

Unidades SIMD Reconfigurables para Procesamiento de Imagen

Por D. Aguado, P.A. Revenga, J.L. Lázaro, J.P. Derutin

D. Aguado,
P.A. Revenga,
J.L. Lázaro,
J.P. Derutin
Departamento de
Electrónica, Escuela
Politécnica Superior,
Universidad de Alcalá,
Campus Universitario
david@depeca.uah.es
revenga@depeca.uah.es
lazar@depeca.uah.es
derutin@lasmea.univ-
bpclermont.fr

Este artículo presenta un nuevo tipo de arquitectura de procesamiento paralela, aplicada a tratamiento de imagen en entornos de tiempo real embarcables. La arquitectura propuesta está basada en elementos de procesamiento integrados en el silicio de las FPGAs, pero aparece un nuevo concepto: módulos SIMD a medida para procesamiento de imagen. Para demostrar la validez de la propuesta se han montado dos prototipos basados en procesadores PowerPC comerciales, que se describen en este artículo, realizando un algoritmo de estabilización de imagen. Los resultados obtenidos se describen aquí y se utilizan para analizar que tareas del algoritmo son las que ocupan más tiempo de CPU y, por ello, son las candidatas a ser migradas a la lógica de la FPGA como unidades SIMD.

Introducción

La mayoría de algoritmos para procesamiento de imagen, y principalmente los relacionados con aplicaciones de visión artificial en tiempo real, demandan mucha velocidad de procesamiento en sus tareas. La alta potencia de computación requerida por estas aplicaciones normalmente no se cubre por procesadores estándar, e incluso es posible que estos altísimos rendimientos no se puedan cubrir ni por DSP's extremadamente rápidos ni por sistemas multiprocesador. Una de las formas más aceptadas para resolver este problema es derivar tareas a elementos con mayor potencia de procesamiento que solo realizan funciones muy específicas [1], [2], [3].

Tradicionalmente los procesadores introducían una sola unidad de datos (ya fuera un byte, un word, etc) dentro de un registro, aunque existiera espacio para varios de estos datos. Un sistema SIMD reúne varios de estos elementos dentro de un registro y realiza la misma operación sobre todos ellos a la vez [4]. Por lo

que SIMD es principalmente una optimización en cuanto a los registros y al Datapath. El principal problema que aparece con las unidades SIMD es el coste de la complejidad asociada al software, relativa a la programación de estas extensiones, pero también hay que tener en cuenta los altísimos rendimientos que se consiguen por la utilización de estos módulos en el resultado final. El hecho de utilizar estos aceleradores hardware lleva los cuellos de botella de rendimiento fuera del procesador y hace que se ubiquen en la jerarquía de memoria [5], por lo que a veces es preferible utilizar más tiempo de programación en optimizar los movimientos de datos que en minimizar el número de ciclos de instrucciones SIMD.

Las extensiones SIMD aparecieron por primera vez, a principios de los 90, en microprocesadores con MAX2 de HP y extensiones VS de Sun [6]. Desde entonces han aparecido multitud de unidades como MMX, SSE, SSE2, SSE3, 3DNow!, PowerPC SPE, VMX o AltiVec, que amplían las posibilidades de los programadores alcanzando el máximo rendimiento del hardware.

Recientemente se han desarrollado diversos trabajos que utilizan FPGAs como elementos altamente flexibles capaces de resolver el problema de la gran capacidad de procesamiento requerida por estos algoritmos [2], [7]. La capacidad de ayudar en este problema aparece por el comportamiento paralelo de estos dispositivos, así como los últimos avances en tecnología FPGA que nos proporciona la capacidad de trabajar a muy altas velocidades y ofreciendo a su vez gran cantidad de lógica. Estas características ofrecen flexibilidad para reconfigurar las unidades de procesamiento y gran capacidad de computación que hacen al sistema capaz de trabajar en tiempo real.

Motivación

AltiVec (Velocity Engine o VMX) es una unidad SIMD y un set de instrucciones de punto flotante y de enteros diseñada por la alianza AIM (Apple, IBM y Motorola), y que se ha venido implementando en las últimas versiones del procesador PowerPC [8], [9]. AltiVec ha sido el sistema SIMD más potente en sistemas de escritorio desde que se introdujo a finales de la década de los 90. AltiVec siempre ha proporcionado más registros y un set de instrucciones mucho más flexible que sus competidores, como las unidades MMX de procesamiento en entero o SSE que ya incluía punto flotante, por parte de Intel, u otros fabricantes de procesadores RISC. Sin embargo, los sets de instrucciones SIMD de las nuevas unidades, SSE2 y SSE3 [10], disponible en los procesadores Pentium 4 desde 2001 y 2004 respectivamente, y también implementadas por AMD en las arquitecturas AMD64 desde el año 2003, tienen más funciones que AltiVec.

Las principales arquitecturas SIMD disponibles, tanto AltiVec como SSE/2/3, están construidas sobre registros vectoriales de 128 bits. De esta forma, estas unidades pueden trabajar hasta con dieciséis chars de 8 bits, con signo o sin signo, ocho shorts de 16 bits, con o sin signo, cuatro enteros de 32 bits o cuatro datos de punto flotante. Actualmente la principal característica proporcionada por AltiVec y que no soporta SSE/2/3, es el tipo de datos pixel RGB para representar pixels de color de 16 bits. Pero, no puede trabajar con floats de 64 bits de doble precisión y además, es imposible mover datos directamente entre registros escalares y vectoriales. Debido a la naturaleza de carga y almacenamiento de la arquitectura PowerPC, tanto registros vectoriales como registros escalares solo se cargan desde memoria, y se almacenan en ella. En cualquier caso, los

tipos de dato y las instrucciones AltiVec son más completas y se dice de ellas que son más "horizontales", dado que pueden trabajar fácilmente a través de elementos dentro de un vector. AltiVec proporciona treinta y dos registros vectoriales de 128 bits mientras que SSE/SSE2/SSE3 ofrece solo registros XMM (registros de 128 bits de ancho), la mayoría de las instrucciones AltiVec utilizan tres registros de operando en comparación con los operando solo registro/registro o registro/memoria en IA-32 [11].

Actualmente, los procesadores embebidos que aparecen en dispositivos reconfigurables como las FPGAs, proporcionan altas potencias de cómputo a relativos bajos consumos. Estos procesadores no tienen extensiones SIMD estándar, como puede ser AltiVec o SSE, pero proporcionan la capacidad de implementar una unidad SIMD a medida en la propia FPGA, de forma que se puede unir al procesador incluso introduciéndola dentro del propio pipeline. La ventaja de este método es incrementar la potencia de procesamiento haciendo un profiling de la aplicación en cuestión, o del campo de aplicación, y desarrollar una extensión SIMD a medida que mejore el rendimiento del sistema. Esta extensión ejecutando una función completa siempre va a ser bastantes veces más rápida que una función software realizada por varias instrucciones estándar. Otra característica introducida es una gran flexibilidad, dado que el módulo se puede redefinir mediante un lenguaje de descripción hardware, como Verilog o VHDL. Esto es bastante interesante dado que las especificaciones siempre están cambiando y las aplicaciones siempre están evolucionando.

Metodología

Para aprovechar el hardware vectorial, las aplicaciones secuenciales se tienen que reescribir con ins-

trucciones vectoriales específicas sobre arrays completos en lugar de operaciones sobre elementos individuales de un array una detrás de otra. A la acción de reescribir el código de bucles en formato vectorial se le llama vectorización [12], y puede ser manual o automática. Cuando la vectorización la realiza automáticamente un compilador, transformando código serie a código vectorial por sí mismo, se llama auto-vectorización [13]. Si no existe dependencia de datos entre las iteraciones de un bucle, las operaciones de diferentes iteraciones se pueden iniciar en paralelo, y se puede por ello aplicar vectorización. Por tanto, el análisis de la dependencia de datos es un paso fundamental en el proceso de vectorización.

En las últimas versiones del compilador GCC 4.x, aparece un vectorizador que implementa una aproximación de vectorización basada en bucle, lo que significa que no tiene en cuenta el código directo, pero que explota paralelismo de datos dentro de las iteraciones de los bucles [14]. Por ello, actualmente el compilador GCC es capaz de compilar código que utilice unidades vectoriales SIMD sin necesidad de que el programador escriba explícitamente código SIMD.

Esto se consigue utilizando las directivas "-maltivec" en plataformas PowerPC o "-msse/msse2" en plataformas x86 y x86_64 en tiempo de compilación.

Otra forma de incrementar la velocidad de esas operaciones, que utilizan mucho tiempo dentro del procesador, es lo que proponemos en este artículo haciendo uso de las plataformas de co-procesamiento actuales basadas en FPGAs. Tras aplicar técnicas de "profiling", es posible detectar las tareas que más tiempo tardan dentro del algoritmo. Estas funciones han de ser analizadas buscando iteraciones que puedan reescribirse en formato vectorial y también ana-

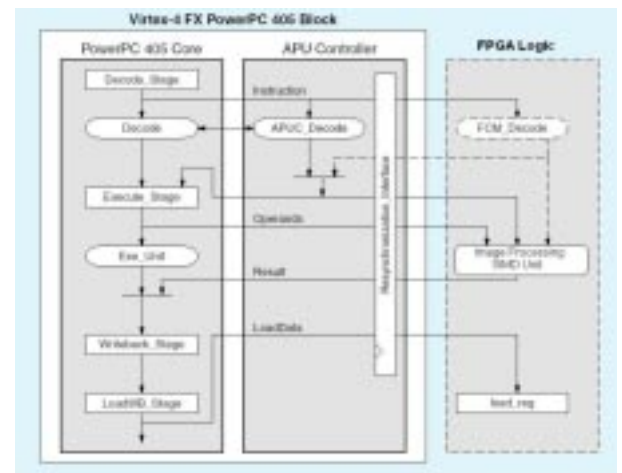


Figura 1. Flujo del Pipeline.

lizando si no existe dependencia de datos entre los operandos de las distintas iteraciones. En caso de no encontrarse dependencias, se puede describir un módulo VHDL a medida e implementarlo unido al procesador (en la lógica de la FPGA) para realizar estas operaciones.

Como se puede apreciar en la Figura 1, el controlador APU (Auxiliary Processor Unit), disponible en los procesadores PowerPC 405 integrados dentro de la familia de FPGAs Virtex-4 FX de Xilinx, proporcionan un interface de gran ancho de banda entre la lógica reconfigurable de la FPGA y el pipeline del core PowerPC 405.

Las unidades SIMD implementadas en la FPGA se pueden conectar al PowerPC empotrado a través del interface APU, permitiendo crear aceleradores hardware reconfigurables a medida. Estas unidades trabajan como extensiones del PowerPC 405, de forma que descargan a la CPU de tareas con gran carga computacional.

La combinación del controlador APU y la aceleración hardware en la FPGA proporciona una gran ventaja en cuanto al rendimiento, respecto a procesamiento software, o el método convencional de añadir co-procesadores en el bus de memoria del procesador.

Resultados preliminares

Para mostrar la validez de la arquitectura propuesta se han desarrollado dos prototipos. Estas máquinas, a las que ha denominado OSSIAN, están basadas en Apple G4 Cubes (con procesador Power-PC 7400, 128 MB, 450 MHz) en un caso y por cuatro iMAC G5 (con procesador Power-PC 970FX, 256MB, 1.6GHz) en el segundo caso, unidos por una red Fast Ethernet (100 Mb/s) para comunicación entre procesos. Estas máquinas explotan dos niveles de paralelismo, SIMD, utilizando instrucciones AltiVec, y MIMD, dado que se han montado como un cluster Beowulf.

Para obtener resultados preliminares, se han medido los rendimientos de estos prototipos OSSIAN, mediante la ejecución de una aplicación de estabilización de imagen en modo "batch", leyendo las imágenes y almacenando los resultados desde memoria. Este algoritmo implica tres partes principalmente: primera, submuestreo de la imagen generando dos nuevas imágenes con menor carga de datos. Segundo, detección de un conjunto de puntos de interés aplicando tres tipos de transformada Wavelet de Haar. Tercero, seguimiento de esos puntos en las proximidades utilizando filtros de correlación. La carga computacional de la segunda parte aumenta con el tamaño de la imagen de entrada y el número de puntos de interés. En el tercer paso, la cantidad depende de

la ventana de seguimiento y de la máscara utilizada para la correlación. En la Tabla 1 se muestra el tiempo para procesar un frame de la imagen de entrada para dos valores de número de puntos de seguimiento (concretamente 21 y 42) en imágenes de 320x240, en el segundo prototipo OSSIAN basado en procesadores PowerPC 970FX. El tamaño de la ventana de seguimiento y de la máscara de correlación es de 41x41 y 8x8 respectivamente.

Trabajo en progreso

Se ha realizado una búsqueda en el código para encontrar la actividad, del algoritmo de estabilización, que más tiempo consume. El resultado obtenido es que casi todas las etapas del algoritmo implican multiplicar matrices. En la segunda etapa, la imagen se multiplica por tres matrices distintas, cada una de ellas con diferentes valores de coeficientes de Haar, para detectar puntos de interés. En la tercera, para seguir estos puntos, se aplica un filtro de correlación a la imagen realizando una multiplicación matricial. Por ello, la operación candidata a ser desarrollada en hardware como unidad SIMD, será un Multiplicador Matricial. La siguiente tarea es detectar posibles dependencias de datos entre iteraciones del módulo SIMD. En esta aplicación, dado que las máscaras o los filtros se aplican sobre píxeles de un frame, no existe dependencia de datos, porque los resultados finales son independientes de los anteriores. Esta característica también es importante para nosotros, dado que hay una gran localidad de datos entre elementos y hace que los datos a utilizar sean fácilmente vectorizables.

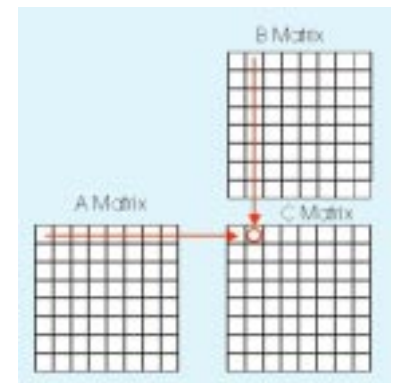
Como todos sabemos, para multiplicar matrices es necesario que el número de columnas de la primera matriz sea igual que el número de columnas de la segunda matriz, y en nuestro caso concreto, estas

son siempre son del mismo tamaño. Si A es una matriz mxn y B es una matriz nxp, entonces el producto AxB es una matriz mxp dada por:

$$(AxB)_x = \sum_{i=1}^n a_{xi} b_{i1} = a_{x1}b_{11} + a_{x2}b_{21} + \dots + a_{xn}b_{n1}$$

$$(AxB)_{x2} = \sum_{i=1}^n a_{xi} b_{i2} = a_{x1}b_{12} + a_{x2}b_{22} + \dots + a_{xn}b_{n2}$$

La Figura 2 muestra como calcular el elemento (C)12 resultante del producto AxB, siendo A y B matrices de dimensión 8x8. Los elementos de cada matriz se emparejan en la dirección de las flechas; cada par se multiplica y los resultados se suman. La posición del resultado en la matriz C corresponde a la fila y columna considerada.



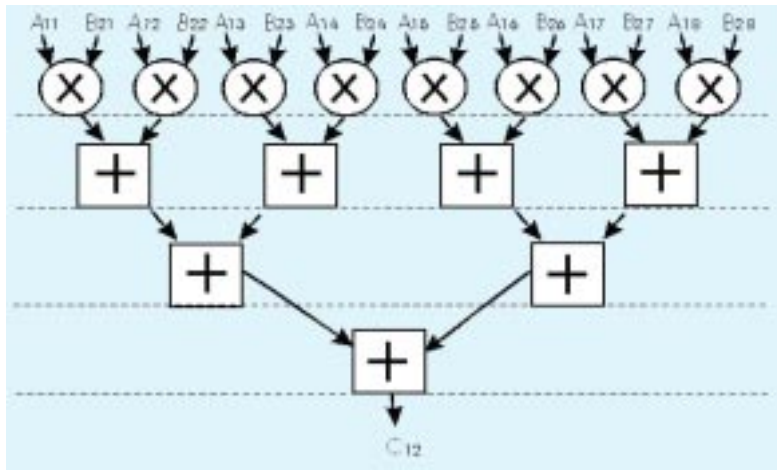
Por ello, para realizar una multiplicación de matrices, hemos realizado un sub-módulo como el mostrado en la Figura 3, que se replica hasta el número de elementos de una línea de la matriz. Por ejemplo, para una matriz de píxeles de tamaño 8x8, existen 8 sub-módulos como el presentado en la Figura 3 trabajando en paralelo. De esta forma, con solo cuatro ciclos se obtiene una línea de resultado de la matriz resultante. Lo mismo sucederá con matrices de mayor tamaño, aunque la utilización de área y la disipación de potencia será mayor también. Nuestro módulo tiene una ocupación de 5852 LUTs (Look Up Tables) y 1728 registros, lo que implica solo una ocupación del 4% de la lógica dispo-

Figura 2. Procesamiento del elemento (C)12.

Tabla 1

TABLA 1 TIEMPO EMPLEADO POR EL ALGORITMO DE ESTABILIZACIÓN PARA PROCESAR UN FRAME		
Configuración	Número de Puntos	Tiempo (ms)
Simple G5 1.6Ghz	21	52.5
G5 1.6Ghz + AltiVec		16.3
Ossian SIMD+MIMD		11.3
Simple G5 1.6Ghz	42	97.4
G5 1.6Ghz + AltiVec		25.2
Ossian SIMD+MIMD		13.4

Resultados tomados de la máquina OSSIAN, y de un simple nodo, estabilizando imágenes desde ficheros.



nible en una Virtex-4 XC4FX140, pero el 50% de nuestra Virtex-4 de menor capacidad XC4FX12. El elemento que más área consume es cada multiplicador, que ocupa 84 LUTs, y como se replica muchas veces, hace que este módulo tenga tanta ocupación. Es posible reducir su tamaño serializando su procesamiento, pero a la vez añadiendo más ciclo de computación a la unidad SIMD. Por ejemplo, desarrollando un multiplicador de 11 ciclos, en vez de uno de un solo ciclo, solo se ocupan 2368 LUTs en el módulo completo. Por lo que una vez más, hay que llegar a una solución de compromiso entre utilización de recursos y velocidad.

El controlador APU decodifica instrucciones de carga y almacenamiento de alto rendimiento entre la caché del procesador, o la memoria del sistema, y la lógica de la FPGA. Una simple instrucción puede transferir hasta 16 bytes de datos, es decir, cuatro veces más que una instrucción de carga y almacenamiento desde los registros de propósito general (GPR) del propio procesador. De esta forma, se crea un enlace de alta capacidad, y gran velocidad, hacia y desde el FCM (Fabric Coprocessor Module) [15].

Nuestra unidad SIMD conectada a través de APU al procesador PowerPC 405, se puede acceder des-

de código C o ensamblador. Las funciones que se han acelerado en Hardware, se integran en el Software con operaciones de carga y almacenamiento o mnemónicos en ensamblador. Estas operaciones son accesibles desde macros de alto nivel. Por ejemplo, para transferir 8 píxeles de 16 bit de memoria a un registro ubicado en la unidad SIMD a medida se realiza con la siguiente instrucción:

```
unsigned short pixel_row[8]; // 8 píxeles, cada uno de 16 bits
lgfcm(0, pixel_row); // transfiere una fila de píxeles al registro 0 dentro del módulo SIMD
```

Por ello, lo que hemos realizado en nuestra aplicación para realizar una transformada wavelet Haar, sobre una ventana de píxeles de tamaño 8x8, es cargar ocho píxeles realizando una carga de 64 bits.

La unidad SIMD ejecuta la transformada wavelet multiplicando las columnas de la matriz de coeficientes de Haar (estos valores han sido cargados previamente) por los ocho elementos de la línea cargada. La sub-operación termina realizando una carga de 64 bits.

La matriz completa se procesa realizando ocho sub-operaciones como esta. Por lo que realizar esta operación nos lleva solo 96 ciclos, mientras que realizar esta misma operación por soft-

ware, utilizando el juego de instrucciones estándar, consumirá un total de 5239 ciclos. De esta forma, el rendimiento obtenido por el módulo SIMD, realizando la misma operación, es 54.6 veces más rápido.

En este momento, el resto de trabajo para realizar las transformadas wavelet, el que se realiza tras la multiplicación por la matriz de coeficientes, se realiza por el propio procesador.

Por tanto, el trabajo que ahora estamos empezando es realizar la transformada Wavelet completa en el módulo SIMD. Una vez desarrollado, este módulo también puede ser replicado para realizar diferentes tipos de Wavelet.

Y también, realizar otras tareas SIMD, como por ejemplo aplicar máscaras de convolución a frames de imagen, dado que este tipo de operaciones utilizan un píxel central y necesitan trabajar con máscaras con número de filas y columnas impares.

Agradecimientos

Este trabajo ha sido apoyado por el proyecto español SILPAR (Sistema de Localización y Posicionamiento Absoluto de Robots).

Desarrollo de un Espacio Inteligente). Programa Nacional de Diseño y Producción Industrial, Ministerio de Ciencia y Tecnología, ref. DPI2003-05067 y por el Proyecto Regional TIFO (Sistema Sensorial de Detección y Transmisión de Imágenes por Fibra óptica para el desarrollo de un espacio Inteligente) de la Comunidad de Madrid ref. GR/MAT/720/2004).

El algoritmo de estabilización ha sido desarrollado por el laboratorio LASMEA, Laboratory of Sciences and Materials for Electronic and Automatic, Mixed Unit of

Figura 3. Esquema del Submódulo.

Research, University Blaise Pascal – CNRS, Francia.

Referencias

M. Ouellette, D. A. Connors "Analysis of Hardware Acceleration in Reconfigurable Embedded Systems". Proceedings of the 19th Parallel and Distributed Processing Symposium, p. 168, April, 2005.

I. Bravo I., R. Mateos, A. Gardel A., A. Hernández, J.L. Lázaro, R. Rivera, J.J. Heras, P. Jiménez "Detección de Bordos de Imágenes mediante Mascaras de 3x3 con FPGAs.", in Computación Reconfigurable & FPGAs (Proc. III JCRA Workshop), pp. 157-164, Madrid, Sept. 2003.

Nalini K. Ratha, Anil K. Jain "Computer Vision Algorithms on Reconfigurable Logic Arrays" IEEE Transactions on Parallel and Distributed Systems, Vol. 10, Num. 1, January 1999.

Alexandre E. Eichenberger, Peng Wu, Kevin O'brien, "Vectorization for SIMD Architectures with Alignment Constraints", PLDI'04, June 9-11 2004.

D. Cheresiz, B. Juurlink, S. Vasiliadis, H. Wijsho. "Performance Scalability of Multimedia Instruction Set Extensions" IEEE Transactions on Very Large Scale Integration (VLSI) Systems archive Volume 13 , Issue 1, pp. 1 – 13, 2005.

N. T. Slingerland, A. J. Smith, "Multimedia instruction sets for general purpose microprocessors: A survey," UCB, Tech. Rep. CSD-00-1124, December 1999.

Nalini K. Ratha, Anil K. Jain "Computer Vision Algorithms on Reconfigurable Logic Arrays" IEEE Transactions on Parallel and Distributed Systems, Vol. 10, Num. 1, January 1999.

K. Diefendorff, P. K. Dubey et al. "Altivec extension to PowerPC accelerates media processing". IEEE Micro, March-April 2000.

Apple Computer: <http://developer.apple.com/hardware/ve/>.

Apple Computer: <http://developer.apple.com/hardware/ve/sse.html>.

Intel: <http://www.intel.com/support/processors/sb/cs-001650.htm>

Randy Allen and Ken Kennedy "Optimizing Compilers for Modern Architectures-A dependence-based approach" Morgan Kaufmann, 2001.

S. Larsen, S. Amarasinghe. Exploiting superword level parallelism with multimedia instruction sets. PLDI, 35(5):145-156, 2000.

D. Naishlos. "Autovectorization in GCC". In Proceedings of the 2004 GCC Developers Summit, pp. 105-118, 2004.

A. Ansari, P. Ryser, D. Isaacs "Accelerated System Performance with APU-Enhanced Processing" Xcell Journal Article, Issue 52, pp. 36-39, 2005.

Biografías de los autores

David Aguado

Obtuvo el grado de Ingeniero Técnico de Telecomunicación e Ingeniero en Electrónica por la Universidad de Alcalá en 2000 y 2003 respectivamente. Actualmente está realizando estudios de doctorado en el Departamento de Electrónica la Universidad de Alcalá. Desde el año 2000 ha desarrollado su labor profesional como Ingeniero de Diseño Hardware para diversas empresas como Indra Sistemas y Fedetec.

Su actividad investigadora se centra en dispositivos reconfigura-

bles, arquitecturas multiprocesador y visión artificial.

Pedro Revenga

Obtuvo el grado de Ingeniero Técnico de Telecomunicación por la Universidad de Alcalá en 1989 y de Ingeniero en Electrónica por la Universidad de Valencia en 2000. Actualmente está finalizando su tesis doctoral en el Departamento de Electrónica la Universidad de Alcalá de donde es profesor titular desde el año 1991.

Su actividad investigadora se centra en procesamiento de imagen, arquitecturas multiprocesador, sistemas de posicionamiento global y sistemas embarcables.

José Luis Lázaro

Obtuvo el grado de Ingeniero en Electrónica y Telecomunicación de la Universidad Politécnica de Madrid en 1985 y 1992 respectivamente, y de doctor en Telecomunicación por la Universidad de Alcalá en 1988 donde es actualmente profesor.

Su actividad investigadora se centra en los sistemas de sensores láser para robótica, infrarrojo y visión artificial aplicados a los espacios inteligentes y arquitecturas multiprocesador.

Jean Pierre Derutin

Actualmente ocupa el puesto de Catedrático de Universidad en la Universidad Blaise Pascal de Clermont Ferrand (Francia). Desde 1983 desarrolla sus actividades docente e investigadora en el CUST (Centro Universitario de Ciencias y Técnicas) de la UBP y en el laboratorio LASMEA (Laboratorio de Ciencias y Materiales para la electrónica y la Automática UMR-6602 CNRS) en el Grupo de Automática y Visión para la robótica.

Su línea de investigación fundamental es el desarrollo de arquitecturas multiprocesador de bajo volumen y consumo, para visión artificial en tiempo real.