 τ_t

$56\ \tau_t + 6\tau_t + 1\tau_t = 63\ \tau_t$

3ª etapa
1 parallel-serial registers (P/S-R) = $1\tau_t$
1 scaled accumulators (MAC) = lg 512 τ_t = $9\tau_t$
1 32x16 bits LUTs = $3.5\tau_t$ x 16 accesos = $56\ \tau_t$
$56\ \tau_t + 9\tau_t + 1\tau_t = 66\ \tau_t$

4º etapa
1 (P/S-R) = $1\tau_t$
1 (MAC) = lg 4096 τ_t = $12\tau_t$
1 LUTs 32x16 bits = $3{,}5\tau_t$ x 16 accesos = $56\ \tau_t$
1 registro = 2 x $0{,}5\cdot\ \tau_t$ = $1\tau_t$
$56\tau_t + 12\tau_t + 1\tau_t + 1\tau_t = 70\ \tau_t$

Total: $(N_1/2\ \log_2 N_1)\ 60 + (N_2/2\ \log_2 N_2)\ 63 + (N_3/2\ \log_2 N_3)\ 66\ \tau_t + (N_4/2\ \log_2 N_4)\ 70\ \tau_t$ = $12.61 + 192.63 + 2304\ 66\ \tau_t + 24576\ 70\ \tau_t$ = **1885200** τ_t

- Transformada de 64K puntos

Estimación del área ocupada
1ª etapa $120\tau_a$
2ª etapa $104\tau_a$
3ª etapa $104\tau_a$
4ª etapa $104\tau_a$
5ª etapa $136\tau_a$
Total $\mathbf{568\tau_a}$

Estimación del tiempo de cálculo (camino crítico a través de 5 etapas)

Total: $(N_1/2\ \log_2 N_1)\ 60 + (N_2/2\ \log_2 N_2)\ 63 + (N_3/2\ \log_2 N_3)\ 66 + (N_4/2\ \log_2 N_4)\ 69\ \tau_t + (N_5/2\ \log_2 N_5)\ 73\ \tau_t$ = $12.61 + 192.63 + 2304\ 66\ \tau_t + 24576\ 69\ \tau_t + 245760\ 73\ \tau_t$ = **19825740** τ_t

Estimación del área ocupada y del tiempo de cálculo en el CBRM

Las estimaciones se realizan para la implementación 1 y la implementación 2, para datos de 16 bits utilizando la arquitectura el CBRM y dos sumadores adicionales.

Estimación del área ocupada por un módulo CBRM

LUT: 40 τ_a/Kbit x 16 bits x 64 celdas= 40 τ_a
Sumador serie: 16 τ_a
Estructura de reducción 4:2+ sumador: 4 τ_a + 16 τ_a = 20 τ_a
Multiplexor: 0,25·x 2 x 16·τ_a = 8 τ_a
Registro: 0,5·16·τ_a = 8 τ_a
Registro de desplazamiento 0,5 16 τ_a = 8 τ_a

Estimación del tiempo de cálculo del CBRM (camino crítico):

LUT: 3.5τ_t x 16 accesos = 56 τ_t
Sumador serie = 16 lg16 τ_t. = 64 τ_t
Estructura de reducción 4:2: 3 τ_t + lg16 τ_t.= 7τ_t
Multiplexor: 0,5 τ_t
Registro:1 τ_t
Registro de desplazamiento: 0,5 τ_t

Teniendo en cuenta que se evalúan dos funciones a la vez de forma cruzada, para cualquier número de puntos, el área total será el doble del área que ocupa la arquitectura CBRM. Hay que prever igualmente dos sumadores secuenciales o dos estructuras de reducción y suma para realizar las sumas adicionales. Para el cálculo de los tiempos se utilizan las ecuaciones relativas al número de operaciones CBRM y de sumas establecidas en el apartado 4.1.

CBRM(n) = 2^{n+1} + 2^n –4 (la mitad corresponde a la parte real y la otra mitad a la parte imaginaria)
sumas(n)= 2^n – 2 (la mitad corresponde a la parte real y la otra mitad a la parte imaginaria)

Área total ocupada:
Implementación1: 2 x (40 τ_a + 16 τ_a + 2x 8 τ_a + 8 τ_a) = **168 τ_a**
Implementación2: 2 x (40 τ_a + 20 τ_a + 2x 8 τ_a + 8 τ_a) = **176 τ_a**

- Transformada de 16 puntos:

Tiempo total:

Implementation 1: (2^4+2^3-2) x (56 τ_t + 64 τ_t + 2x 0,5τ_t +1τ_t+ 8τ_t) + (2^3-1) 65 τ_t = **3380 τ_t**

Implementation 2: (2^4+2^3-2) x (56 τ_t + 7 τ_t + 2x 0,5τ_t +1τ_t) + (2^3-1) 7 τ_t = **1479 τ_t**

- Transformada de 8192 puntos:

Tiempo total:

Implementation 1: $(2^{13}+2^{12}-2)$ x (56 τ_t + 64 τ_t + 2x 0,5τ_t +1τ_t+ 8τ_t)+$(2^{12}-1)$ 65 τ_t =**1863355 τ_t**

Implementation 2: $(2^{13}+2^{12}-2)$ x (56 τ_t + 64 τ_t + 2x 0,5τ_t +1τ_t) + $(2^{12}-1)$ 7 τ_t = **827255 τ_t**

- Transformada de 64K puntos:

Tiempo total:

Implementation 1: $(2^{16}+2^{15}-2)$ x (56 τ_t + 64 τ_t + 2x 0,5τ_t +1τ_t+ 8τ_t)+$(2^{16}-1)$ 64 τ_t = **13238005 τ_t**

Implementation 2: $(2^{16}+2^{15}-4)$ x (56 τ_t + 64 τ_t + 2x 0,5τ_t + 1τ_t) + $(2^{16}-1)$ 7 τ_t = **6848245 τ_t**

		Transformada de 16 puntos		*Transformada de 8192 puntos*		*Transformada de 64K puntos*	
		Area	*Tiempo*	*Area*	*Tiempo*	*Area*	*Tiempo*
Arquitectura segmentada.		**120 τ_a**	**0,7 10 $^3\tau_t$**	**464τ_a**	**1,8 10 $^6\tau_t$**	**568 τ_a**	**1,9 10^7 τ_t**
CBRM	*Impl.1*	168 τ_a	3,38 $10^3\tau_t$	168 τ_a	1,86 10^6 τ_t	168 τ_a	1,32 $10^7\tau_t$
	Impl. 2	176 τ_a	1,48 10^3 τ_t	176 τ_a	0,83 10^6 τ_t	176 τ_a	0,68 10^7 τ_t

Tabla 4.9 Comparación entre arquitectura segmentada y CBRM en ocupación de espacio y tiempo

Conclusión

Los datos que aparecen en la Tabla 4.9 demuestran que el CBRM es una opción que mejora la arquitectura segmentada a medida que aumenta el número de puntos a calcular, tanto en la ocupación de área como en tiempo de cálculo. También es de destacar que la implementación 2 del CBRM es el doble de rápida que la implementación 1 pero que esta ventaja es cada vez menos relevante a medida que aumenta el número de puntos a calcular.

4.3 Comparación CBRM-diseño basado en aritmética distribuida por bloques (BDA)

Descripción de la propuesta BDA

Esta propuesta presenta el cálculo de la transformada de Fourier de longitud variable por medio del control de los parámetros de su arquitectura [Chien, 2002]. El elemento básico procesa 16/32/64 puntos utilizando la descomposición de Cooley y Tuckey de radix 4. Cuando el número de puntos N aumenta, se plantea $N = N_1 \times N_2$ y el procesamiento tiene lugar en una combinación de elementos dispuestos en filas y columnas. Desde el punto de vista formal, los cuatro términos de la descomposición de Cooley y Tuckey se plantean como una convolución cíclica que permite efectuar los cálculos mediante aritmética distribuida basada en bloques. Esto consiste en particionar la memoria en bloques para alojar en cada uno de ellos un grupo de coeficientes que intervienen en las multiplicaciones que figuran en las expresiones. El orden en que tengan que intervenir los coeficientes será controlado por un rotador externo, evitando así tener que guardar en el bloque todas las combinaciones de los mismos elementos (tal y como ocurre en la aritmética distribuida clásica). Esta arquitectura consigue ahorro de memoria a cambio de un aumento del

tiempo de cálculo y de un rotador añadido al circuito. Además, en esta propuesta, se sustituye la memoria ROM por memoria RAM, a fin de tener flexibilidad a la hora de cambiar los contenidos de memoria cuando se quiere cambiar el tamaño de la transformada.

Siguiendo el orden del flujo de datos, la columna básica consta de un buffer de entrada, un procesador CORDIC que realiza las operaciones de multiplicación compleja del pre y postprocesamiento, seguido por un registro paralelo-serie (P/S) y un rotador; cuatro memorias RAM y dieciséis acumuladores implementan la aritmética de bloques. Al final hacen falta cuatro buffers que reordenan los productos parciales para efectuar la operación mínima de cuatro puntos. La complejidad algorítmica de esta propuesta es $O(\frac{N_1}{4M} x W_L)$, siendo

N_1 el tamaño de la transformada

$M = 4$ en este diseño

W_L la precisión de los datos.

Cuando se tiene una transformada larga (de más de 64 puntos) el factor N_1 se sustituye por el producto N_1 x N_2 en la expresión del coste.

En la Tabla 4.11 se presentan algunos resultados obtenidos realizando la descripción del circuito en Verilog HDL y la síntesis en Sinopsis.

	preprocesador	***P/S RAM***	***Sumador+ACC***	***postprocesador***	***DFT 4 puntos***	***Total***
Tiempo por columna	13,71 ns	12,45 ns	14,06 ns	17,7 ns	10,35 ns	68,27 ns
Camino crítico	17,7 ns	17,7 ns	17,7 ns	17,7 ns	17,7 ns	88,5 ns

Tabla 4.10 Camino crítico en el módulo básico del diseño basado en aritmética distribuida por bloques

Comparación

A fin de realizar la comparación entre el CBRM y la arquitectura BDA, se realiza una estimación del área ocupada comparando el hardware que interviene en cada circuito, a nivel de dispositivos utilizados (Tabla 4.12). En la Tabla 4.13 la comparación se realiza en términos de τ_t y τ_a. Para el CBRM, se asume que la LUT es una ROM de 64 celdas ya que los parámetros son tres (α, β, 1); los datos tienen una longitud de 16 bits en ambas propuestas.

N	*Dispositivos utilizados por la arquitectura BDA*	*Dispositivos utilizados por dos módulos CBRM calculando de forma cruzada*
16	5 buffers, procesador CORDIC, P/S-R, un rotador, 4 RAMs 4x16 bits, 16 MAC	4 MUX, 8 S-R, 4 MAC, 2 LUT 64 x16 bits 2 sumadores
64	5 buffers, procesador CORDIC, P/S-R, un rotador, 4 RAMs 16 x16 bits, 16 MAC	
512	9 buffers, procesador CORDIC, 2 P/S-R, un rotador, 8 RAMs 8x16 bits, 32 MAC, memoria de transposición	
4096	9 buffers, procesador CORDIC, 2 P/S-R, un rotador, 8 RAMs 16x16 bits, 32 MAC, memoria de transposición	

Tabla 4.11 Comparación del hardware necesario en las arquitecturas CBRM y BDA

En la Tabla 4.11 no se consideran ni el rotador ni el procesador CORDIC de la arquitectura BDA en el cálculo del área, ya que la referencia consultada no precisa la estructura de ninguno de ellos.

Para estimar el tiempo medio de computación, se utilizan las indicaciones del autor, asumiendo que los tiempos de acceso a la ROM, del sumador y del cerrojo son los que figuran en el capítulo 3.

Tiempo medio de computación: $(\frac{N_1}{4} x W_L)(T_{ROM} + 2T_{ADD} + T_{LATCH})$

N	*Arquitectura BDA*		*Arquitectura CBRM (dos módulos calculando de forma cruzada)*			
			Implementación1		*Implementación2*	
	Area	*Tiempo*	*Area*	*Tiempo*	*Area*	*Tiempo*
16	314 τ_a	3,3 $10^3\tau_t$	168 τ_a	3,38 $10^3\tau_t$	176 τ_a	1,48 10^3 τ_t
64	344 τ_a	13,2 $10^3\tau_t$		14,235 10^3 τ_t		12,44 $10^3\tau_t$
512	632 τ_a	105,6 $10^3\tau_t$		116,36 $10^3\tau_t$		101,36 $10^3\tau_t$
4096	672 τ_a	844,8 $10^3\tau_t$		921,515 $10^3\tau_t$		812,79 $10^3\tau_t$

Tabla 4.12 Comparación de las arquitecturas CBRM y BDA en términos de τ_a y τ_t

Conclusión

Aparece claramente que la arquitectura BDA es más costosa en cuanto a ocupación de área que la arquitectura CBRM. Para los valores de *N* considerados, los tiempos de la BDA son ligeramente inferiores a los tiempos de la implementación1 del CBRM, pero superiores a los de la implementación 2 de éste. Además, la implementación 2 es la que consigue menores aumentos de tiempo cuando aumenta *N*.

Comparación CBRM– otras propuestas

En la tabla 4.13 se reagrupan de forma resumida los costes hardware y de tiempo de diversas propuestas facilitadas en [Chien, 2002]. En todas ellas, *N* representa el número de puntos de la transformada. Las cuatro primeras referencias basan su diseño en matrices sistólicas, la quinta en sumadores y las demás en aritmética distribuida. La última es el CBRM. El CBRM es la mejor propuesta en cuanto al coste hardware; en cambio, el tiempo de cálculo es proporcional al número de

puntos y a la precisión y esto sugiere que para este caso es conveniente plantear la paralelización del CBRM.

	Memoria	*Sum.*	*Mult.*	*Reg. de desplaz.*	*Reg. P/S*	*CORDIC*	*Tiempo medio de cálculo*
Chang and Chen	0	N	N	$6N$	0	0	N x (2Tmult+2Tsum+Tcerrojo)
Fang and Wu	0	$2N$+6	N+4	$6N$	0	0	N x (2Tmult+2Tsum+Tcerrojo)
Murthy and Swamy	0	N	N	$10N$	0	0	N x (2Tmult+2Tsum+Tcerrojo)
Chan and Panchanathan	0	N	N	$8N$	0	0	N x (2Tmult+2Tsum+Tcerrojo)
Chang and Guo	$4N$- 4 (RAM)	$6N$+7	0	$4N$-2	0	0	$N/2$x (Tsum+Tcerrojo+Tsum.red)
Diseño DA	$\frac{N}{4}x2$ (ROM)	$\frac{N^2}{4}$	0	$5N$	N	0	W_L x (T_{ROM}+2Tsum+Tcerrojo)
Diseño BDA	$\frac{N}{4}x2$ (ROM)	$\frac{N}{4}$+4	0	$3N$	$\frac{N}{4}$	$\frac{N}{4}+4$	NxW_L/4 x (T_{ROM}+2Tsum+Tcerrojo)
CBRM	2x W_Lx 2^3 (ROM)	2+2	0	2	0	0	($3N/2$-2)x $W_L T_{ROM}$ + (N-1) W_Lx Tsum

Tabla 4.13 Comparación de la arquitecturas CBRM con otras propuestas

5 Aplicación del CBRM a otras transformadas

La transformada de Fourier puede utilizarse para establecer un patrón de cálculo para otras transformadas. Para ello, se realizan pequeñas transformaciones matemáticas que se exponen a continuación.

- Transformada de Hartley

Sea *H(u)* la transformada de Hartley discreta (HDT) de una función real *f(x)*.

$$H(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)(\cos\frac{2\pi ux}{N} + sen\frac{2\pi ux}{N}) \qquad (4.22)$$

H(u) es una función real cuyo cálculo puede partirse en dos fragmentos, el que corresponde al seno y el que corresponde al coseno, que se sumarán al final. El desarrollo es idéntico al de la DFT, añadiendo la etapa final que es la suma de los fragmentos calculados. Llamando *R(u)* la parte del sumatorio que incluye al coseno e *I(u)* la parte que incluye al seno:

$H(u) = R(u) + I(u)$

- Transformada del coseno

Sea la transformada discreta del coseno (DCT) de una función real *f(x)*.

$$C(u) = e(k)\sum_{x=0}^{N-1} f(x)\cos(2x+1)\frac{\pi u}{2N} \qquad (4.23)$$

C(u) es una función real cuyo cálculo puede partirse en dos fragmentos y recomponerse al final como una suma ponderada de éstos. Los cálculos se organizan de la manerasiguiente:

$$f(x)\cos(2x+1)\frac{\pi u}{2N} = f(x)\cos(\frac{\pi ux}{N}+\frac{\pi u}{2N})$$

Desarrollando el coseno de la suma se obtiene:

$$f(x)(\cos\frac{\pi ux}{N}\cos\frac{\pi u}{2N} - sen\frac{\pi ux}{N}sen\frac{\pi u}{2N})$$

entonces la expresión de *C(u)* es

$$C(u) = e(k)\sum_{x=0}^{N-1} f(x)(\cos\frac{\pi ux}{N}\cos\frac{\pi u}{2N} - sen\frac{\pi ux}{N}sen\frac{\pi u}{2N})$$

Para cada valor de *u* las cantidades *cos[πu/2N]* y *sen[πu/2N]* permanecen constantes y pueden salir fuera del sumatorio

Si $\cos\frac{\pi u}{2N} = \alpha_u, sen\frac{\pi u}{2N} = \beta_u$

$$C(u) = e(k)(\alpha_u\sum_{x=0}^{N-1} f(x)\cos\frac{\pi ux}{N} + \beta_u\sum_{x=0}^{N-1} f(x)sen\frac{\pi ux}{N})$$

Los fragmentos definidos pueden calcularse siguiendo el planteamiento de la DFT, llamando *R(u)* e *I(u)* a los sumatorios que corresponden al coseno y al seno, respectivamente. El resultado final para cada valor de *u* se obtiene mediante la operación de combinación:

$$C(u) = \alpha_u R(u) + \beta_u I(u)$$

En resumen, el CBRM se aplica fácilmente a la evaluación de las transformadas de la imagen que poseen núcleo trigonométrico, utilizando un patrón único de organización de los cálculos.

Para transformadas que no poseen núcleo trigonométrico, el cálculo puede inspirarse en el de la DFT. Como ejemplo, se proponreel cálculo de la transformada de Walsh.

- Transformada de Walsh

La transformada de Walsh de una función real *f(x)* unidimensional se expresa de la manera siguiente:

$$W(u) = \sum_{x=0}^{N-1} f(x) \frac{1}{N} \prod_{i=0}^{n-1} (-1)^{bi(x)bn-i-1(u)} \qquad (4.24)$$

Al igual que la DFT, esta transformada puede evaluarse ordenando el cálculo a partir del modelo de doblamientos sucesivos y aplicando posteriormente el CBRM con la modificación correspondiente, que consiste en igualar a 0 los parámetros β y a 1 los parámetros α de las ecuaciones (4.14)-(4.16), de manera que los valores de *W(u)* resultan ser sumas algebraicas de las muestras *f(x)*.

Se trataría de manera idéntica la transformada de Hadamard.

Volver al índice/Tornar a l'índex

6 Conclusión

El presente capítulo se ha dedicado enteramente a la validación del modelo CBRM, objeto de estudio de este trabajo de investigación. Para ello, se ha procedido a acoplar el CBRM al cálculo de las rotaciones, consideradas éstas como paradigmáticas, por el alto nivel de derivación que precisan en las arquitecturas habituales así como por su presencia en áreas importantes de la ciencia y la ingeniería. En términos prácticos se evalúa la bondad del modelo CBRM comparándolo con otras propuestas en el cálculo de las transformadas de Hough y Fourier, que se basan en rotaciones.

En el caso de la transformada de Hough, el CBRM demuestra su bondad frente al CORDIC, en sus versiones serie y paralela confirmando su mejor comportamiento en el ahorro de área y hardware así como en la velocidad.

En el caso de la transformada de Fourier, se han presentado esencialmente dos propuestas frente a las cuales el CBRM consigue mejores prestaciones.
En una comparación de corte más general efectuada al final del capítulo, aparece que el comportamiento del CBRM es muy bueno en cuanto a relación área-tiempo, y ello lo hace ser un buen candidato para el desarrollo de dispositivos VLSI. El tiempo de cálculo proporcional al número de puntos y a la precisión puede mejorar sensiblemente mediante algún tipo de paralelización, a cambio de un aumento de área ocupada. Hay que destacar que el ahorro de área se debe sobre todo a la memoria que sólo es proporcional al tamaño de los datos.

Capítulo 5

SIMULACIÓN DE COMPORTAMIENTO DIFÍCILMENTE FORMALIZABLE

1 Introducción

Entendemos que un comportamiento es difícilmente formalizable cuando resulta difícil captar su esencia por medio del utillaje matemático del que se dispone habitualmente. Bajo este epígrafe caben los procesos artificiales no suficientemente especificados y, sobre todo, los procesos naturales, relacionados o no con la vida.

Tradicionalmente, la biología ha recurrido a modelos biofísicos para explicar el comportamiento del tejido vivo, con la limitación de tener que relacionar cualquier parámetro del modelo con su equivalente físico [Hodgkin, 1952]. A la hora de remedar las propiedades colectivas de un sistema biológico, las redes neuronales artificiales sustituyen el modelo biofísico, superando así el inconveniente de la falta de genericidad inherente al modelado biofísico. Sin embargo, a pesar de las numerosas analogías que se encuentran entre las redes y los procesos computacionales propios de la biología, existe en general una gran dificultad para establecer relaciones entre el modelo y los resultados experimentales obtenidos. Se tiene conocimiento de ciertas similitudes muy

localizadas, por ejemplo, entre las memorias asociativas y el hipocampo, sede del aprendizaje [O´Keefe, 1979], también con el olfato [Gelperin, 1989] y el procesamiento visual [Koch, 1985]. A pesar de ello, estos modelos no han sido corroborados desde la neurofisiología. Desde un planteamiento inverso, la computación artificial ha considerado en los últimos tiempos la conveniencia de copiar del mundo vivo algunas formas de procesar la información. Todo ello ha dado lugar a una vía de aportación bilateral que ha cristalizado en una enorme cantidad de conocimiento.

La propuesta de un enfoque CBRM para simular el comportamiento de sistemas con estas características persigue el objetivo de una resolución desde la perspectiva funcional. Esta vía es muy atractiva por su capacidad para reproducir los comportamientos sofisticados de un sistema sin necesidad de conocer los detalles estructurales que los provocan. Así, un módulo CBRM puede representar una neurona, un conjunto de neuronas o una red de conjuntos de neuronas, ya que no hay necesidad de establecer correspondencias entre estructuras naturales y artificiales. Más concretamente, el CBRM trata el comportamiento difícil de formalizar como una función no abordable a nivel de primitivas sino con un mayor nivel de derivación.

En este capítulo se aborda de manera empírica la evaluación de funciones a niveles más altos que el nivel de las primitivas, como introducción a lo que será un estudio en profundidad que se deja como línea de trabajo futura. Se presentan dos ejemplos pertenecientes al campo de la neurociencia. Primero, se considera una simplificación bottom-up del modelo de cable de Traub [Traub 1991], que presenta un modelo bicompartimental de la neurona (soma y dendritas). La estimulación de cada compartimento bajo diferentes condiciones experimentales produce patrones de comportamiento diferenciados. Se pone en evidencia la capacidad del CBRM para reproducir fielmente el espigueo obtenido por la simulación de las ecuaciones del modelo biofísico simplificado en la zona CA3 del hipocampo, sede del aprendizaje. En segundo lugar, se

considerarán algunos registros fisiológicos realizados sobre una preparación de tejido nervioso del molusco *Tritonia diomedea* presentando el circuito eléctrico de implementación y comparando los resultados obtenidos por las simulaciones de la memoria asociativa y del CBRM.

2 Aplicación del CBRM a los procesos computacionales biológicos

En este apartado, los procesos computacionales biológicos considerados son los que se tienen lugar en el sistema nervioso.

En 1894 fueron establecidos por S. Ramón y Cajal los primeros principios de la neurociencia. Sobre estos principios se han construido los distintos modelos propuestos, desde el modelo más simple en el que una neurona equivale a un punto (soma) hasta los modelos más completos que hacen intervenir todas las demás estructuras nerviosas (axón, árbol dendrítico,...). Los registros fisiológicos obtenidos por medición directa en el tejido nervioso son funciones de difícil formalización matemática, como evidencian los circuitos electrónicos necesarios para simularlas con todo el detalle, por tanto, son buenos candidatos para la simulación por el CBRM. Los dos ejemplos que se van a tratar en este apartado han sido resueltos previamente por modelo biofísico y por red asociativa, respectivamente. Después de facilitar en cada caso una breve descripción de las características del fenómeno estudiado así como del fundamento del modelo utilizado, se aborda la resolución desde el CBRM comparando posteriormente los resultados.

2.1 Generación de brotes inductores de espigas en la región CA3 del hipocampo

Descripción y simulación del modelo biofísico

La región CA3 del hipocampo genera espigueo sincronizado en condiciones experimentales diversas, habiéndose reducido previamente la inhibición. Existe gran cantidad de estudios experimentales basados en registros fisiológicos (Chamberlin et al.1990), (Miles et al, 1988) y corroborados por modelos matemáticos, de las propiedades intrínsecas de las neuronas de la región CA3 (Traub and Miles, 1991), (Traub et al. 1993). El modelo de Traub, o modelo de cable compartimental, establece una partición en 19 compartimentos de una célula piramidal de CA3. Cada compartimento contiene hasta seis conductancias iónicas controladas por diez variables que representan el cierre o la apertura de canales. La dinámica de estas variables se construye en coherencia con los datos experimentales obtenidos en fisiología. El modelo de Traub, enormemente complejo, ha sido simplificado por Pinsky y Rinzel, (Pinky y Rinzel, 1994), quedando en un modelo de dos compartimentos (soma y dendritas) con ocho variables, que tiene la ventaja de poner en evidencia los aspectos esenciales del modelo de Traub, con un manejo más fácil de la dinámica celular y global, para rangos amplios de valores de los parámetros. Este modelo es el que se analiza aquí. El modelo simplificado segrega las corrientes más rápidas de las más lentas, situando las primeras en el soma y las segundas en las dendritas, a la vez que describe tres comportamientos prototípicos como respuesta a una estimulación somática o dendrítica. Se trata, en todos los casos, de la producción de espigueos, de baja (8 a 20 Hz) o muy baja frecuencia (< 8 Hz). En este modelo las espigas sólo aparecen en un rango intermedio de conductancias de acoplamiento (estas conductancias van ligadas al NMDA, *N-metil D-aspartato* y al AMPA), cuyos extremos son una baja conductancia, responsable de desacoplar los

compartimentos o una alta conductancia, que significa que el soma y las dendritas están fuertemente acoplados, resultando un compartimento único. Las corrientes que generan el espigueo son posibles sólo para valores moderados del acoplamiento electrotónico.

La explicación fisiológica subyacente al fenómeno del espigueo es la siguiente: en este sistema, el brote siempre viene inducido por un pico de sodio que procede del soma. Cuando el acoplamiento entre compartimentos es moderado, este pico repolariza parcialmente las dendritas retardando así el pico de calcio dendrítico. Esta combinación constituye el brote. En el mismo trabajo, Pinsky y Rinzel abordan el estudio de una red de 100 neuronas de tipo bicompartimental, a fin de examinar los fenómenos de sincronización y de saturación. La estimulación breve de una sola célula del conjunto produce un brote sincronizado, siendo la sinapsis del AMPA el mecanismo dominante de sincronización de las espigas. El número de brotes aumenta con el nivel de NMDA, llegando al espigueo indefinido para un nivel suficientemente alto de la conductancia NMDA. Existen dos factores capaces de desincronizar las células cuando las sinapsis de AMPA están bloqueadas que son, por una parte, la heterogeneidad de las propiedades de las células y por otra, la dinámica intrínseca de brotes caóticos que las células presentan.

La figura 5.1 representa un esquema del modelo bicompartimental de Pinsky y Rinzel, en el cual se pueden observar las corrientes aplicadas y las corrientes activas que entran y salen del soma y las dendritas.

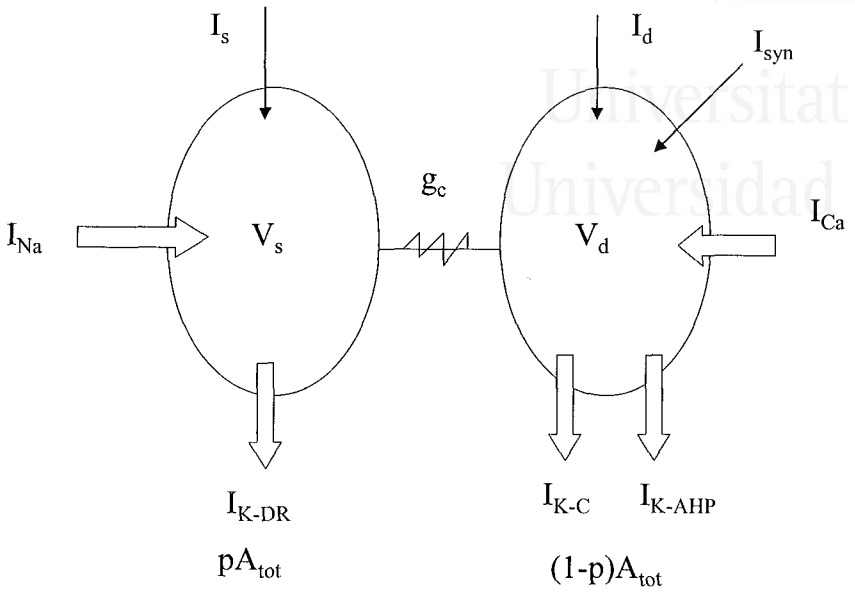

Fig 5.1 Modelo bicompartimental de Pinsky y Rinzel

- Corrientes iónicas:

El primer compartimento (soma) cuenta con dos corrientes dependientes de voltaje que generan picos rápidos de sodio, I_{Na} corriente entrante y una corriente saliente rectificadora inducida por potasio $I_{K\text{-}DR}$.

El segundo compartimento (dendritas) tiene una corriente entrante de calcio I_{Ca}, existen dos corrientes de potasio: $I_{K\text{-}C}$ y $I_{K\text{-}AHP}$ (posthiperpolarización).

- Corriente sináptica:

$I_{syn} = I_{NMDA} + I_{AMPA}$

- Corrientes de electrodo

I_s, I_d son aplicadas al compartimento-soma y al compartimento-dendrita, respectivamente.

El acoplamiento entre soma y dendritas viene gobernado por el parámetro g_c.

A_{tot} se refiere a la totalidad de la membrana de la célula, normalizada a 1.

El modelo biofísico se basa en las ecuaciones diferenciales siguientes, que expresan el balance de las corrientes entre los dos compartimentos:

$$
\begin{aligned}
&C_m V_s' = \\
&-I_{Leak}(V_s) - I_{Na}(V_s, h) - I_{K\text{-}DR}(V_s, n) + (g_c/p)(V_d - V_s) + I_s/p \\
&C_m V_d' = \\
&-I_{Leak}(V_d) - I_{Ca}(V_d, h) - I_{K\text{-}AHP}(V_d, q) - I_{K\text{-}C}(V_d, Ca, c) \\
&- I_{Syn}/(1\text{-}p) + (g_c/(1\text{-}p))(V_s - V_d) + I_d/(1-p)
\end{aligned}
\qquad (5.1)
$$

V_s', V_d' son las variaciones de V_s, V_d respectivamente

Ca concentración de calcio

n activación rápida

q activación lenta variable dependiente de la concentración de Ca

c activación rápida < 6 ms

s activación rápida < 6 ms

h inactivación rápida

p proporción de área ocupada por el soma

Cm capacidad del cable

En las figuras 5.2 a-e se muestra la generación de distintos comportamientos de tipo espiga según los valores de los tres parámetros $I_s(\mu A/cm^2)$, $g_{NMDA}(mS/cm^2)$, $g_c(mS/cm^2)$. La escala vertical que aparece en la figura 5.2a equivale a 40 mV, 200 unidades de Ca para cualquier gráfica; la escala horizontal que sólo aparece en la figura 5.2e representa 400 ms para la figura 5.2 a y 200 ms para las demás.

Todas las simulaciones han sido realizadas escribiendo un programa en FORTRAN, ejecutado en una estación de trabajo IBM RS6000 RISC. Las ecuaciones diferenciales (5.1) se resuelven utilizando el método de Runge-Kutta con un paso de iteración de 0,05 ms. El modelo de neurona presentado cuenta con 8 variables (frente a las 120 del modelo de Traub) y la relación del tiempo de ejecución entre los dos modelos es de 0,09.

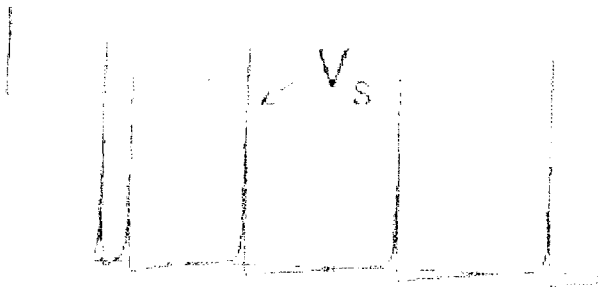

Fig. 5.2 a Brote de muy baja frecuencia inducido por activación somática (0,75-0,0-2,1)

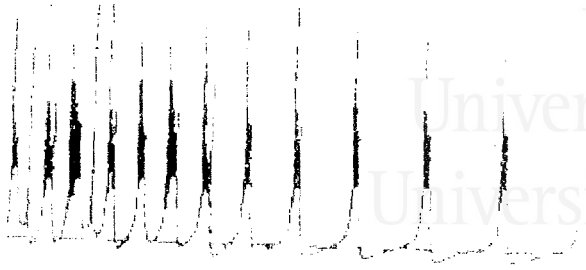

Fig. 5.2 b Brote de baja frecuencia inducido por activación dendrítica (-0,5-1,25-2,1)

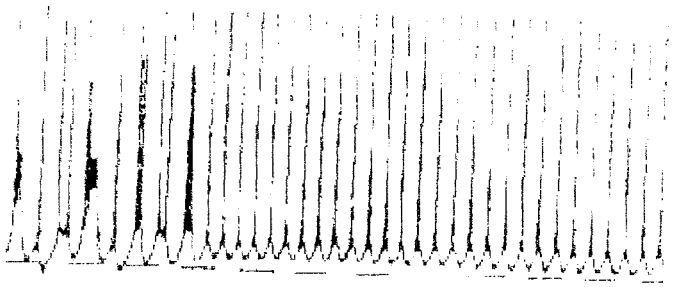

Fig 5.2 c Picos de alta frecuencia en el soma con I_s mayor que en 5.2a (2,5-0,0-2,1)

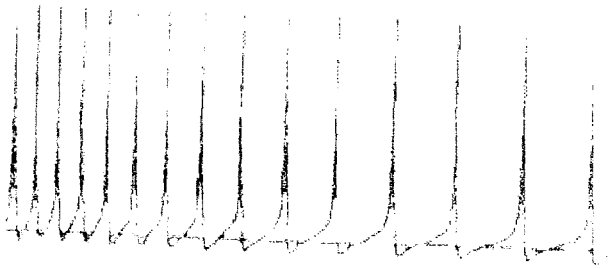

Fig 5.2d Idénticas condiciones a las de 5.2c pero con mayor acoplamiento (2,5-0,0-10,5)

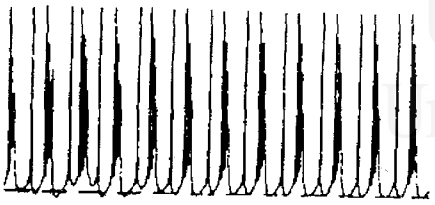

Fig 5.2 e Estimulación dendrítica y acoplamiento bajos (-0,5-1,75-1,425) produce espigueo complejo formado por picos seguidos de brotes.

Simulación CBRM

Las Figuras 5.3 a-e muestran los resultados obtenidos mediante simulación por el CBRM. En el eje de abscisas, se representa el número de la iteración en curso y el valor de las ordenadas depende del valor arbitrario del punto inicial, que habitualmente se toma igual a 1. La búsqueda de los valores de los parámetros que caracterizan la primitiva utilizada en cada caso desborda el alcance de este trabajo de investigación, quedando la presentación de los ejemplos como preámbulo de una de las líneas de investigación futuras.

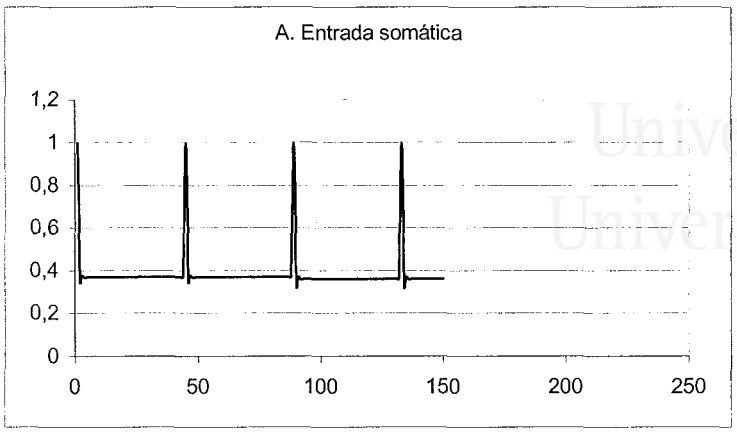

Fig 5.3 a
Simulación por el CBRM del registro de la Figura 5.2 a

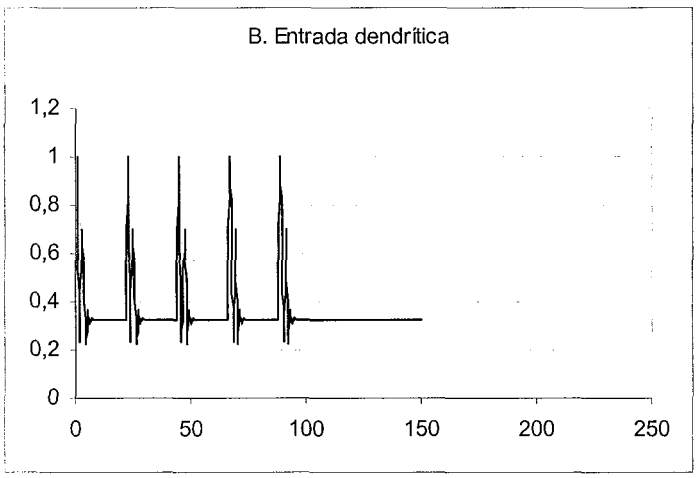

Fig 5.3 b
Simulación por el CBRM del registro de la Figura 5.2 b

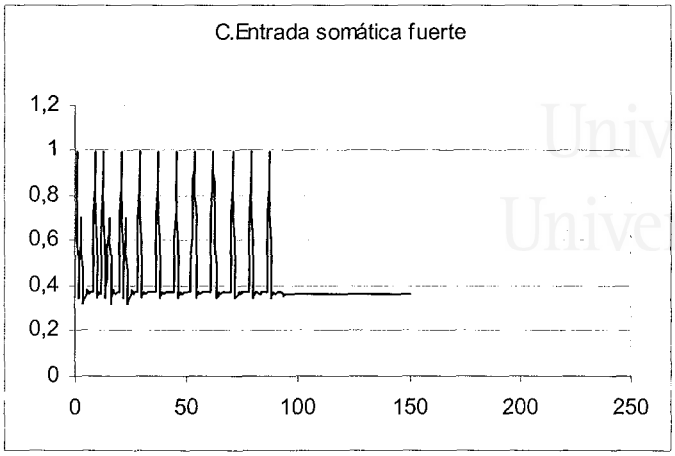

Fig 5.3 c
Simulación por el CBRM del registro de la Figura 5.2 c

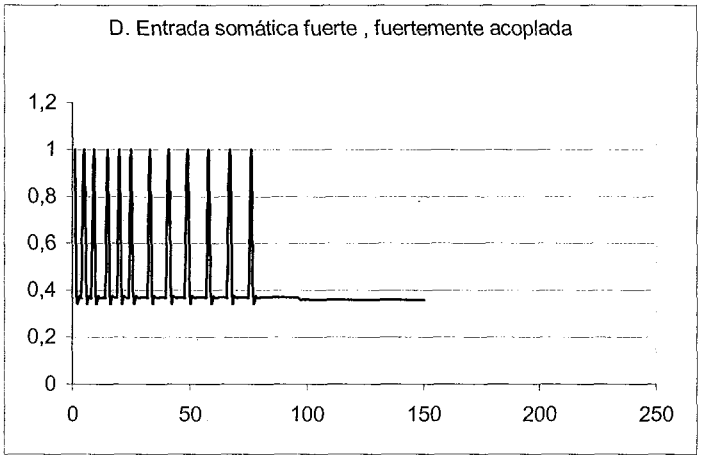

Fig 5.3d
Simulación por el CBRM del registro de la Figura 5.2 d

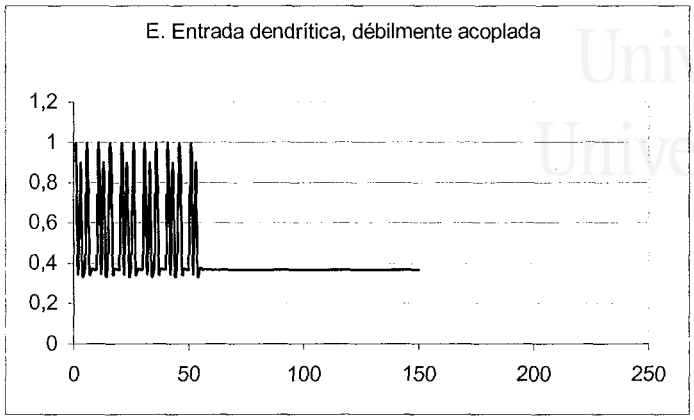

Fig 5.3e
Simulación por el CBRM del registro de la Figura 5.2 e

2.2 Generadores centrales de patrones neuronales (CPGs)

Descripción y simulación por memoria de Hopfield

Los CPGs son redes nerviosas que controlan los músculos involucrados en la ejecución de comportamientos rítmicos bien definidos, como respirar, andar, nadar, masticar. Algunas de estas redes están anatómicamente localizadas, contienen un número reducido de neuronas y su salida consiste en patrones de oscilaciones coherentes. Estas características hacen de los CPGs buenos candidatos para el estudio de la relación entre las propiedades de salida de una red biológica y la circuitería asociada.

Existe una amplia gama de estudios realizados sobre los comportamientos rítmicos que ponen de manifiesto algunos principios generales de funcionamiento:

- Una salida rítmica puede aparecer incluso en ausencia de retroalimentación sensorial de los músculos y estructuras controlados por el CPG y en ausencia de control por parte de los centros nerviosos de más alto nivel (Grillner, 1975).
- Algunos CPGs funcionan sin maracapasos, que es una neurona cuyos disparos determinan el período de salida de la red. Un ejemplo de este caso es el CPG que controla el movimiento natatorio de *Tritonia Diomedea,* que se considera en este estudio.
- El mismo conjunto de neuronas motoras puede estar involucrado en una gran variedad de comportamientos rítmicos en un animal y ello sugiere que el mismo CPG es capaz de producir distintos patrones de comportamientos rítmicos.
- La salida de los CPGs puede ser modulada por entradas externas como son la retroalimentación de los propioceptores y de los centros nerviosos de más alto nivel.

Se presenta en este estudio la simulación del CPG que controla el movimiento natatorio del molusco *Tritonia diomedea* por una memoria asociativa de tipo Hopfield.

El modelo consta de N neuronas interconectadas. La salida de cada neurona $V_i(t)$ varía entre cero (estado de reposo) y uno (estado de máxima actividad). El estado de la red viene especificado por la actividad de salida de todas sus neuronas y un patrón de comportamiento se define por una secuencia temporal en la que aparece un subconjunto de todos los estados de salida existentes.

Sin entrar en la descripción detallada del ritmo natatorio de *Tritonia diomedea*, cabe decir que su CPG consta de cuatro grupos de neuronas, VSI-A, VSI-B, C2 y DSI. Las neuronas VSI son las que accionan las interneuronas de la zona ventral;

C2 es la neurona cerebral y DSI representa la interneurona dorsal. El patrón de salida se compone de los brotes en VSI-A y VSI-B alternando con los brotes de las neuronas de C2 y DSI. Estas configuraciones representan los estados $V_1(t)$, $V_2(t)$, $V_3(t)$ y $V_4(t)$ respectivamente. La Fig 5.4 muestra estos estados que aparecen cada vez más desfasados hasta su desaparición.

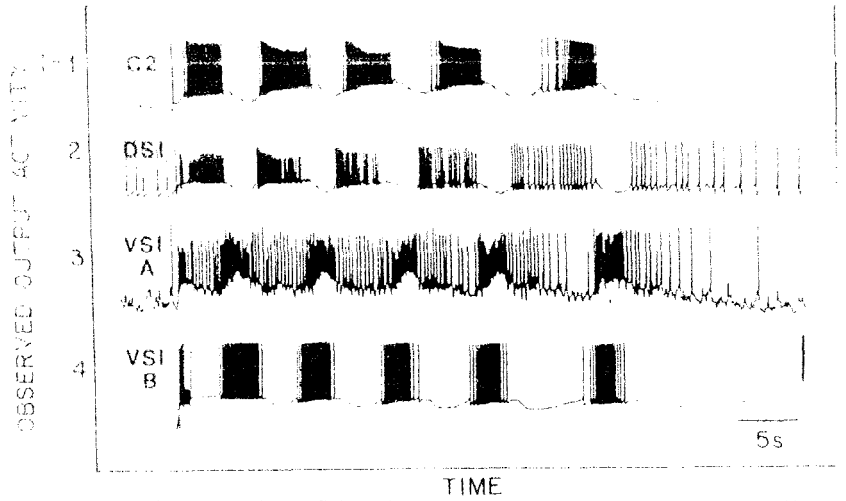

Fig 5.4 Registros fisiológicos de los estados $V_1(t)$, $V_2(t)$, $V_3(t)$ y $V_4(t)$

La simulación por una memoria de Hopfield del comportamiento del CPG no lo reproduce exactamente, como se observa en la Fig 5.5. Sin embargo, el circuito electrónico de simulación tiene cierta complejidad (ver Fig 5.6). En él, las neuronas se representan por amplificadores saturados, las conductancias representan las conexiones entre pares de neuronas y se indica el peso sináptico en cada una de ellas.

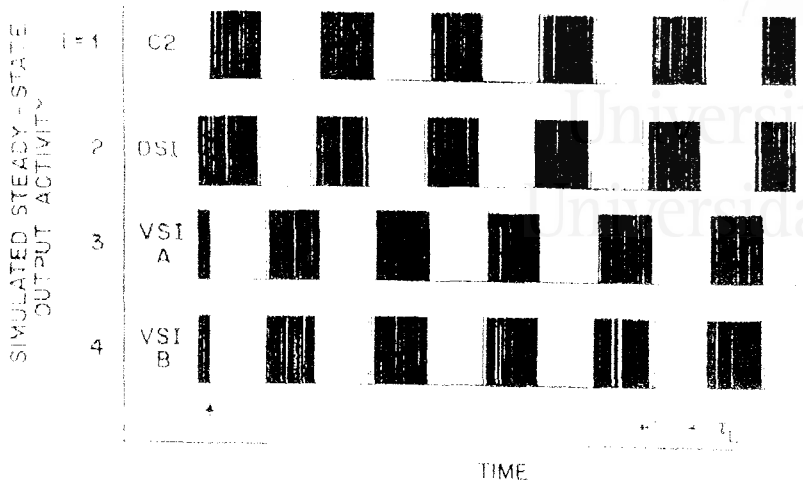

Fig 5.5 Simulación de los estados $V_1(t)$, $V_2(t)$, $V_3(t)$ y $V_4(t)$ del CPG por una memoria de Hopfield

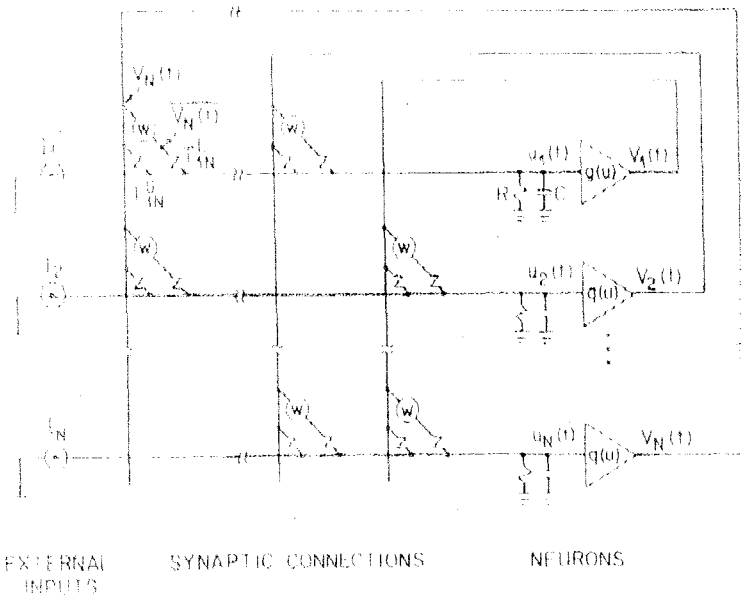

Fig 5.6 Circuito que implementa el CPG modelado por la memoria de Hopfield

Simulación CBRM

La aportación del CBRM a la simulación del CPG se refleja en las figuras 5.7 a-d. Como en el ejemplo anterior, en el eje de abscisas se representa el número de la iteración en curso y el valor de las ordenadas depende del valor dado arbitrariamente al punto inicial, que habitualmente es 1.

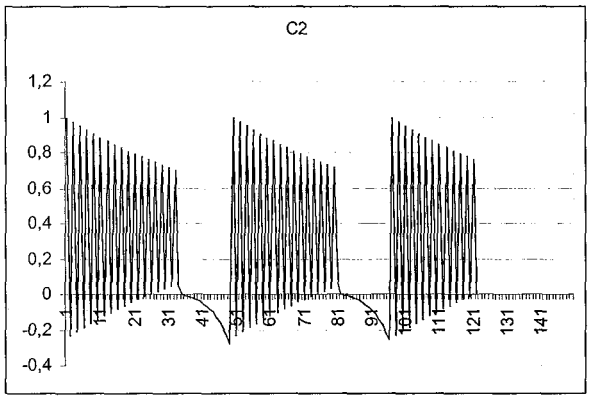

Fig 5.7 a Simulación por el CBRM de V_1 (t)

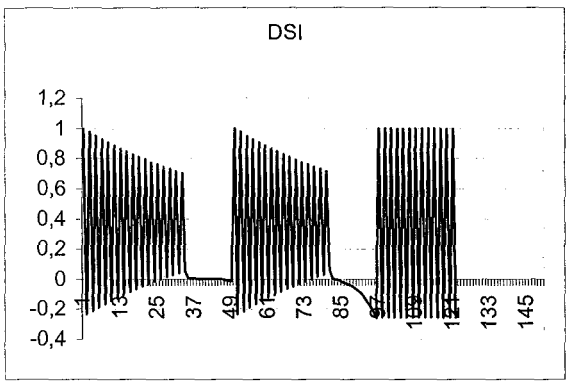

Fig 5.7 b Simulación por el CBRM de V_2(t)

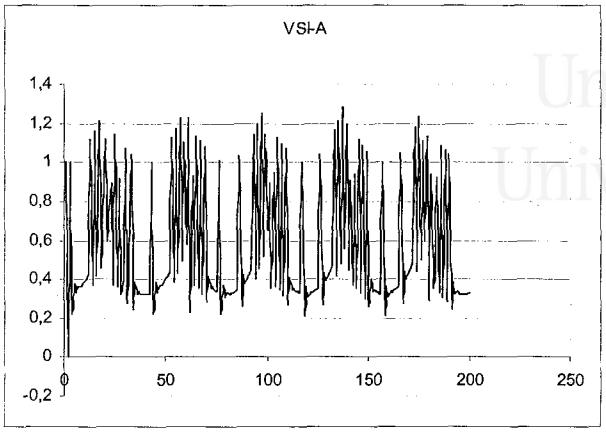

Fig 5.7 c Simulación por el CBRM de $V_3(t)$

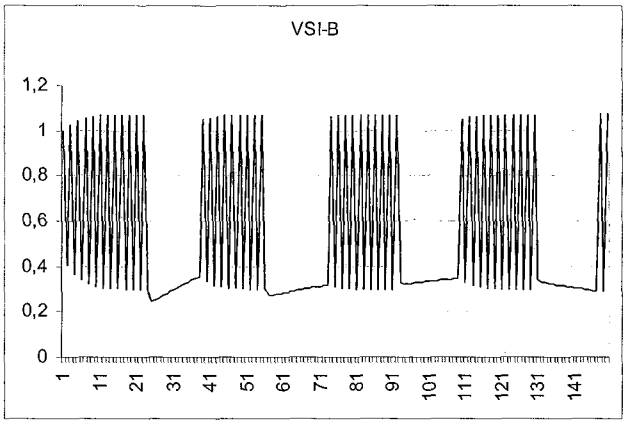

Fig 5.7 d Simulación por el CBRM de $V_4(t)$

3 Conclusión

El presente capítulo contribuye a la validación experimental del CBRM en el ámbito de lo que se ha denominado comportamiento difícilmente formalizable. A diferencia del anterior, este capítulo aborda empíricamente la evaluación de funciones, ya no a nivel de primitivas, sino a niveles superiores de derivación. A

Volver al índice/Tornar a l'índex

partir de los resultados alentadores obtenidos en los casos considerados, el CBRM aparece como una aportación interesante en casos en los que el único conocimiento que se tiene de un comportamiento es una colección de puntos. Se pone de manifiesto que el CBRM es capaz de remedar con apreciable fidelidad el comportamiento de sistemas sin necesidad de conocer el detalle fino de su estructura. Los subsistemas neuronales son un ejemplo de ello. La sistematización de esta cuestión queda como línea abierta de investigación.

Capítulo 6

CONCLUSIONES

1 Aportaciones

Esta memoria recoge los resultados de una investigación orientada a la mejora de prestaciones de los computadores. La aportación esencial consiste en utilizar técnicas iterativas de cálculo del sucesor para poder obtener valores de funciones al nivel de operación de las primitivas del procesador o mediante muy pocos niveles de procesamiento. Se pueden evaluar así funciones que requieren habitualmente niveles muy elevados de derivación, cuando el procesamiento utilizado es el convencional, basado en las técnicas de cálculo polinómico que resultan de considerar las estructuras algebraicas de los conjuntos de los números.

El camino seguido por la investigación se inicia con una revisión de la operatoria habitual efectuada por los computadores, desde la perspectiva de la aritmética del procesador, centrada fundamentalmente en las primitivas suma y multiplicación, así como en los distintos algoritmos empleados en el cálculo de funciones. Asimismo se ha extendido la revisión a algunas primitivas que centran la computación en procesadores especializados, en ramas de la informática como la computación gráfica y el análisis de imagen. Todas estas consideraciones han puesto de manifiesto que los esfuerzos de la investigación que se realiza para mejorar el rendimiento en la computación se vienen desarrollando según dos vías:

por una parte, la búsqueda de primitivas nuevas para tratar casos particulares concretos y, por otra, la mejora en el rendimiento de las primitivas habituales con la finalidad de extender esta mejora a cualquier cálculo derivado.

La convolución de dos funciones expresa el resultado de evaluar una de ellas tomando a la otra como referencia. A partir de esa noción se ha obtenido una expresión recursiva de la función de convolución que permite calcular el valor siguiente mediante una suma ponderada; esto es, mediante la suma de dos multiplicaciones

Los factores de ponderación utilizados en el proceso iterativo adquieren el papel de caracterización de la función resultante lo cual, grosso modo, puede interpretarse como que sus valores constituyen una expresión condensada de la lógica relacional algebraica que se utiliza en la notación explícita de la expresión de las funciones.

Precisamente, es el establecimiento de los valores de los factores de ponderación el objetivo que se encomienda a la unidad de control que se convierte, de esta manera, en una estructura realmente sencilla cuyo cometido, además de cargar los valores de ponderación, es iniciar las operaciones y gobernar la iteración.

Desde el punto de vista de la algoritmia que obtiene los cálculos, el método consiste en obtener un nuevo valor de la función en cada paso de la iteración a partir del valor de la función obtenido en el paso de iteración precedente. Esto significa que se necesita establecer un valor de iniciación de la iteración y un paso de iteración. El ámbito de aplicación se verá potenciado por la necesidad de obtener colecciones de valores de las funciones como, como por ejemplo, muestreos temporales, barridos o representaciones de comportamientos en intervalos.

Los criterios de utilización del paso de iteración abren, por su parte, la vía para la incorporación, de manera intrínseca, de paralización de la operatoria. Se puede, pues, operar en el extremo de la secuencialidad estricta, con el paso básico, el más grande que permita la granularidad del problema y calcular así un valor en cada paso de iteración partiendo del valor obtenido en la iteración anterior. Utilizar un paso de iteración múltiplo del básico permite obtener valores semilla de iniciación a partir de los cuales se pueden lanzar hilos paralelos de cálculo. La paralelización puede extenderse a varios niveles de profundidad, formando un árbol. En esto puede haber una clave para derivar funciones más complejas.

Seguidamente, se resume la aportación realizada por esta tesis a lo largo de sus seis capítulos.

En el capítulo primero, de introducción, un breve repaso de la historia del procesamiento de información pone de manifiesto que esta cuestión ha estado presente, al menos de forma explícita, desde la Antigüedad. La emergencia de la informática y su posterior desarrollo que se debe, tanto al refinamiento de los métodos como a la mejora constante de las realizaciones, persigue un objetivo primordial: resolver cada vez mejor lo que se aborda. Esta tesis pretende aportar su contribución a este objetivo. Después de realizar una revisión del estado del conocimiento actual en torno a las cuestiones consideradas de interés, la propuesta de esta investigación, en aras a la mejora de prestaciones, va por la línea de disminuir el nivel de derivación, propio de los procesadores convencionales, en el cálculo de funciones. Ello conlleva una propuesta de primitivas nuevas que definen un procesador.

El capítulo dos presenta el método de evaluación recursiva basado en convolución, CBRM. Este método se fundamenta sobre dos ideas. Primero, que la convolución de dos funciones expresa el resultado de evaluar una de ellas

tomando a la otra como referencia. Segundo, que una operatoria distinta de la habitual puede disminuir el nivel de derivación que precisa el cálculo de convoluciones. Después de justificar teóricamente el papel que desempeña la convolución, se desarrolla una operatoria recursiva que calcula las convoluciones a nivel de primitivas, iterando una forma paramétrica. Ésta encierra, en el valor de sus parámetros, las características de las funciones que intervienen en la convolución. Los intervalos de pertenencia de los parámetros definen comportamientos diferenciados, poniendo de manifiesto la potencia del método.

El capítulo tres se dedica a la presentación y evaluación de la arquitectura del procesador CBRM. El funcionamiento necesita establecer un valor de iniciación de la iteración y fijar el valor del paso de iteración. Se realizan las mediciones pertinentes de tiempo de cálculo y área del módulo de cálculo del procesador CBRM. Los criterios de utilización del paso de iteración incorporan, de manera intrínseca, la paralelización de la operatoria. En efecto, el paso básico, determinado por la granularidad del problema, permite operar en el extremo de la secuencialidad estricta, y calcular así un valor en cada paso de iteración partiendo del valor obtenido en la iteración anterior. Si se utiliza un paso múltiplo del básico se obtienen valores que inician hilos paralelos de cálculo. La paralelización puede extenderse a varios niveles de profundidad, formando un árbol. Se efectúan mediciones de eficiencia, productividad y ganancia en velocidad, dependiendo de la distribución de los cálculos en los hilos paralelos.

El capítulo cuatro corrobora la validez del CBRM exponiendo su aplicación en el caso de algunas transformadas muy utilizadas en muchos y muy diversos ámbitos. Las transformadas de Hough y el conjunto de transformadas ortogonales, encabezado por la transformada de Fourier, constituyen un banco de pruebas interesante. El estudio y comparación que se realiza de la aplicación del CBRM frente a otras propuestas existentes aporta resultados alentadores, sobre todo en

cuanto a los tiempos de cálculo. Además, todas las transformadas consideradas obedecen a un mismo patrón de cálculo tratable por el CBRM. Ello induce a pensar que este método pueda ir en la línea de la genericidad y la robustez.

El capítulo cinco presenta, bajo el epígrafe de "Simulación de comportamiento difícilmente formalizable" dos ejemplos de funciones no tratables a nivel de primitivas por el CBRM. Se anticipa para ellos una solución empírica a modo de introducción de lo que será el desarrollo posterior del método en cuanto al problema de los niveles de derivación, parte que queda como línea de trabajo futuro.

El capítulo seis, último de este documento, repasa las aportaciones realizadas por esta investigación y propone unas líneas de trabajo futuro.

2 Líneas futuras

A partir de los hitos alcanzados por esta investigación, el trabajo puede seguir varios caminos.

Con respecto a seguir consolidando la fundamentación teórica de la metodología se prevén posibilidades de desarrollo importantes en torno a las cuestiones siguientes:

- La definición de los niveles de derivación es necesaria para completar la formalización del método. Las funciones usuales tratadas en esta tesis son resueltas a nivel de primitivas. Las funciones calificadas como difíciles de formalizar, de las cuales el capítulo quinto muestra algunos ejemplos, constituyen casos en los que han de intervenir más niveles de derivación.

- Los parámetros α y β así como la función auxiliar G de la formulación recursiva empleada definen tendencias comportamentales que caracterizan la función F calculada. Falta avanzar en la sistematización de la correspondencia entre los valores de los parámetros y los comportamientos.
- El resultado alentador del CBRM en la aplicación a las transformadas de Hough y transformadas ortogonales deberá orientar los esfuerzas en la dirección de estudiar su aplicabilidad a otos grupos de funciones.
- El uso de tablas es una opción de implementación reemplazable por cualquier otra. Sin embargo, existe la posibilidad de considerar como línea teórica de investigación en aritmética el hecho de descomponer y encerrar en un número determinado de casillas de una tabla lo esencial de una función, que es su comportamiento, materializado por un número fijo de combinaciones de los parámetros. La evaluación en sí consistirá siempre en un proceso muy sencillo de cálculo, como es la extracción y suma de los contenidos de la tabla.

La investigación en la parte de arquitectura de computadores puede proseguir en vistas a proponer innovaciones en cuanto a la esencia del cálculo de funciones así como a mejorar las prestaciones en rapidez de cálculo, ahorro de espacio. Para ello, se sugieren las siguientes líneas de desarrollo:

- Las tablas LUT son memorias ROM y, por tanto, son dispositivos que evolucionan con la tecnología. Así y todo, es interesante estudiar estrategias para reducir su tamaño de forma que se superen las dificultades del aumento del tamaño al aumentar la precisión de los datos.
- Interesa sin duda pensar en la adecuación del CBRM para tiempo real. La idea es probablemente proseguir en la línea de particionar los datos y realizar

las operaciones de extracción y suma según convenga en función del tiempo disponible. Las restricciones de tiempo acarrearán seguramente mayor error.

Referencias

[Aharonov, 1997]
D.Aharonov y M. Ben-Or. *Fault-tolerant computation with constant error.* Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, pp-176-188, 1997.

[Ahmed, 1990]
H.M. Ahmed. *Directions in DSP Processors*, IEEE Journal on Selected Areas i Communications , vol. 8 nº8 pp. 1420-1427, 1990.

[Akazawa et al, 2004]
Chihiro Akazawa, Hayami Tsuzuki, Yasuko Nakamura, Yo Sasaki, Kanae Ohsaki, Shun Nakamura, Yoshihiro Arakawa, and Shinichi Kohsaka. *The Upregulated Expression of Sonic Hedgehog in Motor Neurons after Rat Facial Nerve Axotomy* J. Neurosci., Sep 2004.

[Alonso, 2000]
D.Alonso y R.V Solé.*DivGame: a Cellular Automata model of rainforest dynamics.* Ecological Modelling 133, 131-141.

[Altwaijry, 1995]
H. Altwaijry y M. Flynn. *Performance/area trade-offs in Booth multipliers.* TSL-TR-95-684. Computer System laboratory. Stanford University, 1995.

[Andraka, 1998]
R. Andraka, A *survey of CORDIC algorithms for FPGAs*, Proceedings of the ACM/SIGDA 6th International Symposium on Field Programmable Gate Arrays, pp. 191-200, February 1998.

[Antelo et al, 1996]
E.Antelo, , J.D.Bruguera, T.Lang, J. Villaba. *High Radix Cordic Rotation Based on Selection by Rounding*. Euro-Par, Vol. II: 155-164-1996.

[Antelo et al, 1997a]

E.Antelo, J.D.Bruguera, T.Lang,E.L.Zapata: *High Performance Rotation Architectures Based on the Radix-4 CORDIC Algorithm*. IEEE Transactions on Computers 46(8): 855-870,1997.

[Antelo et al, 1997b]
E.Antelo, , J.D.Bruguera, T.Lang,E.L.Zapata. *Error Analysis and Reduction for Angle Calculation Using the CORDIC Algorithm*. IEEE Transactions on Computers 46(11): 1264-1271,1997.

[Antelo, 2000]
E.Antelo, T.Lang, J.D.Bruguera, *Very-High Radix Circular CORDIC: Vectoring and Unified Rotation/Vectoring*. IEEE Transactions on Computers 49(7): 727-739, 2000.

[Apu, 2004]
R. A. Apu; M. L. Gavrilova *Adaptive mesh generation for real-time terrain modeling*. Proceedings of the twentieth annual symposium on Computational geometry. 2004.

[Arnold et al 1990]
M.G. Arnold, T.A.Bailey, J.R. Cowles y J.J.Cupal. *Redundant logarithmic arithmetic*. IEEE Trans. on Computers, vol 39. nº 8 pp. 1077-1086. August 1990.

[Arnold, 2001]
M.G. Arnold y C. Walter. *Unrestricted Faithful Rounding is Good Enough for Some LNS Applications*. Proceedings of the 15th IEEE Symposium on Computer Arithmetic. June 2001.

[Arnold, 2002a]
M. Arnold. *21st Century Slide Rules with Logarithmic Arithmetic: High-Speed, Low-Cost, Low-Power Alternative to Fixed Point Arithmetic*. Online Symposium for Electronics Engineers, February 2002. Available at: http://www.osee.net.

[Arnold, 2002b]
M. Arnold, *Improved Cotransformation For LNS Subtraction*. IEEE International Symposium on Circuits and Systems, Scottsdale, AZ, ISBN 0-7803-7448-7, vol. II, pp. 752-755. May 2002.

[Arnold, 2003a]
M. Arnold, *Iterative Methods for Logarithmic Subtraction*

IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP'03) The Hague, The Netherlands. June 2003.

[Arnold, 2003b]
M. Arnold, J.García y M. Schulte. The *Interval Logarithmic Number System*. Proceedings of the 16th IEEE Symposium on Computer Arithmetic. June 2003.

[Baesch, 1997]
Baesch, A. and Steinmetz, N.: "Exceptional solutions of n-th order periodic linear differential equations," Complex Variables, Vol. 34, pp. 7-17, 1997.

[Baeumer, 2003]
B. Baeumer. *On the inversion of the convolution and Laplace transform* Trans. Amer. Math. Soc., Vol. 355 , pp.1201-1212, 2003.

[Ball y Bojanic, 2000]
S. Ball y R. Bojanik. *Table Look-up Method for Evaluation of Functions.* Journal of Approximation Theory 107, 2000.

[Bariani et al. 1997]
M. Bariani, R. Cucchiara, P. Mello. *Exploiting symbolic learning in visual inspection*. Proc. of IDA 97 4-6 1997, Lecture Notes in Computer Science, v. 1280, Springer, pp. 223-234 (ISBN 3-540-63346-4), 1997.

[Bak , 1997]
J.G.Bak, D. McMichael. *Convolution of a measure with itself and a restriction theorem*. Proc. Amer. Math. Soc., Vol. 125, pp. 463-470, 1997.

[Barrera, 1998]
J. Barrera, C.E. Ferreira y R.F. Hashimoto. *Finding Optimal Sequential Decompositions of Erosions and Dilations*. Proceedings of ISMM'98.

[Bartels, 1987]
R. Bartels, J. Beatty y B.Barsky. *An Introduction to Splines for Use in Computer Graphics and geometric Modeling*. Morgan Kaufmann. Los Alamos, California. 1987.

[Bartle, 1995]
R.G. Bartle. *The elements of integration and Lebesgue measure*. Wiley classics library edition. 1995.

[Bascompte, 1998]

J.Bascompte y R.V. Solé. *Spatiotemporal patterns in nature* Trends in Ecology and Evolution 13, 173-174, 1998.

[Beaumont-Smith et al, 1998]
A. Beaumont-Smith, N. Burgess, S.Lefrere. *Reduced Latency IEEE Floating-point Standard Adder Architectures.* Proceedings of the 14th Symposium on Computer Arithmetic, IEEE, 1998.

[Ben-Tzvi, 1990]
D.Ben-Tzvi y M. Sandler. *A Combinatorial Hough Transform.* J.P. Recognition Letters, Vol 11, pp. 167-174, 1990.

[Berkner, 1999]
K.Berkner: *Resolution of singularities of convolutions with the Gaussian kernel.* Proc. Amer. Math. Soc., Vol. 127, pp. 425-435, 1999.

[Bewick, 1992]
G.W. Bewick y M. Flynn. *Binary multiplication using partially redundant multipliers.* TR CSL-TR-92-528. Computer System Laboratory, Stanford University, 1992.

[Bewick, 1994]
G.W. Bewick. *Fast Multiplication: Algorithm and Implementation.* PhD Dept. Of Electrical Engineering, Stanford University, 1994.

[Beziers, 1970]
P.Beziers. *Emploi des machines à Commande Numérique.* Masson et Cie. Paris. 1970.

[Beziers, 1974]
P.Beziers. *Mathematical and Practical Possibilities on UNISURF.* Barnhill y Riesenfeld Editores. *Computer Aided Geometric Design.* Academic Press. Nueva York, 1974.

[Bickerstaff, 2001]
K'A.C. Bickerstaff, E.E. Swartzlander Jr *Analysis of Column Compression Multipliers* Ptoceedings of the 15th IEEE Symposium on Computer Arithmetic. June 2001.

[Booth, 1951]
A.D. Booth. *A signed binary multiplication technique.* Quarterly Journal of Mechanics and Applied mathematics, vol.4, nº 2, pp. 236-240, 1951.

[Bouville, 1985]
C. Bouville *Bounding Ellipsoids for Ray Fractal Intersection*. SIGGRAPH 85. pp 45-52.

[Boyer, 1986]
C.B.Boyer "Historia de las matemáticas" Alianza Univ. Textos, N94, 1986.

[Brocker, 1985]
Brocker, T and T Dieck, T. Representations of Compact Lie Groups . Sringer-verlag, 1985.

[Bruguera, 1993]
J.D.Bruguera, E.Antelo, T.Lang, *Design of a Pipelined Radix 4 CORDIC Processor*. Parallel Computing 19(7): 729-744, 1993.

[Bruguera et al, 1996]
J.D Brughera, N.Guil, T, Lang. *CORDIC based parallel/pipelined architecture for the the Hough Transform*. Journal of VLSI Signal Processing, vol. 12, pp. 207-221, 1996.

[Bruguera, 2001]
J.D.Bruguera, T.Lang *Using the Reverse-Carry Approach for Double Datapath Floating-Point Addition* Proceedings of 15th Symposium on Computer Arithmetic, IEEE 2001

[Bucgholz, 1999]
A.Buchholz. Norm of convolution by operator-valued functions on free groups. Proc. Amer. Math. Soc., Vol. 127, pp. 1671-1682, 1999.

[Cain, 2003]
M. E. Cain, B. S. Kapp y C. B. Puryear. *The Contribution of the Amygdala to Conditioned Thalamic Arousal.* J. Neurosciences. December 2002.

[Callaway, 1997]
T.A. Callaway, E.E. Schwartzlander.*Power delay characteristics of CMOS Multipliers*. Proceedings of the 13th Symposium on Computer Arithmetic. IEEE, 1997.

[Cao, 1997]
J. Cao y B. Wei, *High performance Hardware for Function Generation* Proceedings of the 13th symp. on Computer Arithmetic (ARITH'97).

[Cao, 2001]

J. Cao, B. Wei y J. Cheng. *High performance Architectures* for *forElementary Function Generation* Proceedings of the 15th symp. on Computer Arithmetic. June 2001.

[Cápek, 1997]
V. Cápek: *Exact memory integral in Time-Convolution Generalized Master Equations: Argyres-Kelley projector.* In: Zeitschrift fur Physik B - Condensed Matter, Vol. 104, pp. 323-331, 1997.

[Capelle, 1996]
J.Capelle. *Convolution on homogeneous spaces*. ISBN: 90-367-0686-6, 1996.

[Cardon, 2002]
D.A. Cardon. *Convolution operators and zeros of entire functions.* Proc. Amer. Math. Soc., Vol. 130, pp. 1725-1734, 2002.

[Cardona, 1996]
P. Cardona, Historia de la Informática. Recopilación. Mayo1996. http://www.mallorcaweb.net/mostel/index.html?http://www.mallorcaweb.net/mostel/historia.htm.

[Carlson, 2004]
B.A. Carlson y M. Kawasaki. *The Who's Who Signal in Electric Fish* J. Neuroscience, Septiembre 2004.

[Cavallaro, 1988]
J.R. Cavallaro y F.T Luk. *CORDIC arithmetic for SVD processor.* Journal of Parallel and Distributed Computing , nº5, pp.271-290, 1988.

[Cavallaro, 1991]
J.R Cavallaro y A.C. Lester. *CORDIC processor array for the SVD of a complex matrix. SVD and Signal processing II, Algorithms, Analysis and Applications*. R.J. Vaccado (editor), Elsevier Science Publishers, pp. 227-239, 1991.

[Cohn, 1980].
D.L. Cohn. *Measure Theory*. Birkhauser Boston-1980.

[Coleman, 1999]
J. N. Coleman and E. Chester *A 32 Bit Logarithmic Number System Processor and its Performance Compared to Floating Point.* 14th IEEE Symposium on Computer Arithmetic, pp. 142-152, Adelaide, Australia, April 1999.

[Coleman, 2000]

J.N.Coleman y E.I Chester. *Arithmetic on the European logarithmic processor* IEEE Trans. on Computers., vol 49nº 7 pp. 702-715. July 2000.

[Cornea-Hasegan, 1999]
M. Cornea-Hasegan y B. Norin. *IA 64 Floating-Point Operations and the IEEE Standard for Binary Floating-Point Arithmetic.* INTEL Technology Journal. 4th quarter 1999.

[Corazza et al.,2002]
xG.E.Corazza, P. Salmi, A. Vanelli-Coralli y M. Villanti, M. *Differential Post Detection Integration Techniqu in the Return Link of Satellite CDMA Systems*, IEEEISSSTA 2002 Conference, Czech Republic, 2002.

[Chamberlain, 2002]
Chamberlain, R;Lord, *Real-time 2D floating-point fast Fourier transforms for seeker simulation*." E. Proceedings SPIE. Vol 4717, pp.15-23 Technolologies for Synthetic Evironments: Hardware- in-the-loop Testing VII, Robert Lee Murrer Ed. July 2002.

[Chan, 1993]
E. Chan and S. Panchanathan, *A VLSI architecture for DFT*, Proceedings of the 36th Midwest Symposium on Circuits and Systems, pp.292-295. Detroit Michigan. USA, 1993.

[Chang, 1988]
L.W.Chang and M.Y. Chen A *new systolic array for discrete Fourier transform.* IEEE Trans. On Acoustic, Speech and Sugnal Processing, vol.36 pp.1665-1666, 1988.

[Chang, 2000]
T.S. Chang, J.T. Guo and C.W. Jen. Hardware *Efficient DFT Designs with Cyclic Convolutions and Subexpression Sharing*, IEEE Transactions on CAS II- vol. 47, nº9, pp.886-892, Sept.2000.

[Chang, 2000]
Chang, C.H., Wang, C.L *Efficient VLSI architectures for fast computation of the discrete Fourier transform and its inverse.*, IEEE Transactions on Signal Processing,vol. 48, nº11, pp.3206-3216, Nov.2000.

[Chang, 2001]
C.H. Chang, C.L. Wang,. *A DHT-based FFT/IFFT Processor for VDSL Transceivers*, Proceedings ICASSP'2001, pp.1213-1216.

[Chechile, 2003]
R.A.Chechile. *Mathematical tools for hazard function analysis.* Journal of Mathematical Psycology, Vol. 47, pp. 478-494, 2003.

[Chichyang, 2000]
C. Chichyang, C. Rui-Lin, Y. Chih-Huan. *Pipelined computation of very-large word length LNS addition/subtraction with polynomial hardware cost.* IEEE Trans. on Computers, vol 49. nº 7 pp. 716-726. July 2000.

[Chen, 1998]
C. Chen and C. H. Yang, *Pipelined Computation of LNS Addition/Subtraction with Very Small Lookup Tables*. Proceedings of the International Conference on Computer Design pp. 292-297, Oct. 5-7, 1998.

[Cheng et al, 2000]
F. Cheng, S.H. Unger, M.Theobald. *Self-Timed Carry-Lookahead Adders*. IEEE Transactions on Computers, vol 48 nº 7. Julio 2000.

[Chien, 2002]
Chien-Chang Lin, Chih-Da Chien. *A parametrized hardware design for the variable length discrete Fourier transform.* 15th International Conference on VLSI Design (CAD Symposium) 2002.

[Choi et al, 1997],
J.R. Choi, L.H. Jang, S.W. Jung y J.H. Choi. *Structured Design of a 288-Tap FIR Filter by optimized Partial product Tree Compression.* IEEE Journal of Solid State Circuits, vol.32, nº 3. 1997.

[Chu, 1999]
Chu, C-H. and Leung, C-W. *Harmonic functions on homogeneous spaces.* Monatshefte Math., Vol. 128, pp. 227-235, 1999.

[Chua, 2000]
C. Chua y U. Neumann *Hardware Accelerated Free Form Deformation* Eurographics/SIGGRAPH Graphics Hardware Workshop 2000, pp. 33-40, August 2000.

[Chua, 2001]
C. Chua y U. Neumann *A Modular Approach to Deformable Modeling and Animation* IEEE Computer Animation, pp. 184-191, November 2001.

[Chuang, 1995]

H.Y.H.Chuang y C.C.Li. *A systolic processor for straight line detection by modified HT.* IEEE Conf. on Computer Architecture for Pattern Analysis and Image Database Management, pp. 300-304, 1995.

[Cooley, 1965]
J.W. Cooley, J.W.Tukey, *An algorithm for the machine calculation of complex Fourier series,* Math. Comput. **19**, 297–301 (1965).

[Dadda, 1965]
L.Dadda. *Some Schemes for Parallel multipliers*, Alta Frequenza, vol.45, nº 3. Marzo, 1996.

[Dadda, 1996]
L. Dadda y V. Piuri. *Pipelined adders*. IEEE Transactions on Computers, vol 45 nº 3. March 1996.

[Daemen, 2000]
J. Daemen y V. Rijmen, *The Block Cipher Rijndael*, Smart Card Research and Applications, LNCS 1820, J.-J. Quisquater and B. Schneier, Eds., Springer-Verlag, 2000, pp. 288-296.

[Daemen, 2001]
J. Daemen y V. Rijmen, *Rijndael, the advanced encryption standard*, Dr. Dobb's Journal , Vol.~26, No.~3, pp.~137--139. March 2001,

[da Fontura, 1989]
L.da Fontura y M.B. Sandler. *A binary HT and its efficient implementation in a systolic array architecture.* J.P. Recognition Letters, Vol. 10, pp. 329-334, 89.

[Daniell, 1929]
P.J Daniell. *Stieltjes-Volterra Products,* Congr. Intern. des Math.Strasbourg, 920, pp 130-136. 1929.

[Das et al.1995]
D. Das et al. *Implementation of four co functions on an LNS co-processor.* IEEE Trans. on Computers, vol 44. nº 1 pp. 155-161-Jan. 1995

[Das, 1995]
D. Das, K. Mukhopadhyaya y B. P. Sinha, *Implementation of Four Common Functions on an LNS Co-Processor,* IEEE Transactions on Computers, vol. 44, no. 1, pp. 155-161, Jan. 1995.

[da Silva, 1990]

I. da Silva. *Vectorization from aerial photographs applying the HT method.* Proc. SPIE, Vol 1395, Pt2, pp. 956-963, 1990.

[De Dinechin, 2001]
F. de Dinechin y A. *Tisserand Some Improvements on Multipartite Table Methods*. Proceedings of the 15th IEEE Symposium on Computer Arithmetic. June 2001.

[De Lange et al,1990]
A.A. DeLange, **et al**. *Real time applicactions of the floating-point pipeline CORDIC processor in massive-parallel pipelined DSP algorithms*. Proc. ICASSP-90 pp. 1013-1016- 1990.

[Delgado, 1999]
J.Delgado y R.V.Solé. *Task Fulfilment and Temporal Patterns of Activity in Artificial Ant Colonies.*Lecture Notes in Artificial Intelligence 1674, 606-615.

[De Médicis, 1995]
A. De Médicis, P Leroux. *Generalized Stirling numbers, convolution formulae and p, q-analogues*. Canad. J. Math. Vol. 47, pp. 474-499, 1995.

[Deng, 2001]
D.S. Dixon, y H.El Gindy. *High speed Parametrizable HT using reconfigurable hardware*. Pan-Sydney Area Workshop and Visual Information Processing (VIP), 2001.

[De Reffye et al., 1988]
P. De Reffye, C. Edelin, C. Franon. *Plant Models Faithful to Botanical Stuctures and Development.* Pp. 51-158. SIGGRAPH, 1988.

[Dettweiler, 2003]
M.Dettweiler, S. Reiter. *On the middle convolution* Preprint (math.AG/0305311), 2003.

[Deutsch, 1999]
D.Deutsch. *Quantum theory of probability and decisions*. Proceedings of the Royal Society A455 3129-3197, 1999.

[Dick, 1998]
C. Dick. *Minimum Mutiplicative Complexity Implementation of the 2-D DCT using Xilinx FPGAs* Proceedings of SPIE's Photonics East'98 Configurable computing: Technology and Applications pp.190-201. Boston, MA USA Nov.1998.

[Deutsch, 2000a]
D.Deutsch, A.Eckert y R.Luppachini. *Machines, logics and quantum physics.* Bulletin of Symbolic Logic 3. 3-Sept. 2000.

[Deutsch, 2000b]
D.Deutsch y P. Hayden. *Information Flor in Entangled Quantum Systems.* Proceedings of the Royal Society A456 1759-1774. 2000.

[Dimitrov et al, 2001]
V. S. Dimitrov, J. Eskritt, L. Imbert, G. A. Jullien, W.C. Miller *The Use of the Multi-Dimensional Logarithmic Number System in DSP Applications.* Proceedings of the 15th IEEE Symposium on Computer Arithmetic. June 2001.

[Dong, 2001]
F. Dong, G.J Clapworthy y M.Krokos. *Volume Rendering of Fine Details Within Medical Data.* IEEE Visualization, San Diego, 2001.

[Dorf, 1989]
R.C. Dorf. *Sistemas modernos de control.* Addison-Wesley. Iberamericana 1989.

[D'Ornellas, 1998]
M.C. D'Ornellas y R. van den Boomgaard. *Generic Algorithms for Morphological Image Operators: A Case Study Using Watersheds.* Proceedings of ISMM'98.

[Eckorn et al., 1990]
R.Eckhorn, R. H.J. Reitboeck, M. Arndt y P.Dicke. *Feature linking via synchronization among distributed assemblies: Simulations of results from cat visual cortex.* Neural Comp. 293-307 -1990.

[Eijndhoven, 2003]
S.J.L.V.Eijndhoven, L.C.G.J.M Habets.*Equivalence of Convolution Systems in a Behavioral Framework.* Mathematics of Control, Signals, and Systems, Vol. 16, pp. 175-206, 2003.

[Ercegovac et al, 2000a]
M. D.Ercegovac, L.Imbert, D.W.Matula, J.Muller. *Improving Goldschmidt Division, Square Root and Square Root Reciprocal.* IEEE Trans. on Computers., vol 49 nº 7. Julio 2000.

[Ercegovac et al, 2000b]

M. Ercegovac, T.Lang, J. Muller, A Tisserand. *Reciprocation, Square Root, Inverse Square Root and Some Elementary Functions using Small Multipliers.* IEEE Trans. on Computers., vol 49 nº 7.July 2000.

[Ercegovac, 1994]
M.D.Ercegovac y T.Lang. *Division and Square root: Digit-Recurrence, Algorithms and Implementations*. Kluwer Academic Pub., 1994.

[Even, 2000]
G. Even y P. Seidel. *A comparison of Three Rounding Algorithms for IEEE Floating-point Multiplication*. IEEE Transactions on Computers, vol 49 nº 7. July 2000.

[Fang, 1997]
W.H. Fang, and M.L. Wu *An efficient unified systolic architecture for the computation of discrete trigonometric transform.* Proceedings ISCAS, pp.2092-2095. 1997.

[Feynman, 1983]
R. Feynman. *El carácter de la ley física*. Bosch. Barcelona 1983.

[Fernández et al, 2001]
C.Fernández, A. Galbis, M.C. Gómez-Collado. *Elliptic convolution operators on non-quasianalytic classes*. Arch. Math., Vol. 76, pp. 133-140, 2001.

[FitzHugh, 1961]
R. FitzHugh. Journal of Biophysiology nº 1. pp.445-466.

[Flynn, 1970]
M.Flynn. *On division by functional iteration*. IEEE Transaction on computers, vol.C-19, nº8. Agosto, 1992.

[Foley et al., 1990]
J. Foley, A Van Dam, S. Feiner. *Computer Graphics: Principles and Practice, Second Edition*. Addison-Wesley, Reading, Massachussets.1990.

[Fournier, 1982]
A. Fournier, D. Fussell y L. Carpenter. *Computer Rendering of Stochastic Models.* CACM, 25(6). pp. 371-384. 1982.

[Frigo, 1998]
M. Frigo, S. G. Johnson *FFTW: An Adpative Software Architecture for the FFT* Proceedings ICASSP Conference 1998, vol. 3, p. 1381.

[Frigo, 2000]
M. Frigo, S.G. Johnson *The Fastest Fourier Transform in the West* Technical Report MIT-LCS-TR-728, Massachusetts Institute of Technology, September 1997.

[Fulton, 1991]
Fulton, W., Harris, J. *Representation Theory.* A first Course, Springer_verlag, 1991.

[Gao, 2003]
Y. Gao, E.Nikulina, W.Mellado y M.T. Filbin. *Neurotrophins Elevate cAMP to Reach a Threshold Required to Overcome Inhibition by MAG through Extracellular Signal-Regulated Kinase-Dependent Inhibition of Phosphodiesterase.* J. Neurosci., December 2003.

[García et al. 2003a]
J.M García Chamizo, M.T- Signes Pont, H. Mora Mora, G. de Miguel Casado. *Parametrized Architecture for Hough Transform Recursive Evaluation.* Proc. SMMSP 2003, Barcelona, Spain, 2003.

[García et al. 2003b]
J.M García Chamizo, M.T- Signes Pont, H. Mora Mora, G. de Miguel Casado. *Hough Transform Recursive Evaluation Using Distributed Arithmetic.* Proceedings 12th IFIP International Conference on VLSI-SoC Systems, pp. 301-306, Darmstadt, Germany, 2003.

[Gasteratos, 1998]
A. Gasteratos, I. Andreadis y Ph. Tsalides *Soft Morphological Structuring Element Decomposition.* Proceedings of ISMM'98.

[Gelperin, 1989]
A.Gelperin, D.W.Tank y G.Tesauro. *Olfactory processing and associative memory: cellular and modeling studies.* Neural Models of Plasticity: Theoretical and Empirical Approaches. Eds. J.Byrne and W.O.Berry. Academic Press. New York, 1989.

[González, 1996]
R.C González y R.E Woods. *Tratamiento digital de imágenes.* Addison-Wesley. Iberamericana. S.A. 1996.

[Gottesman, 1999]
D.Gottesman y I.L.Chuang. *Quantum Teleportation is a universal computational primitive.* Nature, 402-392, 1999.

[Goto, 1997]
G.Goto. *A 4.1 ns compact 54x54b Multiplier Utilising Sign-Select Booth Encoders*. IEEE. J. Solid-State Circuits, vol.32, nº 11. Noviembre, 1997.

[Gousseau, 2001]
Y.Gousseau y J.M. Morel. *Are natural images of bounded variation*. SIAM J. on Mathematical Analysis, Vol.33 nº 3 pp. 634-648.

[Grabiner, 2004]
S. Grabiner. *Weak properties of weighted convolution algebras*. Proc. Amer. Math. Soc., Vol. 132, pp. 1675-1684, 2004.

[Hadwiger, 1957]
H. Hadwiger. *Vorslesungen über Inhalt. Oberfläche und Isoperimetrie.* Springer. Berlin. 1957.

[Hamana, 2003]
H. Hamana, J. Hirono, M. Kizumi, y T. Sato. *Sensitivity-dependent Hierarchical Receptor Codes for Odors* Chem Senses; 28(2): 87 104. February 2003.

[Harris et al, 1997]
D.L.Harris, S.F.Oberman, A.M. Horowitz. *SRT Division-Architectures and Implementations* IEEE 13th Symposium on Computer Arithmetic, 1997.

[Harth, 1990]
E.Harth, A.S.Pandya, K.P. Unnikrishnan, *Optimization of cortical responses by feedback modification and synthesis of sensory afferents. A modelf perception and rem sleep.* Concepts Neurosci. 1, 53-68, 1990.

[Haule, 1989]
D.D. Haule, D.D y A.S. *Object Recognition using fast adaptative HT.* IEEE Comp. Pacific Conf. On Communication, Compiler and Signal Processing, pp. 91-94, 1989.

[Haviland, 1980]
G.L Haviland y A.A. Tuszynski. *A CORDIC arithmetic processor chip*. IEEE Trans. on Computers, vol C-29 nº2 pp. 68-79- 1980.

[Hawkins, 1975].
T.Hawkins. *Lebesque theory of integration*. Chelsea.Pub.Co., 1975.

[Heijmans, 1998]
H. J.A.M. Heijmans y Jos B.T.M. Roerdink. *Mathematical Morphology and its Applications to Image and Signal Processing*. Computational Imaging and Vision- Vol. 12. Kluwer Academic Publishers, Dordrecht Hardbound, ISBN 0-7923-5133-9 May 1998.

[Hering, 2003]
H. Hering y M.Sheng. *Activity-Dependent Redistribution and Essential Role of Cortactin in Dendritic Spine Morphogenesis.* J. of Neuroscience., December 2003.

[Hodgkin, 1952]
A.L.Hodgkin y A.F. Huxley. *Journal of Physiology*.nº117- pp.500-544.

[Hogg, 1998a]
T. Hogg. *Highly Structured Searches with Quantum Computers* Physical Review Letters, vol 80. 1998.

[Hogg, 1998b]
T. Hogg y M Yanik *Local Search Methods for Quantum Computers* Xerox PARC technical report. 1998.

[Hogg, 1998c]
T. Hogg *A Framework for Structured Quantum Search* Physica-D vol 120 pp.102-116. 1998.

[Hogg, 2000a]
T. Hogg y D. Portnov *Quantum Optimization",* Information Sciences, vol 128 pp.181-197. 2000.

[Hogg, 2000b]
T. Hogg *Quantum Search Heuristics* Physical Review A, vol 61. 2000.

[Hogg, 2000c]
T. Hogg *Single-Step Quantum Search Using Problem Structure,* Intl. J. of Modern Physics C, vol. 11. 2000.

[Hogg, 2003]
T. Hogg *Adiabatic Quantum Computing for Random Satisfiability Problems.* Physical Review A vol 67. 2003.

[Hough,59]

P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures.* International Conference on High Energy Accelerators and Instrumentation, CERN, 1959.

[Hsiao, 2000]
S,.F.Hsiao, W.R Shiue, *Design of low-cost and high throughput linear arrays for DFT computations: algorithms, architectures and implementations.*, IEEE Transactions on Circuits and Systems II, Vol 47(11), pp.1188-1203. Nov. 2000.

[Hu, 1992]
Y.H.Hu. *CORDIC-based VLSI architectures for Digital Signal Processing.* IEEE Signal Processing Magazine, nº 7 pp. 16-35- July 1992.

[Huang, 2000]
L.Y.Huang, Z.Hu y F.M. Sun. *A New Automatic Quasar Recognition Technique Based on PCA and the Hough Transform.* ICPR 2000, pp. 2499-2502, 2000.

[Hyman et al, 2003]
J.M. Hyman, B.P. Wyble,V.Goyal, C.A. Rossi, y M.E. Hasselmo *Stimulation in Hippocampal Region CA1 in Behaving Rats Yields Long-Term Potentiation when Delivered to the Peak of Theta and Long-Term Depression when Delivered to the Trough* J. Neuroscience, December 2003.

[Ito, 1997]
M. Ito y T. Naofumi. *Efficient Inicial Approximation for Multiplicative Division and Square Root by a Multiplication with operand Modification.* IEEE Transactions on Computers, vol 46, nº4. April 1997.

[Jackson, 2004]
A C. Jackson, G.L. Yao y B.P. Bean *Spontaneous Firing in Clock Neurons.* J. Neuroscience, Sept. 2004.

[Janzing, 2001]
D. Janzing y Th. Beth, *Complexity measure for continuous-time quantum algorithms,* Phys. Rev. A **64** 022301. 2001.

[Jozsa, 1998]
R. Jozsa, *Quantum algorithms and the Fourier transform,* Proceedings of. R. Soc. Lond. A **454**, 323-37. 1998.

[Jozsa, 1999]
R. Jozsa, *Quantum effects in algorithms*, Lecture Notes in Computer Science 1509, 103-12. 1999.

[Kajiya, 1983]
J. Kajiya *New Techniques for Ray Tracing Procedurally Defined Objects.* SIGGRAPH 83. pp 91-102.

[Kalampukas et al, 2000]
L.Kalampukas, D.Nikolos, C. Efstathiou *High-Speed Parallel-Prefix Modulo 2^n-1 Adders.* IEEE Transactions on Computers, vol 48 nº 7. Julio 2000.

[Karasik, 1998]
Y.B. Karasik. *How to compute three-dimensional convolution and/or correlation optically: a mathematica foundation.* Journal of Modern Optics, Vol. 45, pp. 817-823, 1998.

[Kantabutra, 1993]
V. Kantabutra. *Designing optimum one-level carry-skip adders.* IEEE Transactions on Computers, vol 42 nº 6. Junio 1993.

[Katsuhiko, 1993]
O. Katsuhiko *Ingeniería de control moderna.* (2ª edición). Prentice-Hall Hispanoamericana. 1993.

[King, 2002]
King, J.A., Burgess, N., Hartley, T., Vargha-Khadem, F., & O'Keefe, J. (2002). *The human hippocampus and viewpoint dependence in spatial memory.* Hippocampus 12(6):811-20.

[Koch, 1985]
C. Koch, J. Marroquin y A.Yuille. *Analog neuronal networks in early vision.*

[Koren, 1990]
I.Koren, O.Zinati: *Evaluating Elementary Functions in a Numerical Coprocessor Based on Rational Approximations.* IEEE Transactions on Computers 39(8): 1030-1037 1990.

[Koren, 1993]
I. Koren. *Computer Arithmetic Algorithms.* Cap. 9, pp. 163-180. Englewood Cliffs, N.J.: Prentice Hall, 1993.

[Kornerup, 2003]
P. Kornerup *Revisiting SRT Quotient Digit Selection* Proceedings of the 16th IEEE Symposium on Computer Arithmetic.June, 2003.

[Koshimizu, 1990]

H.Koshimizu y M. Numada. *On fast Hough Transform method PLHT based on piecewise-linear Hough function.* J. System Computer in Japan, Vol 21 nº5, pp. 62-73, 1990.

[Kuhlmann y Pahi, 1998]
M. Kuhlman y K.K.Pahi. *Fast Low-Power Architecture.* Proceedings of International Conference on Computer Design. IEEE 1998.

[Kwon et al, 2000]
O. Kwon, K. Nowka y E.E. Shwarzlanger. *A 16-bit MAC Design Using Fast 5:2 Compressors.* IEEE Conference on Application-Specific Systtems, Architectures and Processors.

[Laflamme, 1996]
R. Laflamme, C.Miquel y J.P.Paz. *Perfect quantum error correction code.* Phys.Rev. Lett., 77: 198, 1996, arXive e-print quant-ph/9602019.

[Lai, 1991]
F. Lai y C.Wu, C. *A hybrid number system processor with geometric an complex arithmetic capabilities.* IEEE Trans. on Computers, vol 40. nº 8 pp. 952-962. August 1991.

[Lai, 1993]
F. Lai. *The Efficient Implementation and Analysis of a Hybrid Number System Processor.* IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processsing, vol. 40, no. 6, pp. 382-392, 1993.

[Lang y Montuschi, 1999]
T. Lang y P.Montuschi . *Very High Radix Square Root with Prescaling and Rounding and a Combined Division/Square Root Unit.* IEEE Transaction on Computers, vol 48 nº 8 Agosto 1999.

[Lang y Antelo, 2001]
T. Lang y E Antelo *Correctly Rounded Reciprocal Square-Root by Digit Recurrence and Radix-4 Implementation.* Proceedings of the 15th IEEE Symposium on Computer Arithmetic. June 2001.

[Lang y Antelo, 2003]
T. Lang y E Antelo *Radix-4 Reciprocal Square-Root and Its Combination with Division and Square Root* IEEE Transaction on Computers Vol. 52, No. 9 Septiembre 2003.

[Lang y Bruguera, 2004]

T. Lang y J.D. Bruguera *Floating-Point Multiply-Add-Fused with Reduced Latency*) IEEE Transaction on Computers Vol. 53, No. 8 Agosto 2004.

[Law et al., 2000]
C: Law, K. Yeo y S. Rofail. *A redundant-binary partial-product generator based on a five-bit recoding technique.* International Journal of Electronics, vol. 87, nº4, 2000.

[Lewis, 1990]
D.M. Lewis.*An architecture for addition and subtraction of long-word length numbers in the logarithmic number system.* IEEE Trans. on Computers, vol 39. nº 11 pp. 1325-1336. Nov.1990.

[Li, 2003]
X. Li, M.Davison y C.Essex. *Fractional Differential Equations and Stable Distributions,* Submitted to Journal of Applied Probability, 2003.

[Li et al. 1986]
H.F.Li, M.A. Lavin y R.J. Master. *Fast Hough Transform: a hierarchical approach.* J. Computer Vision Graphics Image Processing, Vol.36, pp. 139-161, 1986.

[Lindenmayer, 1968]
A. Lindenmayer. *Mathematical Models for Cellular Interactions in Development, Parts I and II.* J. Theor. Biol., 18. pp 280-315. 1968.

[Ma, 2003]
N.Y. Ma y R.P. King, R.P. *The n-fold convolution of generalized exponential-sum distribution functions*, Appl. Math. Comput. Vol. 142, pp. 23-33, 2003.

[Mc Cullogh, 1943]
W.S. Mac Cullogh y W. Pitts. *Bull. Math. Biophys.5.* 115-133.

[Mc Cann y Pippenger, 2003]
M.McCann y N. Pippenger. *SRT Division Algorithms as Dynamical Systems.* Proceedings of the 16th IEEE Symposium on Computer Arithmetic. June 2003.

[Markowska, 2002]
A. L. Markowska y A. V. Savonenko. *Effectiveness of Estrogen Replacement in Restoration of Cognitive Function after Long-Term Estrogen Withdrawal in Aging Rats.* J. Neurosciences December 2002.

[Mandelbrot, 1982]

B. Mandelbrot. *Tecnical correspondance*. CACM, 25(8) pp.581-583.

[Matheron, 1967]
G. Matheron. *Élements pour une théorie des milieux poreux*. Masson. Paris. 1967.

[Matula, 2001]
D. W. Matula Improved *Table Lookup Algorithms for Postscaled Division* . Proceedings of the 15th IEEE Symposium on Computer Arithmetic, June, 2001

[Matula y Fit-Florea, 2003]
D. W. Matula y A. Fit-Florea. *Prescaled Integer Division*. Proceedings of the 16th IEEE Symposium on Computer Arithmetic, June, 2003.

[Maurer, 2001]
S. Maurer; T. Hogg y B.A. Huberman *Quantum Portfolios* Physical Review Letters, vol 87. 2001.

[Max, 1979]
N.L.Max. *ATOMLLL: ATOMS with Shading and Highlights*. SIGGRAPHS 79, pp.165-173.

[Miel, 1993]
G. Miel *Constant Geometry Fast Fourier Transforms on Array Processors* Transactions on Computers March 1993,Vol. 42, No. 3. pp 371-375.

[Mintzer, 1996]
Les Mintzer *Large FFTs in a single FPGA*, Proceedings of ICSPAT'96.

[Mora, 2001]
J.M Mora Pascual. *Unidades Aritméticas en coma flotante para tiempo real*. Tesis Doctoral. Dept. Arquitectura y Tecnología de Computadores. Universidad de Alicante. 2001.

[Montuschi y Cimiera, 1993]
P. Montuschi y L. Cimiera. *Reducing Iteration Time when result digit is zero for 2 SRT división and square root with redundants remainders*. IEEE Transactions on Computers, vol 42 nº 2 1993.

[Montuschi y Cimiera, 1994]
P. Montuschi y L. Cimiera. *Over-Redundant digit sets and the design of digit-by-digit units*. IEEE Transactions on Computers, vol 43 nº 3 1994.

[Montuschi y Lang, 2001]
P.Montuschi y T. Lang. *Very High Radix Division with Prescaling and Selection by Rounding* IEEE Transaction on Computers, vol 50 nº 1 Enero 2001.

[Muamar, 1991]
H.K Muamar y M. Nixon. *Tristage Hough Transform for multiple ellipse extraction*. IEEE Proc. Part E: Computer and Digital Techniques, Vol 138 nº 1, 1991.

[Muller, 1998]
J.M. Muller, A. Sherbyna y A.Tisserand. *Semi-logarithmic number system.* IEEE Trans. on Computers, vol 47. nº 2 pp. 145-151. Feb. 1998.

[Murthy, 1994]
N.R Murthy and M.N.S. Swamy. *On the real-time computation of DFT and DCT through systolic architectures.* IEEE Trans. on Signal Processing, vol. 42, nº 4, pp.988-991, 1994.

[Nakayama, 1989]
Nakayama et al. *A 6.7 MFLOPS floatig-point coprocessor with vector/matrix instructions*. IEEE Journal on Solid-State Circuits, vol 24 nº 5 pp. 1324-1330 - 1989.

[Nielsen, 1997]
O.A.Nielsen. *An introduction to integration and measure theory.* John Wiley and Sons Inc. 1997.

[Oberlin, 2002]
D.M. Oberlin. *Some convolution inequalities and their applications*, Trans. Amer. Math. Soc., Vol. 354, pp. 2541-2556, 2002.

[Oberman y Flynn, 1997]
S.F. Oberman y M.J. Flynn. *División Algorithms and Implementations*. IEEE Transactions on Computers.1997.

[Obermann et al., 1997]
S.F.Oberman, H Altwaijry, M.J. Flynn. *The SNAP Project: Design of Floating-Point Units.* IEEE Proceedings of the 13th Symposium on Computer Arithmetic. IEEE 1997.

[O'Keefe, 1979]

J. O'Keefe. *A review of the hippocampal place cells*. Progr. Neurobiol., 13, 419-439.

[Oklobdzija et al, 1996]
V.G. Oklobdzija, D. Villeger, S.S. Liu. *A method for speed optimizedpartial product reduction and generation of fast parallel multipliers using an algorithmic approach.* IEEE Transactions on Computers. Vol. 45, nº 3. March 1996.

[Omondi, 1994]
A.R. Omondi. *Computer Arithmetic Systems*. Prentice Hall. 1994.

[Paliouras, 2001]
V. Paliouras and T. Stouraitis. *Low-Power Properties of the Logarithmic Number System.* Proceedings of the 15th IEEE Symposium on Computer Arithmetic, Vail, pp. 229-236, June 2001.

[Paliouras, 2002]
V. Paliouras. *Optimization Of LNS Operations For Embedded Signal Processing Applications*. IEEE International Symposium on Circuits and Systems, Scottsdale, AZ, ISBN 0-7803-7448-7, vol. II, pp. 744 - 747, 28 May, 2002.

[Parks, 2000]
M. Parks. *Number-Theoric Test Generation for Direct Rounding*. IEEE Transactions on Computers, vol 49 nº 7 July 2000.

[Parr, 2002]
A.W. Parr, *Compactly bounded convolutions of measures*, Proc. Amer. Math. Soc., Vol. 130, pp. 2661-2667, 2002.

[Parthasarathy, 1980]
K.R.Parthasarathy. *Introduction to probability and measure.*McMillan Press, 1980.

[Pease, 1968] *An adaptation of the fast Fourier transform for parallel processing*. J of the ACM Marshall C. Pease., vol.15,pp.252-264, 1968.

[Peiming, 2001]
Yan Peiming, Mo Yulong. *Image restoration based on the discrete fraction Fourier transform*. Proceedings SPIE. Vol 4552, pp.280-285 Image Matching and Analysis. Bir Bhanu Ed. Sept.2001.

[Peitgen, 1986]
H.O.Peitgen y P.H Richter.*The Beauty of Fractals:Images of Complex Dynamic System.* Springer-Verlag, Berlin,1986.

[Peszynska, 1996]
M. Peszynska. *Finite element approximation of diffusion equations with convolution terms*. Math. Comp., Vol. 65, pp. 1019-1037, 1996.

[Piñeiro, 2001]
J.A Piñeiro. J.D.Bruguera y J.M.Muller. *Faithful powering computation using table look-up and a fused accumulation tree.* Proc. of the 15th International Symposium of Computer Arithmetic (ARITH'15), 2001.

[Piñeiro, 2002a]
J.A. Piñeiro, M. Ercegovac y J.D. Bruguera. *High-Radix Logarithm with selection Rounding.* IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP'02), p-101, July 2002.

[Piñeiro, 2002b]
J.A. Piñeiro, J.D. Bruguera. *High-Speed Double precision Computation of Reciprocal Division, Square Root and Inverse Square Root.* IEEE Transactions on Computers, vol.51, nº 12, pp.1377-1388, 2002.

[Piñeiro, 2003]
J.A Piñeiro.M.D Ercegovac y J.D.Bruguera. *High-Radix Iterative Algorithm for Powering Computation.* Proceedings of the 16th IEEE Symposium on Computer Arithmetic. June 2003.

[Porter, 1979]
T.Porter. *The shaded Surface Display of Large Molecules.* SIGGRAPH 79.pp.234-36.

[Prusinkiewicz, 1988]
P. Prusinkiewicz, A. Lindenmayer y J. Hanan. Developmental Models of Herbaceous Plants for Computer Imagery Purposes. pp.141-150. SIGGRAPH 1988.

[Quach y Flynn, 1990]
N.T.Quach, M.J. Flynn. *An improved algorithm for high-speed floating-point addition.* TR CSL-TR-90-442. Computer System Laboratory, Stanford University.1990.

[Quach y Flynn, 1992]
N.T.Quach, M.J. Flynn. *High-speed addition in CMOS.* IEEE Transactions on

Computers. Vol. 41, nº 12. December, 1992.

[Rabiner, 1975]
L.R Rabiner and B. Gold. *Theory and application of digital signal processing.* Prentice –Hall, Englewood Cliffs, NJ, 1975.

[Rall, 1964]
W. Rall. Neural Theory and Modeling. R. Reiss Ed.pp.73-97. Stanford University Press.1964.

[Rall y Sheperd, 1968]
W. Rall y G. Sheperd. Journal of Neurophysiology nº 31, pp.884-915.

[Ramón y Cajal, 1894]
S. Ramón y Cajal. *New Ideas on the Strcture of the Nervous System in Man and Vertebrates.* Bradford Books. MIT Press. 1894.

[Randi, 2000]
T. Randi *An Architectural Performance Study of the Fast Fourier Transform on vector IRAM.* Technical report nº UCB/CSD-00.1106 Computer Science Division. University of California, Berkeley. June 2000.

[Sáez et al. 1998]
E.Sáez, E. et al. *FPGA implementation of a variable precision CORDIC processo".* 13th Conf. on Design of Circuits and Integrated Systems (DCIS'98).pp. 604-609. Madrid.Nov. 1998.

[Sánchez et al., 1997]
M. Sánchez, J.López, O, Plata, E.L.Zapata. *An efficient Architecture for the in-place Fast Cosine Transform.* Proceedings of IEEE International Conference on Application-Specific Systems, Architectures and Processors.pp 499-508. Zurich. July 1997.

[Schmookler, 2001]
M.S. Schmookler y K.J Nowka. *Leading zero anticipation and detection-a comparison of methods.* Proceedings of the 15th IEEE Symposium on Computer Arithmetic, June 2001.

[Schulte, 1994]
M.J. Schulte y E. Schwartzlander. *Hardware Designs for Exactly Rounded Elementary Functions.* IEEE Transactions on Computers, vol 43, nº 8, pp 964-972.August 1994.

[Schulte, 1997]
M.J.Schulte y J.E.Stine. *Accurate function approximations by symmetric table look-up and addition.* 11th International Conference on application-specific, systems, architecture and processors, 1997.

[Schwarz, 1996]
E.M. Schwarz. *Rounding for Quadratically Converging Algorithm for Division and Square Root*. Proceedings of the 29th ASILOMAR, IEEE 1996.

[Schwarz, 1999]
E.M. Schwarz y C.A Krygovski. *The S/390 GS Floating-Point.* IBM Journal of Research and Development, vol 43 nº 5/6. 1999.

[Schwarz, 2003]
E.M. Schwarz. *Panel: Revisions to the IEEE 754 Standard for Floating-Point Arithmetic.* Proceedings of the16th IEEE Symposium on Computer Arithmetic June, 2003.

[Seeger, 1996]
A. Seeger. *Singular integral operators with rough convolution kernels.* J. Amer. Math. Soc., Vol. 9, pp. 95-105, 1996.

[Seidel, 2001]
P.M.Seidel, L.D.McFearin, D.W.Matula, *Binary Multiplication Radix-32 and Radix-25* Proceedings of the15th IEEE Symposium on Computer Arithmetic June, 2001.

[Serra, 1965]
J. Serra. *L'analyse des textures par la géométrie aléatoire.* Compte-rendu du Comité Scientifique de l'IRSID.

[Serra, 1969]
J. Serra. *Introduction à la Morphologie Mathématique.* Cahiers du Centre de Morphologie Mathématique.Booklet nº 3. 160 pp. E.N.S.M.P.

[Serra, 1978]
J. Serra y R. Miles *En matière d'introduction..* In "Buffon Symposium". Lecture Notes in Biomathematics. Springer-Verlag, 1978.

[Serra, 1989]
J. Serra. *Image Analysis and Mathematical Morphology.* Vol.1.Academic Press. 1989.

[Shin y Jeon, 2000]
K.Shin y H. Jeon. *High-speed complex-number s based on redundant binary representation of partial products*. International Journal of Electronics, vol. 87, nº 6. 2000.

[Shankar, 1990]
R.V.Shankar y N. Asokan. *A parallel implementation of the Hough Transform method to detect lines and curves in pictures*. IEEE 32th Midwest Symp. On Circuits and Systems, pp. 321-324, 1990.

[Sherstyuk, 1999]
A. Sherstyuk. *Kernel functions in convolution surfaces: a comparative analysis.* The Visual Computer, Vol. 15, pp. 171-182, 1999.

[Shor, 1994]
P.Shor. *Algorithms for quantum computation: Discrete logarithms and factoring.* Proceedings of 35th Annual Symposium on Foundations of Computer Science. p124. Los Alamos, CA, 1994. IEEE Press.

[Shor, 1995]
P.Shor. *Scheme for reducing decoherence in quantum computer memory.* Phys.Rev. A, 52:2493-2496, 1995.

[Simon, 1996]
B. Simon. *Representations of finite and compact groups*. American Mathematical society 1996.

[Smith, 1984]
A.R. Smith. *Plants, Fractals and Formal Languages.* pp.1-10. SIGGRAPH 84.

[Solé,1994]
R.V. Solé y O Miramontes. *Information at the edge of chaos in fluid neural networks.* Elsevier Science B.V., Amsterdam.1994.

[Solé, 2001a]
R.V. Solé, *Complex Systems: Chaos and Beyond* SIAM review 43, 738-740-2001 .

[Solé, 2001b]
R.V. Solé y J.Mª Montoya. *Complexity and Fragility in Ecological Networks*Proceedings of the Royal Society of London B 268, 2039-2045, 2001.

[Solé, 2003]

R.V. Solé y S. Valverde *Information theory of complex networks* Networks: Structure, Dynamics and Function, Lecture Notes in Physics, Springer-Verlag, 2003.

[Solé et al, 2003a]
R.V.Solé, B.Luque y S.Kauffman *Order and chaos in random genetic networks with multiple states* International Journal of Bifurcations and Chaos 2003.

[Solé et al, 2003b]
R.V. Solé, J.M Montoya, R.Ferrer y J. Escoda *Universality and evolution in complex biological networks* Trends in Ecology and Evolution 2003.

[Song, 1991]
P.Song, G. De Michelli. *Circuit and Architecture Trade-Offs for High Speed Multiplication*. IEEE J.Solid State Circuits, vol. 26, nº 9, Septiembre, 1991.

[Stalling, 1995]
D. Stalling, H.C. Hege. *Fast and Resolution IndependentLine Integral Convolution* , Proc. ACM SIGGRAPH, pp. 249–256, 1995.

[Steane, 2001]
A.M. Steane y D. M. Lucas *Quantum Computing With Trapped Ions, Atoms and Light*'. Fortschritte der Physik special issue . October 2001.

[Steffen et al., 2003]
M. Steffen; W. van Dam;T. Hogg; G. Breyta y I. Chuang *Experimental Implementation of an Adiabatic Quantum Optimization Algorithm* Physical Review Letters, vol 90, 2003.

[Stelling et al., 1998]
P.F. Stellin, C.U. Martel, V.G. Oklobdzija, R.ravi. *OptimalCircuits for parallel Multipliers*. IEEE Transaction on Computers, vol.47, nº3. March, 1998.

[Strettoi, 2004]
Enrica Strettoi, Alan J. Mears y Anand Swaroop. *Recruitment of the Rod Pathway by Cones in the Absence of Rods* J. Neurosci., August 2004.

[Suga, 1990]
Suga, N. *Cortical computational maps for auditory imaging*. Neural Networks, 3, 3-21 -1990.

[Sural, 2001]

S.Sural y P.K.Das. *A genetic algorithm for feature selection in a neuro-fuzzy OCR system.* Sixth International Conference on Document Analysis and Recognition, pp. 987-991. Seattle, 2001.

[Swartzlander et al, 1993]
E.E. Swartzlander y DVS Chandra. *Sign/logarithmic arithmetic for FFT implementation.* IEEE Trans. on Computers, vol 32. nº 6 pp. 526-534. June 1983.

[Swarztrauber, 1987] *Multiprocessor FFTs*, P.N. Parallel Computing nº5, pp.197-210, 1987.

[Takagi et al, 1985]
N. Takagi, H. Yasuura y S. Yajima. High-speed VLSI multiplication Algorithm with a Redundant Binry Addition Tree. IEEE transaction on Computers, vol C-34 nº 9. Sept., 1985.

[Takagi y Horiyama, 1999]
N. Takagi y T. Horiyama. *A High-Speed Reduced-Size Adder Under Left-to-Right Arrival.* IEEE Transactions on Computers, vol 48 nº 1. January 1999.

[Takagi, 2001]
N. Takagi *A Hardware Algorithm for Computing Reciprocal Square Root* Proceedings of the 15th IEEE Symposium on Computer Arithmetic. June, 2001.

[Tan, 2003]
D Tan; A. Danysh y M. Liebelt. *Multiple-precision fixed-point vector multiply-accumulator using shared segmentation.* Proceedings of the 16th IEEE Symposium on Computer Arithmetic. June, 2003.

[Tandori, 1983]
K Tandori, *The life and works of Lipót Fejér, Functions, series, operators,* Colloq. Math. Soc. János Bolyai 35 (Amsterdam-New York, 1983), 77-85.

[Tchebychev, 1890]
P. Tchebychev *Sur deux théorèmes relatifs aux probabilités.* Acta. Math., t XIV, pp.305-315. 1890.

[Temperton, 1991]
C. Temperton, *Self-sorting in-place fast Fourier transforms.* SIAM J.Sci.Stat. Comput., vol 12 nº4, pp.808-823, 1991.

[Tenca y Ercegovac, 1998]

A.F.Tenka, M.D. Ercegovac. *On the Design of the High-Radix On-Line Division for Long Precision.*

[Tezmol et al., 2002]
A.Tezmol, H. Sari-Sarraf, S. Mitra. *Customized Hough Transform for Robust Segmentation of Cervical Vertebrae from X-Ray Images*. Fifth IEEE Southwest Symposium on Image Analysis and Interpretation, Santa Fe, New Mexico, April 2002.

[Traub, 1991]
RD Traub y R.Miles R *In: Neuronal networks of the hippocampus*. Cambridge, UK: Cambridge UP, 1991

[Ueki et al, 2003]
T. Ueki, M. Tanaka, K. Yamashita, S. Mikawa, Z.Fu Qiu, N. J. Maragakis, R. F. Hevner, N. Miura, H. Sugimura y K. Sato. *A Novel Secretory Factor, Neurogenesin-1, Provides Neurogenic Environmental Cues for Neural Stem Cells in the Adult Hippocampus.* J. Neuroscience December 2003.

[Um y Kim, 2001]
J.Um y T.Kim. *An optimal Allocation of Carry-Save Adders in Arithmetic Circuits.* IEEE Transactions on Computers, vol 50 nº 3. March 2001.

[Van Dalen, 1972]
D. Van Dalen y A.F. Monna. *"Sets and Integration"*. Wollers-Noordhoff, 1972.

[Van Dam, 2002]
W. van Dam, *Quantum algorithms for weighing matrices and quadratic residues,* Algorithmica 34, 413-428. 2002.

[Villalba, 1995]
J. Villalba, J. *Diseño de Arquitecturas CORDIC multidimensionales*. Tesis Doctoral Dept. de Arquitectura de Computadores. Universidad de Málaga, Nov.1995.

[Villaba,1996 et al]
J. Villalba, J., E. Antelo, J.D.Bruguera, E.L. Zapata. *Unified CORDIC architecture in redundant arithmetic.* Technical report. Universidad de Santiago de Compostela, Grupo de Investigación de Arquitectura de Computadores. December 1996.

[Villaba,1998]

J.Villalba, T.Lang, E.L. Zapata, *Parallel compensation of scale factor for the CORDIC algorithm.* Journal of VLSI Signal Processing Systems for Signal, Image and video technology, vol. 19 nº 3, pp. 227-241, August 1998.

[Volder, 1959]
J.E. Volder. *The CORDIC trigonometric computing technique.* IRE Trans. Elect. Comput., vol EC- 8 pp. 330-334. Sept. 1959.

[Volterra, 1913]
V. Volterra. *Leçons sur les fonctions de lignes.* Gauthier- Villars, 1913.

[Voss, 1987]
R.Voss. *Fractal in Nature. Characterization, Measurement and Simulation.* En Course Notes 15 for SIGGRAPH 87. Anaheim. California. 1987.

[Wallace, 1964]
C.S. Wallace. *A Suggestion for a Fast Multiplier.* IEEE Trans. Computers, vol.13, nº 2. February, 1962.

[Walther, 1971]
J.S Walther. *A unified algorithm for elementary functions.* Proc. Spring. Joint. Comput. Conf., pp379-385, 1971.

[Wang, 1996]
S.Wang y V.Piuri *A unified view of CORDIC processor design.* Application Specific Processors, edited by Earl E. Schwarzlander, Jr, Ch.5, pp 121-160, Kluwer. Academic Press, November 1996.

[Weinberger, 1981]
A Weinberger. *4:2 Carry-Save Module.* IBM Technical Disclosure Bull, vol. 23. january 1981.

[Weyl, 1927]
H. Weyl. y F.Peter. *Die Vollständigkeit der primitiven Darstellungen einer geschlossen kontinuierlichen Gruppe.* Math.Ann. t. XCVII. pp.737-755, 1927.

[Wheeden, 1977]
R.L. Wheeden,.A. Zygmund. *Measure and Integral.* Marcel Dekker, Inc. New York. 1977.

[White, 1989]
S. A. White, *Applications of Distributed Arithmetic to Digital Signal Processing*: Tutorial Review. IEEE ASSP Magazine, pp. 4-19, July 1989.

Volver al índice/Tornar a l'índex

Williams et al, 1995]
T.E. Williams, M.A. Horowitz. *A zero- overhead self-timed 160 ns 54-bit CMOS divider*. IEEE Solid-State Circuits, vol 26, nº 11. Nov.1995.

[Wiswman, 2000]
H.M. Wiseman y B.L. Hollis, *Space-bounded computation: quantum is better than classical,* LANL Preprint quant-ph/0009054. 2000.

[Wissam, 1996]
Rabadi *Wissam Iterative multiresolution algorithm for image reconstruction from the magnitude of its Fourier transform.* A. Texas Instrument Inc.; University of Central florida. Optical Engineering 35(04), 1015-1034. Brian J. Thomson Ed. April. 1996.

[Wong, 1994]
W.F. Wong, E. Goto. *Fast Hardware-Based Algorithms for Elementary Function Computations Using Rectangular Multiplier.* IEEE Transaction on Computers, vol. 43, no 4, pp. 278-294. 1994.

[Yamazava, 2000]
K.Yamazawa; Y.Yagi y M. Yachida. *3d Line Segment Reconstruction by Using Hyperomni Vision and Omnidirectional Hough Transforming. ICPR00,* Vol III: 487-490, 2000.

[Ye, 2002]
P.Ye, *The Approximation Theorem of Convolution Operator in Δp Set-valued Function Space*, Acta Mathematicae Applicatae Sinica, Vol. 18-3, pp. 495-500, 2002.

[Yeh , 2000]
W.Yeh y C. Jen. *High-Speed Booth Encoded Parallel Multiplier Design*. IEEE Transactions on Computers, vol.49, nº 7. July 2000.

Enlaces
La norma IEEE-754: http://cch.loria.fr/documentation/IEEE-754/index.html
Online Symposium for Electronics Engineers: http://www.osee.net.