Integración Hw/Sw para aplicaciones médicas basadas en el estándar OpenMAX.

David de la Fuente, Julio D. Dondo, Jesús Barba, Julián Caba and Juan Carlos López
University Of Castilla-La Mancha
Ciudad Real, Spain
{*David.Fuente, JulioDaniel.Dondo, Jesus.Barba, Julian.Caba, JuanCarlos.Lopez*}*@uclm.es*

La utilización de recursos Hw es una técnica muy común cuando se requiere acelerar partes de un sistema con fuertes exigencias temporales o computacionales. Un campo en el que cada vez se demandan más estas exigencias es el de la medicina. Por este motivo, los fabricantes implementan en Hw algoritmos de procesamiento utilizados en las aplicaciones médicas con el fin de mejorar su rendimiento. Por normal general, estas implementaciones no siguen ningún estándar de desarrollo y suelen hacerse a medida, lo que limita la capacidad de reutilización, portabilidad e integración del desarrollo.

A través de la plataforma de integración Hw/Sw presentada en este trabajo, se ofrece la posibilidad de integrar los diversos IPs o cores en una cadena de procesamiento basada en el estándar OpenMAX, el cuál ha sido adaptado y migrado a las necesidades específicas de los sistemas empotrados. Un IP implementado sobre esta plataforma se comunicará con el resto del sistema de manera transparente, homogénea y descentralizada, con una mínima sobrecarga añadida en la infraestructura de integración.

1. Introducción

Cada vez es más evidente la aplicación de técnicas de procesamiento multimedia en el campo de la medicina. Aplicaciones destinadas a la adquisición y tratamiento de señales biomédicas, al procesado de imágenes médicas de alta resolución o las de apoyo a las discapacidades sensoriales son un claro ejemplo. Las exigencias computacionales y temporales asociadas a estas aplicaciones se han incrementado notablemente, hasta el punto de tener aplicaciones como las de monitorización o las de cirugía asistida por computador cuya ejecución en tiempo real es obligada. Esto ha dado lugar a sistemas heterogéneos Hw/Sw en los que, con el fin mejorar su rendimiento, se han implementado en Hw las partes más críticas del sistema a modo de coprocesadores, *intellectual property (IP)* o aceleradores [1] [2].

Generalmente, la encapsulación de diversos IPs en los sistemas heterogéneos es algo habitual pero debido a que suelen ser de diferentes fabricantes aparecen problemas de incompatibilidad y de integración. A causa de la escasa utilización de estándares de desarrollo multimedia en muchas facetas de la integración y en el desarrollo del Hw para el procesado multimedia, los diseños suelen implementarse de manera ad-hoc, dificultando en gran medida la reutilización y la portabilidad de los sistemas. Dado que en los SoCs las arquitecturas de bus son las más extendidas para comunicar diversos bloques funcionales, la tendencia actual para facilitar la reutilización e integración de IPs pasa por estandarizar la manera de acceder a su funcionalidad. Típicamente, esto se consigue estandarizando los buses de comunicación a los que están conectados los IPs (ARM AMBA^{TM}o IBM CoreConnect^{TM}) o bien definiendo un envoltorio que independice la parte de comunicación de su comportamiento interno [3][4].

Basándose en lo anterior, este trabajo presenta una plataforma de integración Hw/Sw para sistemas de procesamiento multimedia y que adicionalmente proporciona la infraestructura y los mecanismos necesarios para incorporar a los IPs a un flujo de aplicación estandarizado basado en OpenMAX [5]. Esta plataforma hará uso del middleware de comunicación *Object Oriented Communication Engine (OOCE)* [6] para ofrecer una gestión trasparente de la comunicación y aumentar nivel de abstracción en el que se define el sistema con el fin de desacoplar el campo de aplicación de los detalles inherentes al diseño.

El resto del documento se estructura de la siguiente manera: en la sección 2, se ofrece una visión general de la propuesta haciendo hincapié en la manera de trasladar el estándar OpenMAX al ámbito de los sistema empotrados y cómo permitir una fácil integración de elementos de procesamiento Hw/Sw. La visión de la plataforma por parte del programador se detalla en la sección 3. Para finalizar, se enumerarán una serie de conclusiones en la sección 4.

2. Propuesta

Nuestra propuesta se basa en el uso de las FPGAs y el estándar OpenMAX para el diseño e implementación de sistemas multimedia heterogéneos flexibles y portables en un corto espacio de tiempo. OpenMAX es un estándar abierto, desarrollado por el grupo Khronos [1] cuya meta es reducir el coste y la complejidad de migrar Sw multimedia entre múltiples sistemas operativos y plataformas. Una aplicación típica de OpenMAX está compuesta por un conjunto de *componentes OpenMAX* que encapsulan parte de la funcionalidad del sistema y forman una cadena de procesamiento. Tras el análisis realizado en [7] de las tres capas que componen OpenMAX, se aprecia que la capa de integración (*Integration Layer (IL)* representada en la figura 1) ejerce las funciones de un middleware como la carga y descarga dinámica de componentes OpenMAX, inicialización, conexión de los componentes, gestión de la sincronización y de la comunicación etc.

La implementación Hw de ciertos elementos del la

[1] `https://www.khronos.org/`

capa IL puede repercutir notablemente en el rendimiento final del sistema. Un candidato a ser desarrollado en Hw es el concepto de componente OpenMAX ya que al encapsular funciones de procesado puede requerir mayores capacidades computacionales. A la implementación Hw de un componente OpenMAX se denominará componente OpenMAX Hw. Dicha implementación lleva asociada la necesidad de proporcionar una conectividad estandarizada del componente con resto del sistema, obligando a implementar los mecanismos de comunicación entre componentes descritos en el estándar OpenMAX.

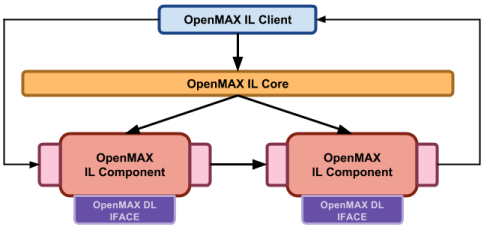

Figura 1: Estructura de OpenMAX Integration Layer.

Para tener una visión global de la propuesta, la figura 2 representa una cadena de procesamiento OpenMAX utilizando el enfoque de nuestra plataforma de integración Hw/Sw. En el ejemplo, los componentes B y C están asignados a una implementación Hw de un componente OpenMAX. En la plataforma, la comunicación a nivel Sw será llevada a cabo respetando los protocolos y mecanismos descritos la implementación de referencia de OpenMAX. Los componentes destinados a ser implementados en Hw (B y C) forman una sub-cadena de componentes e intercambiarán buffers (mínima unidad de intercambio de datos en OpenMAX) siguiendo la filosofía del modelo de comunicación tunelada definida en el estándar. Por el hecho de ser una comunicación directa entre componentes sin la intervención de elementos adicionales, se ha elegido el modelo tunelado por considerar que es el más eficiente de los modelos que ofrece el estándar.

Desde el punto de vista del usuario de la aplicación, gracias a la utilización de OOCE la integración de los componentes OpenMAX Hw en el sistema es completamente transparente.

2.1. Componentes OpenMAX Hw

Un componente OpenMAX Hw (figura 3) está compuesto por dos partes bien diferenciadas con el fin de independizar la parte de computación del componente de la parte de comunicación, aumentando sus posibilidades de reutilización tal y como se ha visto en la introducción. Por un lado se encuentra el denominado *Hw Media Core (HMC)* que encapsula la implementación Hw del algoritmo de procesamiento multimedia de esa etapa de la aplicación, y por otro lado está el adaptador para dicho HMC (a partir de

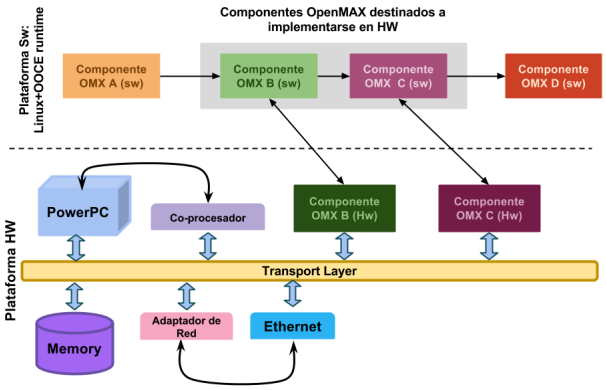

Figura 2: Visión general de la plataforma de integración Hw/Sw.

ahora *OpenMAX Hardware Adapter, OHA*) que aparte de estandarizar el acceso al componente será el responsable de gestionar la comunicación y los datos de entrada y salida.

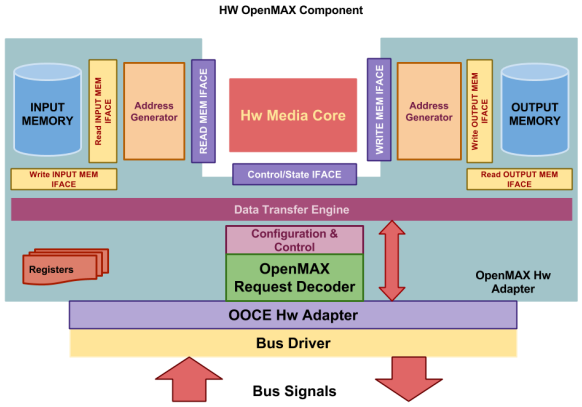

Figura 3: Micro-Arquitectura del Componente OpenMAX Hw.

Se ha fijado una sencilla interfaz entre el HMC y el OHA que proporciona al HMC la independencia de la tecnología de comunicación (canal, protocolo y sincronización) y permite gestionar y controlar su ejecución. A través de esta interfaz, el HMC ofrece información relativa a su estado de ejecución al OHA el cual, en función de dicho estado, activará las señales de control pertinentes para la gestión del HMC (inicio, pausa, configuración, etc). A parte de esta interfaz de control, el HMC interactúa con el OHA por medio de una interfaz estandarizada de memoria compuesta por un puerto de lectura y otro de escritura. De esta forma, el HMC también permanece ajeno a las tecnologías de memoria utilizadas, aumentando de nuevo el grado de reutilización. Siempre que se respete la interfaz con el OHA, cualquier algoritmo o IP con capacidad de procesar datos a su entrada y generar nuevos datos a su salida es candidato a ser encapsulado en el HMC y pasar a formar parte de un flujo de aplicación OpenMAX.

Los elementos predominantes en el OHA son las me-

morias locales de entrada y salida. Desde el punto de vista del OHA, el uso de las memorias se reduce a escribir datos en la memoria local de entrada (que posteriormente serán leídos por HMC) y leer los datos generados por el HMC de la memoria local de salida que más tarde serán transmitidos. La independencia de la tecnología subyacente de las memorias se logra por medio del uso de la implementación Hw del patrón iterador [8] por parte del HMC y del OHA.

Otro elemento destacable dentro del OHA, es el *Data Transfer Engine (DTE)*, que se encarga de controlar el estado de las memorias locales y gestiona el proceso de trasmisión de datos con el resto de componentes de la cadena. Este proceso es controlado de manera autónoma por el DTE, sin la intervención de rutinas Sw, convirtiéndose de esta manera, en un proceso completamente descentralizado y transparente al HMC, con la ventaja añadida de tener liberado al microprocesador de la tarea de transferir los datos entre componentes OpenMAX Hw.

La parte de acceso al bus de comunicaciones es responsabilidad de los adaptadores de OOCE que implementan una semántica de *Invocación a Método Remoto (RMI)*. Son los encargados de transformar los datos que provienen del canal a llamadas a los métodos presentes en la interfaz del OHA. Estos métodos son el conjunto de primitivas derivadas del estándar OpenMAX relativas a la configuración y ejecución del componente y a la gestión de la comunicación con otros componentes OpenMAX (Hw o Sw) del sistema. Gracias a la utilización de los adaptadores de OOCE, el componente OpenMAX Hw representa un **objeto** en el sistema y puede utilizar los mecanismos de comunicaciones definidos en OOCE, facilitando una integración Hw/Sw completamente transparente al usuario.

2.2. OOCE como herramienta de integración

Una de las soluciones más extendidas cuando se trata de comunicar una gran variedad de elementos heterogéneos (aplicaciones, redes, Sw, Hw ...) es la utilización de un middleware. En este trabajo, se ha optado por usar un middleware desarrollado en la Universidad de Castilla la Mancha denominado *Object Oriented Communication Engine (OOCE)*. Basándose en el paradigma de objetos distribuidos, OOCE proporciona una visión unificada de todo el sistema donde todos los elementos (Hw o Sw) son considerados objetos y pueden ser invocados a través de una interfaz de objeto bien conocida. La especificación de dicha interfaz es totalmente independiente de la tecnología de comunicación, permitiendo homogeneizar la comunicación Hw/Sw.

OOCE define el modelo de comunicación cliente/servidor con varias extensiones para soportar la semántica RMI anteriormente mencionada. Las invocaciones entre objetos clientes y servidores se representan por medio de mensajes, los cuales contienen toda la información necesaria para alcanzar su destino, los parámetros de la invocación y la dirección de retorno si fuera necesario.

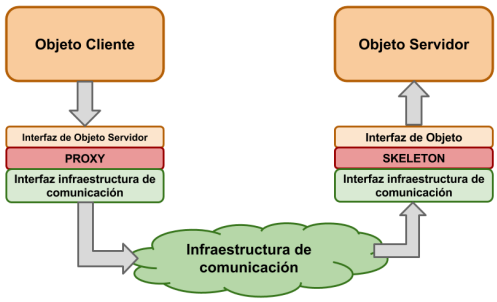

Figura 4: Actores en una invocación a método remoto (RMI).

Para lograr establecer una comunicación basada en RMI es necesario la presencia de dos elementos adaptadores esenciales: un proxy y un skeleton. Como se observa en la figura 4, el proxy es una entidad que representa al objeto servidor y se sitúa entre el objeto cliente y el medio de transporte. Por su parte, el skeleton suplanta al objeto cliente en el servidor y es el encargado de recoger los mensajes generados por el proxy y generar la invocación en el servidor una vez hayan sido decodificados (ver figura 5). La clave está en proporcionar a los objetos que intervienen en la comunicación la ilusión de tener una conexión directa entre ellos.

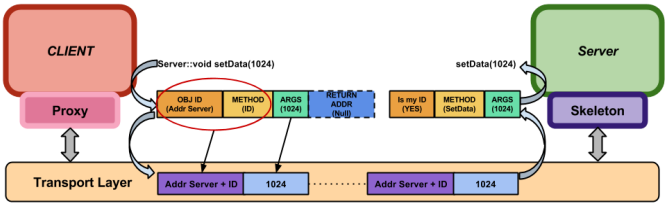

Figura 5: Transformación de invocación remota a transacción por el canal de comunicación y viceversa.

Siguiendo está filosofía, tal y como se muestra en la figura 6, OOCE da soporte a los tipos de comunicación (Hw/Hw, Hw/Sw y Sw/Hw) necesarios para poder integrar un componente OpenMAX Hw en un flujo de aplicación OpenMAX.

3. Visión de la plataforma por parte del desarrollador

Para que la inclusión de un nuevo componente OpenMAX Hw no afecte al resto de componentes presentes en el sistema, se ha definido una *"fachada"* Sw del componente Hw que entienda OpenMAX al 100 %. En realidad, esta fachada es una versión de un componente OpenMAX Sw obtenido de la implementación de referencia del grupo Khronos libre de carga de procesamiento. Además de poseer una interfaz 100 %

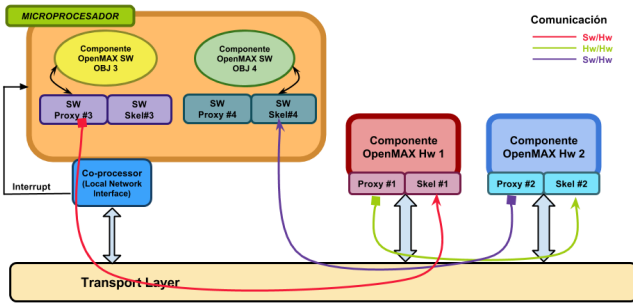

Figura 6: Ejemplo de los distintos tipos de comunicación en OOCE.

OpenMAX y no tener capacidad de procesamiento, la fachada del componente OpenMAX Hw realiza funciones de representación y únicamente redirige al componente OpenMAX Hw asociado las primitivas del estándar relativas a su configuración y sincronización. Estas primitivas son transformadas por su fachada a invocaciones remotas a método de la interfaz del componente Hw. De esta forma, la integración de un componente OpenMAX Hw es completamente transparente al resto del sistema y compatible al 100 % con el estándar OpenMAX. Una vez configurados y puestos en funcionamiento, los componentes OpenMAX Hw son lo suficientemente autónomos como para poder comunicarse entre ellos sin la intervención de sus fachadas.

El ejemplo del código fuente 1 trata de ilustrar el uso de los componentes OpenMAX Hw integrados de manera transparente en una cadena de procesamiento formada por componentes OpenMAX. La aplicación está destinada al procesamiento de imagen y va a estar formada por cuatro componentes OpenMAX (ETH_READER, RBG2BW, SOBEL y ETH_SINK) de los cuales dos (RBG2BW y SOBEL) van a ser implementados como componentes OpenMAX Hw. La aplicación OpenMAX consiste en recibir imágenes a través de un puerto ethernet, convertir las imágenes a blanco y negro, aplicar un filtro de detección de bordes y finalmente enviar las imágenes procesadas de nuevo por la ethernet.

```
int main(int argc, char** argv) {
 OMX_Init();

 /*Getting Components Handler*/
 OMX_GetHandle(&appPriv->ethreaderhandle, "
     ETH_READER" NULL, &READERcallbacks);
 OMX_GetHandle(&appPriv->rgb2bwhandle, "
     RBG2BW", NULL, &RGB2BWcallbacks);
 OMX_GetHandle(&appPriv->sobelhandle, "SOBEL
     ", NULL, &SOBELcallbacks);
 OMX_GetHandle(&appPriv->ethsinkhandle, "
     ETH_SINK", NULL, &SINKcallbacks);

 /* Set basic parameters for Components*/
 OMX_SetParameter(appPriv->ethreaderhandle,
     OMX_IndexParam, eth_reader_param_list);
 OMX_SetParameter(appPriv->rgb2bwhandle,
     OMX_IndexParam, rgb2rw_param_list);
 OMX_SetParameter(appPriv->sobelhandle,
     OMX_IndexParam, sobel_param_list);
 OMX_SetParameter(appPriv->ethsinkhandle,
     OMX_IndexParam, eth_sink_param_list);

 /*Set the size for img Hw OMX Components*/
 sSize.sWidth.nValue = 640;
 sSize.sHeigth.nValue = 480;
 OMX_SetConfig(appPriv->rgb2bwhandle,
     OMX_IndexConfigImgSize, &sSize);
 OMX_SetConfig(appPriv->sobelhandle,
     OMX_IndexConfigImgSize, &sSize);

 OMX_UseBuffer(appPriv->rgb2bwhandle, &
     RGB2BWinBuffer0, 0 , NULL, nBufferSize1
     , outBufferETH_READER->pBufferpBuffer1)
     ;
 OMX_AllocateBuffer(appPriv->sobelhandle, &
     SOBELoutBuffer1, 1, NULL,
     buffer_out_size);

 /*Setting up tunneled communication*/
 OMX_SetupTunnel(appPriv->rgb2bwhandle, 1,
     appPriv->sobel,0);

 /* Change HW OMX Component state */
 OMX_SendCommand(appPriv->rgb2bwhandle,
     OMX_CommandStateSet, OMX_StateIdle,
     NULL);
 OMX_SendCommand(appPriv->sobelhandle,
     OMX_CommandStateSet, OMX_StateIdle,
     NULL);
 OMX_SendCommand(appPriv->rgb2bwhandle,
     OMX_CommandStateSet, OMX_StateExecuting
     , NULL);
 OMX_SendCommand(appPriv->sobelhandle,
     OMX_CommandStateSet, OMX_StateExecuting
     , NULL);
 ...
 OMX_DeInit();
 return 0;
}
```

Listing 1: Ejemplo de una implementación de una cadena de procesamiento multimedia Hw/Sw con Componentes OpenMAX Hw.

Tal y como puede observarse en el código, el programador trata los componentes OpenMAX Hw exactamente igual que a cualquier otro componente. Esto se debe a que en realidad el desarrollador interactúa con sus fachadas que tienen una interfaz 100 % OpenMAX. En segundo plano, la fachada re-direccionan los parámetros a su componente Hw asociado a través de una invocación a un método en la interfaz del componente Hw. Al utilizar el middleware, cada fachada realiza dicha invocación utilizando el proxy Sw que será el encargado de hacer uso de la infraestructura ofrecida por OOCE para comunicar objetos Sw con objetos Hw. Como se aprecia en el código, el envío de parámetros a objetos Sw y Hw es completamente transparente al programador de la aplicación.

Para simplificar la tarea del desarrollador y aumentar el nivel de abstracción de la plataforma y del proceso de integración, se ha automatizado en gran medida la generación del código fuente de los elementos que intervienen en dicho proceso. Esto supone un ahorro considerable en coste de desarrollo y mantenimiento, a la par que se reduce el esfuerzo de integrar un IP en una cadena de procesamiento basada OpenMAX.

4. Conclusiones y trabajo futuro

En este trabajo se ha presentado una plataforma de integración Hw/Sw basada en una arquitectura de bus que ofrece una interfaz OpenMAX de acceso común a los componentes presentes en ella y por tanto permite a IPs comerciales formar parte de un flujo de aplicación estandarizado. Además, se ha conseguido una comunicación entre componentes OpenMAX Hw eficiente y descentralizada, evitando la intervención de rutinas u otros elementos en las transferencias de datos entre ellos.

Debido a la diversidad de requisitos en los IPs de procesado multimedia, una línea futura de investigación es analizar el impacto de construir una jerarquía de buses para poder agrupar los diferentes IP en base a sus necesidades computacionales. Por otra parte, se está dotando a la plataforma de la infraestructura necesaria para dar soporte a la reconfiguración dinámica de los componentes OpenMAX Hw, permitiendo cambiar su funcionalidad en tiempo de ejecución.

5. Agradecimientos

Esta investigación ha sido financiada por el Ministerio Español de Ciencia e Innovación bajo el proyecto DREAMS (TEC2011-28666-C04-03), y por el Ministerio Español de Industria y el Centro para el Desarrollo Tecnológico Industrial bajo el proyecto ENERGOS (CEN-20091048).

5.1. Referencias

[1] Martijn J. Rutten, Jos T.J. van Eijndhoven, Evert-Jan D.Pol, Egbert G.T. Jaspers, Pieter Van der Wolf, Om Prakash Gangwal and Adwin Timmer. “Eclipse: A Heterogeneous Multiprocessor Architercture For Flexible Processing”, Philips Research Laboratories, 2002.

[2] Paul Brelet, Arnaud Grasset, Philippe Bonnot, Frank Ieromnimon and Dimitrios Kritharidis. “System Level Design for Embedded Reconfigurable Systems using MORPHEUS platform”, IEEE Annual Symposium on VLSI, 2010.

[3] Abbes, F.; Abid, M.; Casseau, E., “Interface Architecture Generation for IP Integration in SoC Design”, Computer Engineering and Systems, The 2006 International Conference on , vol., no., pp.66,70, 5-7 Nov. 2006.

[4] Jer-Min Jou; Shiann-Rong Kuang; Kuang-Ming Wu, “A hierarchical interface design methodology and models for SoC IP integration”, Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on , vol.2, no., pp.II-360,II-363 vol.2, 2002

[5] Openmax webpage. http://www.khronos.org/openmax, 2014.

[6] J. Barba and F. Rincon and F. Moya, J.D. Dondo and F.J. Villanueva and D. Villa and J.C. Lopez, “Object-Oriented Communication Engine for SoC Design”, DSD - Euromicro Conference on Digital System Design. Lubeck (Germany), 2007.

[7] Barba, J. and de la Fuente, D. and Rincon, F. and Moya, F. and Lopez, J.C., “Openmax hardware native support for efficient multimedia embedded systems”, Consumer Electronics, IEEE Transactions on, 2010.

[8] Rincon, Fernando and Moya, Francisco and Barba, Jesús and Lopez, Juan Carlos, “Model Reuse Through Hardware Design Patterns”, Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1, DATE 2005.