Throughput Unfairness in Fair Arbitration Interconnection-Buses for Aerospace Embedded Systems

Salvador Ibarra-Delgado, Remberto Sandoval-Arechiga, *Member, IEEE,* María Brox, and Manuel Ortíz-López.

***Abstract*—Currently, embedded systems are composed of processors, memories, and Intellectual Property Cores (IP Cores) interconnected to develop a set of specific tasks. Therefore, the selection of an appropriate interconnection architecture is critical in terms of system performance and functionality. A Network-on-Chip provides an efficient and scalable interconnection solution when there are a large number of elements in the system. However, the bus-based interconnection system remains the best option to connect a few cores. The bus arbiter uses an allocation policy to select which IP Core obtains access to the bus. The so-called fair policies ensure that all processors in the system have the same opportunity to access the bus. However, they fail to offer a fair share of the bandwidth or transmission rate, especially when there are heterogeneous IP Cores. As a study case, we analyze an embedded aerospace system for earth observation. Different IP Cores preprocess satellite images at distinct execution times -and unbalanced processing rates- affecting the delivery rate of images to earth. We study the phenomenon of uneven bus transmission rates due to improper bus allocation using policies such as Round Robin, FIFO, and Lottery. Also, we propose a metric to compute the maximum number of IP Cores without bus saturation.**

***Index Terms*—Fairness, arbitration, bus, aerospace, embedded systems.**

I. Introducción

LAS metodologías actuales para diseño de sistemas embebidos aprovechan el reuso de bloques de propiedad intelectual. Estos aceleradores de hardware —bloques que realizan una función muy específica, conocidos como IP Cores– se conectan empleando buses, conmutadores o redes en chip, construyendo un Sistema en Chip (SoC). Este paradigma permite que los diseñadores se concentren en la funcionalidad y desempeño del sistema. Este último está subordinado principalmente a una comunicación eficiente entre los procesadores y a una distribución balanceada de la carga entre ellos, más que en su frecuencia de operación. Por lo tanto, la interconexión de los bloques se convierte en el cuello de botella conforme la complejidad del sistema aumenta [1].

Las redes en chip son una opción eficiente y escalable para niveles de integración del orden de decenas de núcleos en el mismo SoC. Pero en sistemas pequeños –de cuatro a ocho procesadores, como se afirma en [2]–, las arquitecturas de interconexión basadas en buses siguen siendo la solución de referencia en el estado del arte de los sistemas multinúcleo debido a su diseño simple y menor costo de hardware. Sin embargo, los diseños modernos llevan a estas arquitecturas al límite del desempeño dentro de los configuraciones disponibles en los buses comerciales.

Sin importar la arquitectura de interconexión usada, en los sistemas embebidos es común tener recursos compartidos —memoria, periféricos, aceleradores, IP Cores, etcétera— entre múltiples núcleos de procesamiento independientes. El *Árbitro* del bus se encarga de administrar y decidir quién tiene acceso a los recursos en un instante de tiempo dado. El arbitraje es determinante en el desempeño del sistema conforme asigna a los procesadores prioridades de acceso a los recursos compartidos. Dos aspectos son fundamentales en el arbitraje [3], [4], [5], [6]: las prioridades, que asignan el acceso al recurso compartido cuando dos o más peticiones existen; y la equidad, que garantiza a las peticiones de baja prioridad el acceso al recurso compartido sin importar el número de peticiones en el sistema.

Entre los esquemas de arbitraje más comunes se encuentran *Round Robin*, *FIFO*, *TDMA* y *Lottery*, los cuales tienen un alto grado de equidad en términos del número de peticiones asignadas [2]. Sin embargo, no se garantiza equidad en la tasa de transmisión. Cuando las peticiones de los diferentes procesadores se refieren a transacciones con duración distinta, las más largas se apoderan de mayor ancho de banda en detrimento de procesadores con transacciones más cortas [2]. Este fenómeno lo observamos en el procesamiento de imágenes adquiridas por los sistemas satelitales del tipo CubeSat [7], que deben de pasar por un conjunto de transformaciones antes de ser transmitidas. Las diferencias en la duración de las transacciones de cada uno de estos procesos no permite generar flujos homogéneos de datos, lo que disminuye la tasa de transmisión de imágenes a tierra en la ventana de tiempo disponible.

En la literatura actual encontramos estudios diversos sobre árbitros justos. En [6] se estudia una implementación asíncrona de arbitraje *Round Robin* y se realiza una evaluación para diferentes patrones de tráfico. Incluso en los enrutadores empleados en las redes en chip se han hecho estudios recientes de implementaciones de arbitrajes *Round Robin* [8] y los problemas que puede ocasionar una selección inadecuada de arbitraje en los enrutadores [9]. En [5] se estudia un esquema de arbitraje en buses basado en *Lottery*, el cual

S. Ibarra-Delgado y R. Sandoval-Arechiga trabajan en el Centro de Investigación, Innovación y Desarrollo en Telecomunicaciones (CIDTE), Universidad Autónoma de Zacatecas, Zacatecas, 98010, México,(e-mail: sibarra@uaz.edu.mx y rsandoval@uaz.edu.mx).

M. brox y M. Ortíz-López trabajan en la Universidad de Córdoba, España, (email: mbrox@uco.es y el1orlom@uco.es).

Manuscript received Sept 13, 2019; revised xx, 2019.

mejora la utilización del bus y la equidad en el acceso de los esquemas tradicionales. La equidad en la tasa de transmisión ha sido explorada mediante dos técnicas. En primer lugar, usando *Worst Case Execution Time (WCET)* y créditos para garantizar anchos de banda en aplicaciones espaciales en [2]; y en segundo lugar, empleando esquemas de reservación con ventanas de transmisión periódicas en buses AXI [10]. Sin embargo, observamos que en la literatura hace falta un análisis más detallado de la magnitud del problema de inequidad en la tasa de transmisión ocasionada por los árbitros justos, especialmente cuando las duraciones de las transacciones —en términos de ciclos de reloj, que a su vez depende del tamaño de la transacción y el ancho de bus-– son significativamente diferentes.

Las aportaciones de este trabajo se enlistan como sigue:

- Se propone una medida para encontrar el número máximo de IP Cores esclavos —en un sistema homogéneo, con maestros y esclavos del mismo tipo-– que pueden operar de manera simultánea sin saturar la capacidad del bus.
- Se evaluaron tres políticas de arbitraje empleando modelos en SystemC trabajando con precisión de ciclo de reloj, y se encontró que no hay diferencias significativas entre ellas en términos de capacidad de transmisión y tiempos de acceso al bus.
- Finalmente, cuando se evaluaron diferentes escenarios tanto sintéticos como con aplicaciones espaciales, se encontró que en un sistema heterogéneo, no hay equidad en términos de la tasa de transmisión.

El trabajo está dividido como sigue: en la sección II se describe la motivación del trabajo y la arquitectura del sistema propuesto. La sección III presenta el modelo del sistema en SystemC y las suposiciones realizadas. Los resultados son analizados en diferentes escenarios en la sección IV y finalmente se presentan las conclusiones en la sección V.

II. Descripción del Problema

Nuestro interés en el problema de inequidad por el arbitraje en sistemas embebidos inició con el estudio de un sistema satelital de observación terrestre. En dicha aplicación se tiene la necesidad de capturar y transmitir hacia una estación terrena imágenes satelitales. A través del procesamiento de imágenes se hace la identificación de sequías, incendios, inundaciones, deslaves y deforestación, entre otros desastres naturales. Para ello se diseñó una arquitectura (ver Fig. 1) basada en sistemas embebidos a bordo de nano-satélites empleando el estándar CubeSat.

Estos satélites capturan imágenes de alta resolución en diferentes espectros —frecuencias-– que después serán combinadas para extraer la información de interés. Sin embargo, la capacidad de procesamiento de imágenes se ve comprometida por el tráfico de datos en el bus de interconexión de los IP Cores de procesamiento en el satélite. Es decir, el *Árbitro* justo garantiza el acceso al mismo, pero no hay una garantía que los procesos tengan una tasa de transmisión dada -–esto impacta en una variación en la cantidad de las imágenes listas para transmitir en un tiempo dado-– conforme aumenta la carga en el sistema. De aquí la necesidad de cuantificar el problema y bosquejar posibles soluciones para futuras aplicaciones.

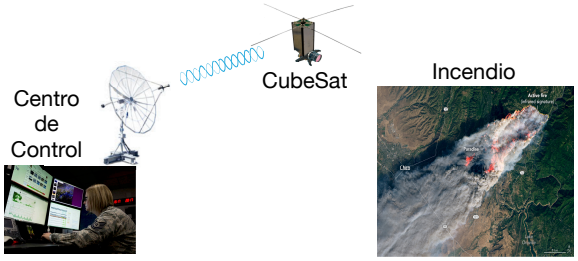


Fig. 1. Sistema satelital de observación terrestre para prevención de desastres naturales.

A. Arquitectura del CubeSat

La arquitectura del sistema embebido CubeSat se puede observar a la izquierda en la Fig. 2. En este trabajo nos concentramos en la parte del procesamiento digital, el cual se encuentra embebido en un SoC. Se plantea el desarrollo de un sistema de transmisión segura de imágenes satelitales. La implementación se realiza sobre un sistema de interconexión tipo bus. Desde el punto de vista del procesamiento de imágenes y de la información, el sistema debe de cumplir con los siguientes requerimientos:

1) Generar imágenes libres de errores radiométricos generados por la óptica de la cámara o por interferencias del medio ambiente.
2) Comprimir las imágenes sin pérdidas para que sean menos costosas en términos del tiempo de transmisión.
3) Transmisión segura utilizando algún método de cifrado.

Bajo estas restricciones se plantea una arquitectura del SoC mostrada a la derecha de la Fig. 2. Una imagen de 1024x1024 pixeles es adquirida por un medio externo y es almacenada en la memoria principal (1). Un maestro –generalmente procesadores– toma esta imagen línea por línea y genera una imagen filtrada utilizando el IP Core de filtrado (2). Esta imagen es tomada por el siguiente maestro en el sistema (3), el cual por medio del IP Core de compresión, se encarga de reducir el tamaño de la imagen (4). La imagen comprimida es tomada por el último maestro en el flujo (5) y la cifra por medio del IP Core encargado de este proceso (6). Finalmente la imagen cifrada es transmitida por el Radio Tx (7).

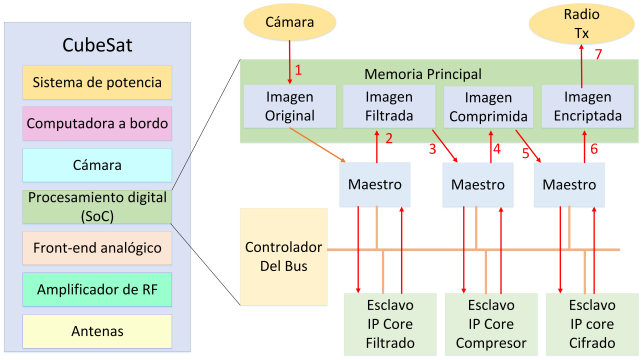


Fig. 2. Arquitectura del sistema embebido CubeSat. Se muestra el flujo de datos de la aplicación empleada.

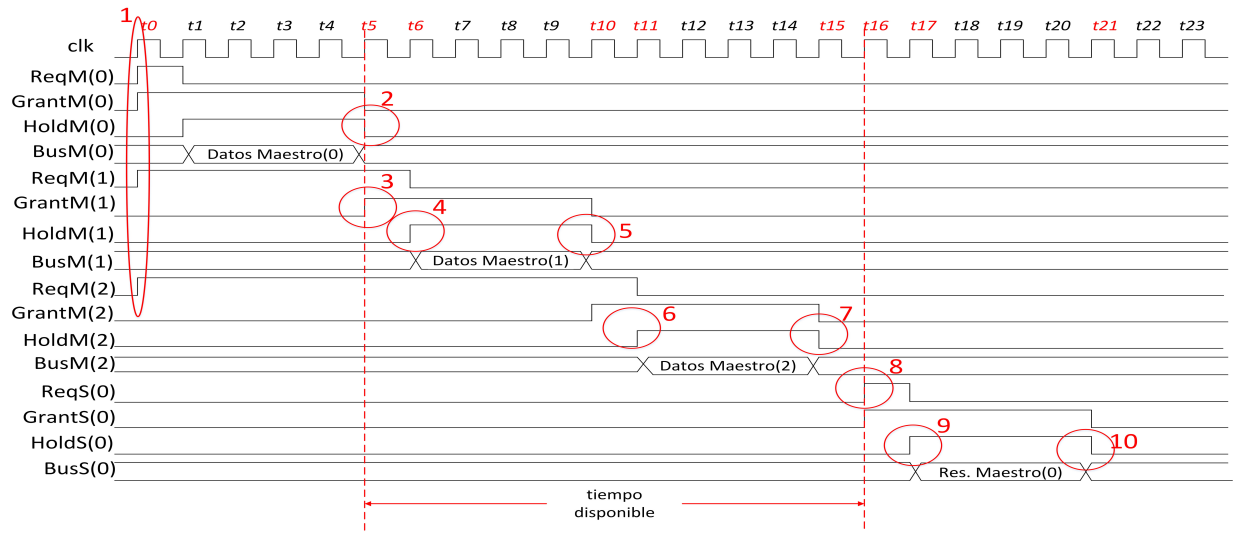


Fig. 3. Diagrama temporal de las señales del bus con la técnica *Split-Transaction*. El tiempo disponible t_e entre transacciones del maestro 0 es aprovechado por transacciones de otros maestros, mejorando la utilización del bus.

B. Descripción de los IP Cores

Los IP Cores empleados tienen diferentes características como tamaño de transacción y tiempo de ejecución del procesamiento. Para el sistema aeroespacial planteado en este estudio, se tiene lo siguiente:

- *Filtro*, para eliminación de ruido causado debido a las condiciones en la que la imagen fue adquirida. La implementación hardware de estos filtros normalmente se realiza por medio de arreglos sistólicos que utilizan una máscara de convolución y realizan el proceso de filtrado a razón de un pixel por ciclo de reloj. El filtro empleado acepta como entrada una línea de una imagen original y genera como salida una línea de la imagen filtrada. Para la prueba se utilizó una implementación propia del algoritmo de suavizado de imagen con una latencia de un ciclo de reloj por pixel.
- *Compresor*, disminuye el tamaño de la imagen para facilitar su transporte. En este trabajo, se decidió utilizar el algoritmo sin pérdidas *Lossless Image Compression Algorithm*, LOCO, con la implementación realizada en [11] con una compresión de hasta 50% del tamaño original. La implementación hardware de este algoritmo recibe como entrada una línea de la imagen, procede a realizar el proceso de compresión y entrega una línea comprimida. El IP Core tiene un tiempo de procesamiento de 16 ciclos/pixel.
- *Cifrador*, la protección de información es clave en los sistemas embebidos. Se utilizó un cifrador AES el cual, una vez que se le ha dado la llave de cifrado, recibe 16 Bytes de datos y entrega 16 Bytes de datos cifrados. La implementación AES-128 utilizada está descrita en [12], con un tiempo de procesamiento de doce ciclos de reloj.

C. Esquema Split-Transaction

En un esquema de bus tradicional, cuando un maestro toma el control del bus, manda un paquete de datos a un esclavo y queda a la espera de la respuesta manteniendo el control del bus, por lo que en este periodo el bus está inactivo. *Split-Transaction* es una técnica utilizada en sistemas de buses compartidos para evitar la inactividad del bus. Esta técnica divide la transacción en dos: primero, el maestro toma el control del bus, transmite el paquete, libera el bus y queda en espera de la respuesta por parte del esclavo; segundo, cuando el esclavo tiene un resultado disponible solicita acceso al bus y cuando lo tiene, transmite el resultado al maestro que solicitó la operación. El tiempo que tarda el esclavo en tener la respuesta, se denomina tiempo de respuesta t_r. Durante este tiempo el bus queda libre y a disposición de cualquier otro elemento en la red. En la Fig. 3 se observa que en la transición positiva del reloj en t0 (1), tres maestros solicitan el uso del bus (ReqM(0) = 1, ReqM(1) = 1, ReqM(2) = 1). El *Árbitro* otorga el bus al maestro 0 (GrantM(0) = 1) el cual inicia con la transmisión de un paquete, conservando el uso del bus (HoldM(0) = 1). Cuando termina de transmitir el paquete en t5, libera al bus (HoldM(0) = 0) (2) y el maestro 0 queda a la espera de la respuesta por parte del esclavo. Al estar el bus libre, ahora el *Árbitro* lo asigna al maestro 1 en t5 (GrantM(1) = 1) (3), este procede con la transmisión de datos desde t6 (HoldM(1) = 1) (4) hasta t10 (HoldM(1) = 0) (5), y el maestro 1 queda a la espera de la respuesta por parte del esclavo al que transmitió el paquete. Como el bus vuelve a quedar libre, el *Árbitro* concede su uso al maestro 3, y este procede con la transmisión de un paquete de datos entre t11 (6) y t15 (7). En t16 (8) el esclavo asociado al maestro 0 tiene disponible el resultado, solicita el uso del bus, lo obtiene y procede a la transmisión del resultado entre t17 (9) y t21 (10). En la Fig. 3, se puede observar que el espacio de tiempo entre t5 y t16, es el tiempo que tarda el esclavo 0 en procesar los datos que le fueron transmitidos por el maestro 0, espacio que fue utilizado por el maestro 1 y el maestro 2 para hacer uso del bus. Se puede comprobar que con esta técnica se tiene un mayor aprovechamiento del bus.

D. Tipos de Arbitraje Justos

Se dice que un *Árbitro* es justo si este asegura una atención igual a los núcleos que están solicitando ser atendidos. Según [13], la justicia puede ser: *Justicia Débil* si cada petición es eventualmente servida; *Justicia Fuerte* si las peticiones de los diversos núcleos son igualmente servidas; *Justicia FIFO* cuando las peticiones son atendidas en el estricto orden en que se realizaron.

En la literatura, se pueden encontrar diferentes propuestas de solución al problema de acceso justo al bus. Una de estas propuestas, quizá la más conocida e implementada, es el arbitraje *Round Robin*. Este *Árbitro* se basa en la idea de que un núcleo que obtuvo el acceso al bus en una ronda de competencia, obtiene la prioridad más baja para la siguiente ronda de competencia. De este modo, asegura que otros núcleos obtengan el acceso al bus antes que él en subsecuentes rondas de competencia, presentando este *Árbitro* una *Justicia Fuerte*. Entre las diferentes implementaciones de este *Árbitro* que se presentan en la literatura, se encuentran las de [13], [14], [15], [8]. Aunque su comportamiento es similar, la diferencia entre estas implementaciones se encuentra en la cantidad de recursos utilizados, la potencia consumida y la frecuencia máxima de operación permitida, lo cual incide directamente en el rendimiento del sistema. Sin embargo, las métricas de desempeño utilizadas en este estudio son insensibles a la implementación de las políticas de acceso.

Otro tipo de *Árbitro* que provee un acceso justo al medio es el denominado *Queuing Arbiter* [13] que opera bajo una filosofía de prioridad tipo *FIFO (First In First Out)*. El *Árbitro* cuenta con un temporizador global que es incrementado en cada ciclo de reloj generando lo que se conoce como un *Timestamp*. Cuando un núcleo genera una petición al *Árbitro*, se genera una etiqueta que incluye el índice del núcleo que realiza la petición más el *Timestamp*, de tal forma que, en la siguiente ronda de arbitraje, el bus será asignado al elemento que tenga la etiqueta más antigua. Si un núcleo no está generando una petición, la etiqueta se genera con el valor más grande que pueda tomar el *Timestamp*, y con un valor de índice que no decodifique una señal de *Grant*, de modo que los núcleos que no están requiriendo el acceso son eliminados de la contienda.

Lottery Arbiter [16] es un *Árbitro* que otorga el acceso al bus en base a un sistema de lotería, donde cada uno de los núcleos que pretende acceder al bus tiene asignada estática o dinámicamente una cierta cantidad de boletos. Cuando es necesario contender por el bus, se genera un número aleatorio el cual corresponde con un boleto que pertenece de forma única a un núcleo. Aquel núcleo que contenga el boleto ganador es el que obtiene el derecho de acceder al bus. Aunque este *Árbitro* puede otorgar el medio en base a una prioridad y que la probabilidad de acceder al bus es creciente con el número de boletos asignados, es razonable pensar que si el número de boletos asignados a cada uno de los núcleos es igual y la función de distribución de probabilidad utilizada por el generador de números aleatorios es uniforme, entonces la probabilidad de acceso al medio de cada núcleo es la misma. Bajo las condiciones anteriormente mencionadas este *Árbitro* presenta una *Justicia Fuerte*.

E. Arquitectura del Sistema de Interconexión

La arquitectura del sistema de interconexión diseñado, trabaja bajo la técnica de comunicación *Split-Transaction*. En la Fig. 4 se muestran los elementos e interconexiones que componen esta arquitectura. Los *Maestros* son los encargados de ejecutar las tareas del sistema, éstos se apoyan en los *Esclavos* —IP Cores para cifrado de datos, compresión de datos, etc.– que ejecutan tareas específicas de forma eficiente. Los *Maestros/Esclavos* se comunican entre sí utilizando el sistema de interconexión por medio de una *Interfaz de Red (IR)*. La *IR* cumple una doble función: una, recibir paquetes de datos dirigidos al *Maestro/Esclavo* atado a ella por medio del *Receptor (Rx)*, el cual identifica si la dirección destino en el *flit* –elemento de control de flujo que compone un paquete o transacción, que es transmitido en un ciclo de reloj– de cabecera le corresponde. Si es así, almacena el paquete de datos, verifica su integridad, le indica al *Maestro/Esclavo* el arribo de un paquete y su estado, y regresa a verificar el bus para la espera del arribo de otro paquete. La otra función, enviar un paquete de datos al destino solicitado por el *Maestro/Esclavo*, el envío es efectuado por medio del *Transmisor (Tx)*. Cuando *Tx* detecta por medio de una bandera de control que hay un paquete listo para ser transmitido por el bus, activa la señal de *Solicitud* o *Request (Req)* indicando la necesidad de acceder al bus. Esta señal continua activa hasta que el *Controlador del Bus (CB)* le indica por medio de la señal de *Otorgamiento* o *Grant* que a partir de ese momento tiene el control del bus. *Tx* desactiva la señal de *Req* y activa la señal de *Sostener* o *Hold* para indicarle al *CB* que mientras esta señal se encuentre activa, la *IR* correspondiente tiene el control del bus. *Tx* transmite el paquete de datos incluyendo información para la detección de errores, cuando termina desactiva la señal de *Hold* liberando el bus para que este pueda ser usado por otro elemento en el SoC.

El *Controlador del Bus (CB)* tiene la función de gestionar el bus por medio de dos componentes: el primero es el *Árbitro*, el cual está encargado —en base a una política específica de arbitraje– de otorgar el control del bus a una de las *IR's* que lo están solicitando. El *Árbitro* realiza un ronda de arbitraje cuando no existe ninguna señal de *Hold* activa. Si esto sucede,

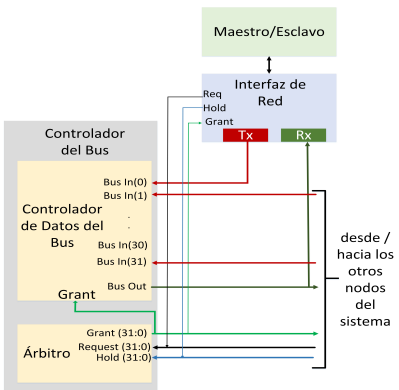


Fig. 4. Arquitectura de la interconexión.

el *Árbitro* verifica cuales *IR's* tienen la señal *Req* activa, siendo éstas las que entran a competir, y solo una de ellas es la ganadora. El *Árbitro* indica cual *IR* es la ganadora activando la señal de *Grant* asociada a ésta. A partir de este momento en cada ciclo de reloj el *Árbitro* verifica la señal de *Hold* de la *IR* a la que otorgó el control del bus; mientras esta señal se encuentre activa el *Árbitro* mantiene la señal de *Grant* asociada activa permitiéndole continuar con el control del bus. Cuando la señal de *Hold* es desactivada por la IR, el *Árbitro* vuelve a entrar en una ronda de competencia. El segundo componente es el *Controlador de Bus de Datos (CDB)*. Este es el elemento que permite que el paquete que está siendo transmitido por la *IR* que tiene el control del bus se vea reflejado en el *Bus de Salida* o *Bus Out* el cual a su vez se encuentra conectado en común a todos los *Rx* de las *IR's*. El *CDB* se comporta como un selector que por medio de la señal de *Grant* activa, genera una ruta entre el *Bus In* y el *Bus Out* del sistema, permitiendo con esto la comunicación entre dos o más nodos de la red.

III. Modelo del Sistema

Como fue mencionado anteriormente, *Split-Transaction* permite aprovechar con mayor eficiencia el bus gracias a que este es liberado entre el momento que un paquete es mandado por un maestro y el momento en el que se obtiene la respuesta por parte del esclavo. A este periodo se le denomina tiempo de ejecución t_e. El tiempo que un paquete tarda en ser transmitido se denomina t_p. Bajo la suposición de que todos los elementos en el sistema son del mismo tipo, se puede decir que en el tiempo de ejecución pueden ser transmitidos IP_{max} paquetes de tal modo que:

$$IP_{max} = \lfloor t_e/t_p \rfloor \tag{1}$$

Este valor indica el máximo número de paquetes/transacciones que pueden ser transmitidos en t_e sin saturar el bus. Si todos los maestros se encuentran transmitiendo paquetes del mismo tipo, entonces IP_{max} es el número máximo de maestros que pueden ser conectados al sistema, antes de que este entre en saturación.

La transmisión de un paquete en un ciclo de reloj está determinada por un proceso de Bernoulli cuya probabilidad de transmisión se denomina λ. La razón promedio λ con la que un maestro requiere de los servicios de un esclavo para realizar una operación, puede variar por diferentes circunstancias. Este comportamiento aleatorio afecta de forma significativa al rendimiento del bus. La razón λ es denominada tasa promedio de inyección de paquetes, la cual queda dada por $\lambda = \rho \cdot \mu$ donde $0 < \rho < 1$. Por su parte μ se encuentra definida por $\mu = 1/t_r$, donde t_r es el tiempo de respuesta, que se define como el tiempo que tarda un paquete en ser procesado cuando existen las mejores condiciones del sistema, es decir, tanto el maestro como el esclavo encuentren libre el medio cuando lo necesiten. Este tiempo se mide desde que un maestro envía el paquete con los datos a ser procesados hasta el momento en que tiene almacenado en su memoria de entrada el resultado enviado por parte del esclavo. t_r queda definido por:

$$t_r = t_{pMaestro} + t_{pEsclavo} + t_e \tag{2}$$

donde: $t_{pMaestro}$ es el tiempo de transmisión de un paquete del maestro en ciclos de reloj; $t_{pEsclavo}$ es el tiempo de transmisión de un paquete del esclavo en ciclos de reloj; y t_e es el tiempo de ejecución del esclavo en ciclos de reloj. El tiempo de transmisión del paquete tanto en el maestro como en el esclavo se calcula en términos del número de flits de datos que tiene el paquete más los flits de encabezado y cola. El tiempo de ejecución t_e es calculado en base a la latencia de cada algoritmo y es suministrado al sistema en términos de ciclos de reloj por paquete, es decir, el número de ciclos de reloj que tardará un paquete en ejecutarse.

A. Modelo en SystemC

Con la finalidad de observar el comportamiento del sistema de bus tipo *Split-Transaction* utilizando diferentes políticas de arbitraje, se implementó en SystemC la arquitectura propuesta con precisión de ciclo de reloj. Se efectuaron tres implementaciones, una para cada política de arbitraje diferente. Es importante mencionar, que entre estas implementaciones el único cambio se encuentra en el *Árbitro*, el cual cambia la política de arbitraje. Los parámetros que pueden ser modificados para cada uno de los maestros son: tamaño del paquete de datos; tasa de inyección de paquetes λ. Los parámetros que pueden ser configurados en los esclavos son: tiempo de ejecución t_e; tamaño del paquete de respuesta del esclavo. La estadística que se recolecta por núcleo es: número de paquetes procesados; tiempo promedio en el *buffer* de salida. Las políticas de arbitraje que fueron implementadas son: *Round Robin*; *FIFO*; y *Lottery*. Aunque en la literatura existen diferentes implementaciones de *Round Robin*, se decidió implementar la propuesta por [14] ya que ésta asegura una *Justicia Fuerte*, además el tipo de implementación *Round Robin* no impacta en el comportamiento que en este estudio se mide. El sistema soporta hasta un máximo de 16 nodos del tipo Maestro y 16 nodos del tipo Esclavo.

IV. Análisis de Resultados

Para observar el comportamiento de las políticas de arbitraje se efectuaron pruebas bajo diferentes escenarios. Estos escenarios utilizan IP Cores con diferentes tiempos de ejecución (t_e), y paquetes de diferente tamaño, tanto de datos como de resultados. La relación IP_{max} influye en el comportamiento del sistema de bus, de tal modo que se plantearon núcleos con diferentes valores. La Tabla I muestra los IP_{max} utilizados en las pruebas. Se emplea tráfico sintético además del generado por la aplicación de procesamiento de imágenes satelitales descrita en la sección II. Para dar certeza estadística se promediaron 10 realizaciones de cada experimento realizado.

Con la finalidad de comprobar el máximo número de maestros que pueden trabajar simultáneamente antes de saturar el bus, se creó un escenario con IP Cores del mismo tipo. Se evaluó el número de paquetes procesados, suponiendo que el maestro siempre tiene un paquete por enviar ó $\lambda = 1$. La prueba se realiza modificando el número de maestros que tiene el sistema. Se encontró que el tipo de arbitraje no influye en el número de maestros que saturan el bus, por lo anterior, sólo se presentan los resultados de la política *Round Robin*. Las

TABLA I
PARÁMETROS DE LOS IP CORES.

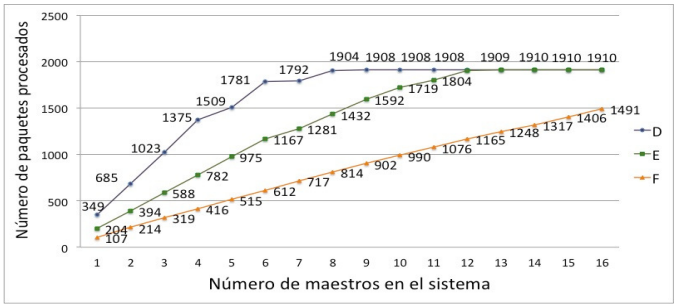
Relación	t_e (ciclos)	$t_{pMaestro}$ (ciclos)	IP_{max} calculado	IP_{max} simulado
A	14	4	3	3
B	26	4	6	5
C	38	4	9	6
D	1024	128	8	7
E	2048	128	16	11
F	4096	128	32	No Satura
G	386	64	6	–
H	16384	128	128	–

pruebas se realizaron para IP_{max} = A, B, C, D, E, y F, como muestra la Fig. 5. En todos los casos el IP_{max} calculado es mayor o igual que el obtenido en la simulación. Se observa que para un mismo t_p –Maestro–, si la relación t_e/t_p es mayor, el valor de saturación IP_{max} simulado se aleja del calculado. Esto se debe a que transacciones de mayor tamaño causan mayor contención en el bus.

Otro de los objetivos del estudio es comparar el comportamiento de las políticas de arbitraje *Round Robin*, *Lottery* y *FIFO* en escenarios homogéneos y heterogéneos, en términos de capacidad de procesamiento (bits/ciclo) y tiempo promedio de espera en la memoria de salida (ciclos) de las *IR's*. En todas las pruebas realizadas la tasa de inyección de paquetes varía en la misma proporción para cada uno de los maestros. Para el escenario homogéneo, en la Fig. 6a, se muestra el comportamiento de las tres políticas de arbitraje para un sistema trabajando con doce maestros con una IP_{max} = 1024/128. Se puede observar que las diferentes políticas, con respecto a la capacidad de procesamiento, tienen un comportamiento similar para cualquier valor de λ. Lo mismo puede decirse, observando la Fig. 6b, con respecto al tiempo promedio de espera en la memoria de salida de la *IR*. Para la prueba de elementos heterogéneos con diferentes políticas de arbitraje, se seleccionó un escenario empleando doce elementos –cuatro de cada relación: A, C y G–. En la Fig. 7a se puede observar que el tiempo promedio de espera en la memoria de salida de las *IR*'s es similar entre las políticas de asignación evaluadas. El mismo comportamiento puede ser observado en la Fig. 7b para la capacidad de procesamiento entre las políticas analizadas con los tres tipos de IP Cores utilizados. Sin embargo, conforme aumenta la tasa promedio de inyección de paquetes se va generando una disparidad en la capacidad de procesamiento entre los tipos de IP Cores, posesionándose del bus los paquetes más grandes.

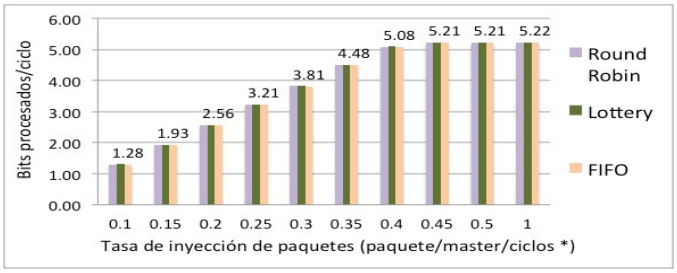
Para analizar con detalle la equidad en la tasa de transmisión –bits/ciclo transmitidos en el bus– o capacidad de procesamiento –bits/ciclo procesados por el sistema– se crearon dos casos heterogéneos: uno con tráfico sintético y otro basado en una aplicación aeroespacial. El escenario de tráfico sintético se creó con cuatro IP Cores de cada relación A, B, y G, para tener un total de doce núcleos. Las pruebas fueron realizadas para las tres políticas de arbitraje seleccionadas; los resultados en los tres casos son similares; y por motivos de espacio sólo se presentan los resultados de la evaluación para la política de arbitraje *Round Robin*. En la Fig. 8 se denomina Homogéneo al escenario donde sólo compiten por el bus los cuatro IP Cores de cada tipo, respectivamente, y Heterogéneo cuando compiten todos los diferentes tipos de IP Cores. Para el caso de la aplicación aeroespacial, se emplearon cuatro IP Cores para filtrado de imágenes, cuatro para la compresión de imágenes y cuatro para el cifrado de datos. El compresor de imágenes mejora su rendimiento conforme procesa mas líneas de la imagen, ya que incrementa la información de contexto, por lo que al inicio su razón de compresión es baja y mejora

(a) Paquetes procesados con IP_{max} = A, B y C.

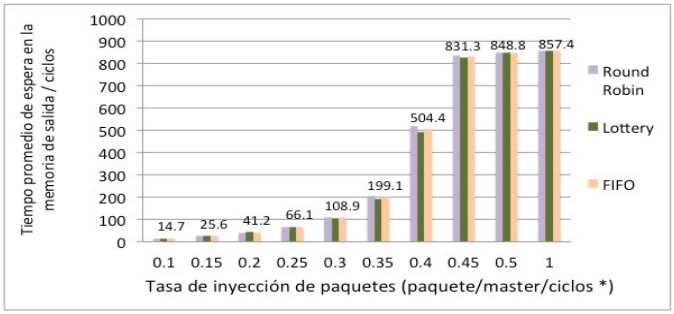


(b) Paquetes procesados con IP_{max} = D, E y F.

Fig. 5. Saturación del bus cuando se tiene un número IP_{max} de maestros en el SoC.

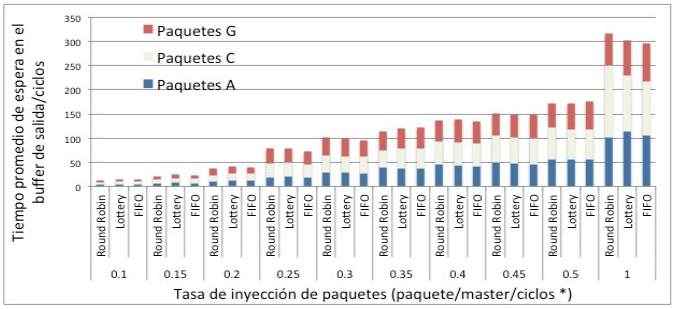


(a) Capacidad de procesamiento para diferentes políticas del bus.

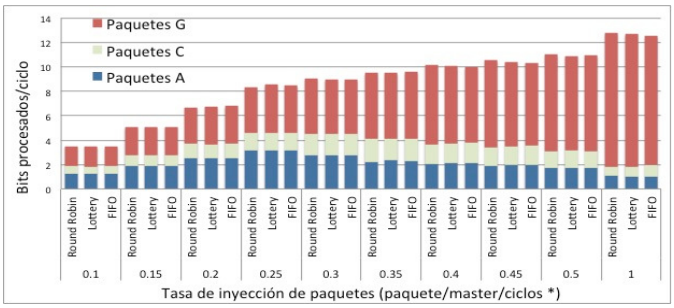


(b) Tiempo de acceso para diferentes políticas del bus.

Fig. 6. En sistemas homogéneos, las métricas de rendimiento no difieren significativamente al comparar buses con diferentes políticas de arbitraje justo, y aumentar la tasa de inyección de paquetes.

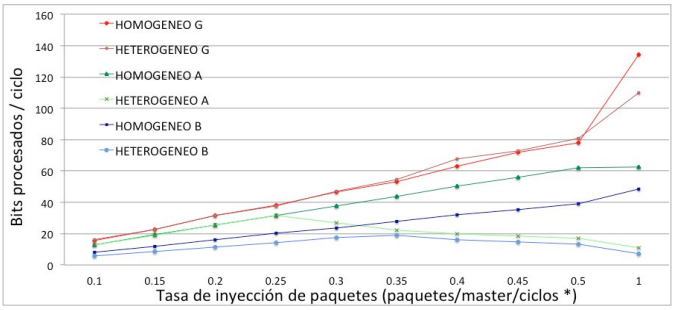


(a) Tiempo de acceso al bus similar entre los diferentes tipos de IP Cores.

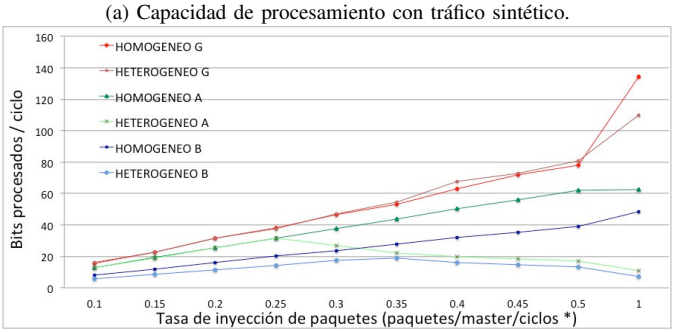


(b) Capacidad de procesamiento desigual entre los diferentes tipos de IP Cores.

Fig. 7. Las diferentes políticas analizadas logran equidad en el acceso al bus pero disparidad en los bits procesados por cada tipo de IP Cores.



(a) Capacidad de procesamiento con tráfico sintético.



(b) Capacidad de procesamiento en la aplicación aeroespacial.

Fig. 8. Disminución de la capacidad de procesamiento cuando existen diferentes tamaños de paquetes, independientemente del escenario, apoderándose del bus los IP Cores con los paquetes más grandes.

conforme el proceso avanza. Aquí se supone que la razón de compresión siempre es del 50 %, es decir, el paquete de respuesta es la mitad del paquete de datos recibido. La relación IP_{max} de cada uno de ellos es: *Filtrado*$\rightarrow D$, *Compresión* $\rightarrow H$ y *Cifrado* $\rightarrow A$.

La Fig. 8a muestra que en valores pequeños de λ los tres tipos de IP Cores atienden una cantidad similar al sistema homogéneo, ya que el bus se encuentra lo suficientemente holgado para procesar todos los paquetes. Sin embargo, para λ =0.25 el IP Core A baja su rendimiento debido a que los otros dos tipos de elementos demandan más capacidad del bus. Se observa un comportamiento similar para el tipo de IP Cores B en $\lambda = 0.35$. Sin embargo, para el tipo de elemento G, no hay saturación, sólo una baja en su rendimiento cuando λ =1.0; esto es debido al ancho de bus que ocupan los otros tipos de IP Cores. Con respecto al tráfico de la aplicación aeroespacial, en la Fig. 8b se observa que los tres tipos de IP Cores trabajan a su capacidad hasta una λ =0.35, donde los elementos de cifrado se saturan y a partir de aquí su rendimiento decrece notablemente hasta llegar a ser una tercera parte para λ =1. Se observa que el IP Core de compresión tiene un rendimiento acorde a su capacidad, dado que es el elemento que genera las transacciones más largas. El IP Core de filtrado trabaja a su capacidad hasta una λ =0.4, a partir de aquí su rendimiento se ve disminuido con respecto a su capacidad, sin embargo sigue en constante crecimiento. De lo anterior se puede observar que los IP Cores con transacciones más grandes —*Filtro* y *Compresor*– consumen mayor ancho de banda en detrimento de los IP Cores con transacciones más cortas —*Cifrador*–, esto provoca una inequidad en la capacidad de procesamiento –o transmisión de paquetes en el bus– entre los diferentes IP Cores.

V. Conclusiones y Trabajo Futuro

En este trabajo se realizaron pruebas para comprobar que el máximo número de IP Cores de un tipo en particular que pueden trabajar en un sistema de bus tipo *Split-Transaction* está dado por la ecuación (1). De los resultados obtenidos se puede concluir que este valor es un máximo ya que por problemas de contención cuando la relación IP_{max} es grande no se alcanza este valor. Sin embargo, puede servir como una buena aproximación. Aumentar el número de IP Cores mas allá del valor de IP_{max}, no mejora el rendimiento y sólo aumenta la cantidad de recursos y energía consumidos. Por otro lado, se comprobó que las tres políticas de arbitraje evaluadas se comportan similarmente, en términos de rendimiento y tiempo promedio de espera en la memoria de salida de las *IR*'s. Con las pruebas realizadas también se comprobó, que aunque los árbitros evaluados son justos en términos de acceso al medio, no lo son en rendimiento. Lo anterior puede convertirse en un problema, cuando una aplicación requiera que el sistema necesite que los tiempos de procesamiento de los diferentes tipos de núcleos, tengan que generar tiempos de entrega específicos, para asegurar calidad en el servicio.

Las políticas de arbitraje que fueron probadas aquí, incluyendo *Lottery*, con una asignación de boletos igual para los núcleos que están en el bus, muestran que no es posible asegurar equidad en la capacidad de transmisión para los elementos que integran el sistema. Es necesario investigar otras políticas de arbitraje, que permitan asignar diferentes pesos a los núcleos para lograr esta equidad. En el futuro, se evaluarán políticas de arbitraje como *Lottery* o *Weighted Round Robin*, modificando sus boletos o pesos, para intentar lograr este objetivo. También debe de estudiarse mecanismos que permitan, en tiempo real, medir el rendimiento del sistema, para verificar

que se cumple con los requerimientos de calidad en el servicio, en términos de equidad en la capacidad de transmisión, y que permita realizar ajustes si estos requerimientos no son alcanzados.

AGRADECIMIENTOS

Los autores agradecemos a los revisores por sus valiosas sugerencias; y al Fondo ANR-CONACYT 2015-2016 por el apoyo al proyecto "TOLTECA: Climate Monitoring and Disaster Prevention using a Reconfigurable Satellite Communication System", con número 273562 para el desarrollo del presente trabajo.

REFERENCIAS

[1] F. Poletti, D. Bertozzi, L. Benini, and A. Bogliolo, "Performance Analysis of Arbitration Policies for SoC Communication Architectures," *Design Automation for Embedded Systems*, vol. 8, no. 2/3, pp. 189–210, jun 2003.

[2] M. Slijepcevic, C. Hernandez, J. Abella, and F. J. Cazorla, "Design and implementation of a fair credit-based bandwidth sharing scheme for buses," in *Proceedings of the 2017 Design, Automation and Test in Europe, DATE 2017*. Institute of Electrical and Electronics Engineers Inc., may 2017, pp. 926–929.

[3] F. E. Guibaly, "Design and Analysis of Arbitration Protocols," *IEEE Transactions on Computers*, vol. 38, no. 2, pp. 161–171, 1989.

[4] I. Ben-Hafaiedh, S. Graf, and M. Jaber, "Model-based design and distributed implementation of bus arbiter for multiprocessors," in *2011 18th IEEE International Conference on Electronics, Circuits, and Systems, ICECS 2011*, 2011, pp. 65–68.

[5] A. M. and A. A., "A Bus Arbitration Scheme with an Efficient Utilization and Distribution," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 3, 2017.

[6] Y. Yang, R. Wu, L. Zhang, and D. Zhou, "An asynchronous adaptive priority round-robin arbiter based on four-phase dual-rail protocol," *Chinese Journal of Electronics*, vol. 24, no. 1, pp. 1–7, jan 2015.

[7] P. J. Bellardo, "CubeSat," 2020. [Online]. Available: https://www.cubesat.org

[8] M. Oveis-Gharan and G. N. Khan, "Index-based round-robin arbiter for NoC routers," in *Proceedings of IEEE Computer Society Annual Symposium on VLSI, ISVLSI*, vol. 07-10-July. IEEE Computer Society, oct 2015, pp. 62–67.

[9] K. Jain, S. K. Singh, A. Majumder, and A. J. Mondai, "Problems encountered in various arbitration techniques used in NOC router: A survey," in *2015 International Conference on Electronic Design, Computer Networks and Automated Verification, EDCAV 2015*. Institute of Electrical and Electronics Engineers Inc., mar 2015, pp. 62–67.

[10] M. Pagani, E. Rossi, A. Biondi, M. Marinoni, G. Lipari, and G. Buttazzo, "A Bandwidth Reservation Mechanism for AXI-Based Hardware Accelerators on FPGAs," in *31st Euromicro Conference on Real-Time Systems (ECRTS 2019)*, S. Quinton, Ed. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, pp. 24:1—-24:24.

[11] M. Hernandez-Calvino, S. Ibarra-Delgado, R. Sandoval-Arechiga, J. Flores-Troncoso, and L. Garcia-Luciano, "Image Compressor IP-Core based on LOCO Algorithm for Space Photography Application," in *2018 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*. IEEE, nov 2018, pp. 1–4.

[12] L. Garcia-Luciano, S. Ibarra-Delgado, H. Gallegos-Ruiz, and R. Sandoval-Arechiga, "Hardware implementation of a block cipher with AXI Stream Interface," *Memorias del Congreso Internacional de Investigación Academia Journals Celaya 2017*, vol. 9, no. 6, pp. 2245,2251, 2017.

[13] W. J. Dally and B. P. Towles, *Principles and practices of interconnection networks*. Elsevier, 2004.

[14] S. Q. Zheng and M. Yang, "Algorithm-hardware codesign of fast parallel round-robin arbiters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 1, pp. 84–95, 2007.

[15] Y.-L. Lee, J. M. Jou, and Y.-Y. Chen, "A high-speed and decentralized arbiter design for NoC," in *2009 IEEE/ACS International Conference on Computer Systems and Applications*. IEEE, may 2009, pp. 350–353.

[16] K. Lahiri, A. Raghunathan, and G. Lakshminarayana, "The LOTTERYBUS on-chip communication architecture," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 6, pp. 596–608, jun 2006.

Salvador Ibarra-Delgado obtuvo el grado de Ingeniería en Sistemas Electrónicos por el Instituto Tecnológico y de Estudios Superiores de Monterrey, Monterrey, Nuevo León, México, en el año 1986, Obtuvo el grado de Maestría en Informática y Tecnologías Computacionales por la Universidad Autónoma de Aguascalientes, Aguascalientes, México, en el año 2008. Actualmente es profesor investigador en la Universidad Autónoma de Zacatecas, Zacatecas, México. Sus principales áreas de interes son el desarrollo de sistemas embebidos y núcleos de propiedad intelectual para telecomunicaciones y procesamiento de imágenes.

Remberto Sandoval-Arechiga obtuvo los grados de Ingeniero en Comunicaciones y Electrónica por la Universidad Autónoma de Zacatecas (UAZ) en el 2002, Maestría y Doctorado en Ciencias con especialidad en telecomunicaciones en el CINVESTAV del IPN en Zacatenco (2006) y Guadalajara (2016), respectivamente. Actualmente es profesor investigador en el Centro de Investigación y Desarrollo en Telecomunicaciones (CIDTE) de la UAZ. Sus intereses de investigación son las telecomunicaciones, los SoC's y las redes en chip. El Dr. Sandoval-Arechiga es miembro del IEEE en las sociedades *ComSoc*, *Computer* y *CAS*.

María Brox obtuvo el grado de Ingeniería en física por la Universidad de Córdoba, Córdoba, España, en el año 2004, el grado de Doctor en microelectrónica lo obtuvo en el año 2013, por la Universidad de Sevilla, Sevilla, España. Del año 2005 al año 2007 obtuvo una beca de posgrado por parte del gobierno de España, realizando una estancia e el Instituto de Microelectrónica de Sevilla (IMSE-CNM-CSIC), Sevilla, España. Actualmente es profesora asistente en el Departamento de Ingeniería en Electrónica y Computadores, en la Universidad de Córdoba, Córdoba, España. Su área de interés es el desarrollo de herramientas CAD para el diseño de controladores difusos embebidos en FPGA's.

Manuel Agustín Ortíz López Profesor Titular del Departamento de Ingeniería Electrónica y de Computadores de la Universidad de Córdoba. Doctor por la Universidad de Córdoba desde diciembre de 2013, Licenciado en Ciencias Físicas, especialidad en Electrónica, desde julio de1987. Profesor de la Universidad de Córdoba desde 1996, trabajó en el departamento de I+D de Fujitsu España, en el departamento de I+D de Tecosa (grupo Siemens) y en el departamento de Comunicaciones Informáticas de Telefónica de España. Su labor de Investigación está relacionada con el desarrollo de sistemas empotrados y en HW/SW codesing.