El procesador CELL, desde un enfoque histórico

Ramiro Barcala Roca, Valentin Angrigiani, Gabriel Hackl

Resumen—Este trabajo trata sobre el procesador CELL de Sony. Se toma un enfoque investigativo y contrastante entre arquitecturas del CELL, y otras de la época (y la actualidad). Las aplicaciones principales para las que fue diseñado, y las que se descubrieron luego, junto con su importancia histórica. También veremos conceptos básicos sobre la computación heterogénea (distintos nucleos) y un análisis a futuro relacionándolo con lo anterior.

Index Terms—SPE, PPE, SIMD, Pipeline/ing, Paralelismo, Calculos vectoriales, Computacion heterogenea, Celdas.

1. Introducción

A Mediados de los 2000, mientras los CPU's multinucleo recien se estaban introduciendo al mercado, la empresa Sony empieza a investigar y desarrollar su sistema de entretenimiento *PlayStation 3*. Todo esto en un mercado competitivo frente a otras marcas como Microsoft y Nintendo. Terminan diseñando un procesador multi-core, el cual tiene como particularidades los sub-núcleos. Sony tambien quería estandarizar su procesador en dispositivos de todas las gamas y usos multimedias.

Sin embargo, tanto el producto como su procesador fueron un fracaso comercial, pero se descubrieron usos investigativos/científicos/economicos, entre otros.

2. Época Pre-CELL

Para ponernos en contexto, vamos a explicar la situación de las arquitecturas del momento, como de las empresas que lanzaban nuevos productos (del mismo rubro o parecido) en aquella época.

2.1. Empresas del rubro de las consolas

En el año 2005, las siguientes empresas de consolas/videojuegos, competían en el mercado:

- **Sony**: Venía de un éxito rotundo con su anterior producto: la *PlayStation 2*. Tenía grandes expectativas con su nueva consola, apostando a lo grande, con nuevas funcionalidades no esenciales (Ejemplo: puertos HDMI, Blue-Ray, etc).
- **Microsoft**: Venía de un éxito moderado con su primer producto: la *Xbox*. En su nuevo producto, la Xbox 360, se centra en abaratar los costos y proveer servicios en línea de gran calidad. Esto les pasa factura con los problemas de hardware que tendria la consola mas adelante.
- **Nintendo**: No tuvo mucho éxito en su consola anterior: la *Gamecube*. En la *Wii*, se decide no hacer costos de fabricación muy caros. Pero además, se centra en facilitar experiencias novedosas para el usuario final, mediante el uso de movimientos corporales para controlar los juegos, como extensiones al mismo control (volantes, soportes, etc).

Es necesario comprender que la industria de los juegos estaba en su apogeo. No solo se estaban demandando consolas cada vez más rápidas y potentes, sino también mucha mas gente se introducía a este tipo de consumo.

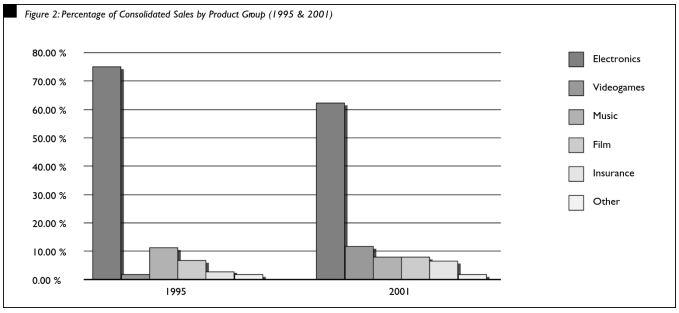

Figura 1. Análisis del mercado del entretenimiento, entrando al nuevo milenio.

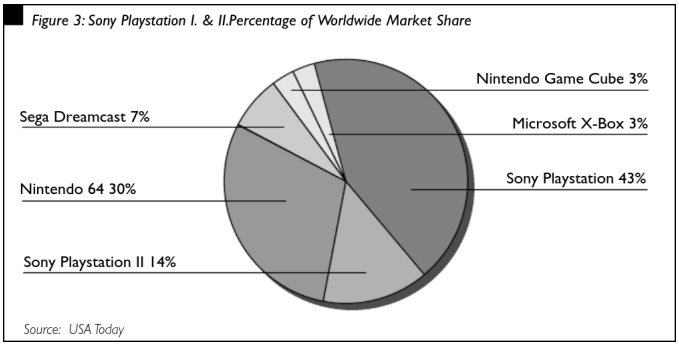

Figura 2. Porción del mercado de las consolas, de cada empresa.

3. Arquitectura CELL

Diseñada por Sony, en conjunto con Toshiba e IBM, esta arquitectura trae al mercado un procesador que maneja fuertemente la computación heterogénea, mediante el uso de sus 9 núcleos.

Si bien originalmente se pensaba incluirlo en dispositivos multimedia y del entretenimiento varios, se terminó destinando casi unicamente en la PlayStation 3. De todas formas, Sony realmente quería traer una nueva revolución al rubro del diseño de arquitecturas.

3.1. Distintos módulos del Cell

Para empezar, hablaremos del PPE (*Power Processing Element*). Este nucleo, es el principal de la arquitectura. Es de proposito general, y a su vez se encarga de controlar el resto de los sub-núcleos (cargar memoria y ejecutar instrucciones). Estaba conformado por una memoria caché L1 (de 64 KB), una memoria caché L2 (de 512 KB), y por el PPU (*Power Processing Unit*).

Luego, los SPE (*Synergistic Processing Element*), son los distintos sub-núcleos de la arquitectura. No son de proposito general; más bien se encargan de realizar operaciones vectoriales dirigidas desde el PPE. Cada uno tiene su memoria interna (*Local Store*), un módulo *Memory Flow Controller* para poder recibir/transmitir datos desde el Bús, y el SPU (*Synergistic Processing Unit*). Cada uno de estos SPE trabajan con instrucciones SIMD con punto flotante, pudiendo llegar a realizar 25,6 Gigaflops por segundo (25,6 mil millones de operaciones de punto flotante por segundo). Vale la pena mencionar que las instrucciones SIMD se encargan de aplicar la misma operación a un conjunto de datos, para así poder obtener un mayor rendimiento en tiempo.

Adicionalmente cuenta con el *Memory Interface Controller* (MIC, un controlador de memoria externa), y con el *Bus Interface Controller* (BIC, con 2 vías de entrada y salida). En la siguiente figura podemos observar cómo están organizados los distintos módulos:

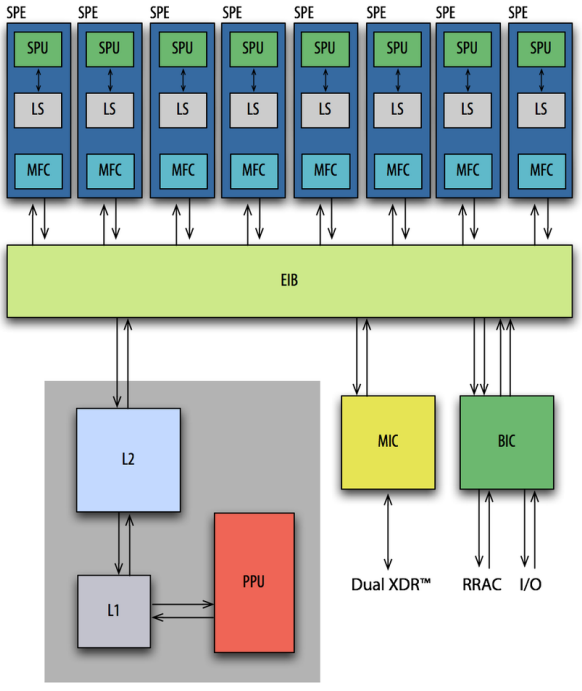

Figura 3. Arquitectura del CELL Broadband Engine.

3.2. Interconexión

La interconexion de los distintos módulos y núcleos (además de la memoria externa, y la interfaz I/O) se realizaba mediante el *Element Interconnection Bus* (EIB), que estaba conformado por 4 anillos unidireccionales tokenizados, 2 en cada dirección. Por ende, la información no se retransmitía más de lo que debía mediante un identificador del modulo al que debía llegar la informacion. Por cómo estaba armado, nunca sería posible saturar dicho bus (ancho de banda interno máximo de 204,8 GB/s).

3.3. Computacion heterogenea en el Cell

La computacion heterogénea puede ser entendida como aquella computación que requiere el manejo y codificado de instrucciones para una arquitectura que usa distintos tipos de núcleos.

A gran escala, podríamos decir que la gran mayoría de consolas o computadoras orientadas al *Gaming*, hoy en dia, requieren de la misma. Se debe instruccionar al CPU, GPU, etcétera. Sucede lo mismo con modelados de financiación o de computación científica, los cuales se benefician de ambos aspectos. Sin embargo, se puede dar a una escala muchísimo más chica.

Resulta que la computación heterogénea, dentro del Cell, se aplica a los distintos SPE's que residen en el mismo. De forma, que no solo programamos el PPE, sino los SPE, los cuales reciben sus instrucciones (y memoria cargada) del primero mencionado. Además, el propio desarrollador puede realizar cierto tipo de *pipelining*: Paralelizar los programas de los SPE, vectorizándolos y computarizando las salidas de los mismos.

El código destinado a los SPE's debe ser escrito aparte del programa del PPE, y traducido al lenguaje de máquina por compiladores específicos de SPE's.

3.4. Contraste con arquitecturas actuales

La idea del Cell era aplicar el tipo de arquitectura de çeldas", donde cada módulo se encarga de una tarea lo más específica posible. Así, aplicamos el lema "*Divide y vencerás*", pudiendo manejar operaciones más voluminosas y complejas. También, como cada módulo funcionaba por separado, se lo podía escalar tanto como se quisiese. Esto facilitó el creado de sistemas que aplicaban este paradigma (como las supercomputadoras).

Mientras que en el Cell se requería de una computación heterogénea como la mencionada anteriormente, en las arquitecturas actuales esto no es tan así (no así en los sistemas que los abarcan. Ejemplo: Computadoras dedicadas al *Gaming*, ya que manejamos núcleos de distintos tipos). No están diseñadas con núcleos diferentes, a diferencia del Cell. Entonces, el enfoque al paralelizar tareas variaba; el Cell requeria de un manejo especial por parte del desarrollador (con los SPE's), mientras que hoy en día hay muchas herramientas que facilitan la división de tareas entre los núcleos.

3.5. Arquitecturas de la competencia

Ya habiendo explicado la arquitectura del Cell, veremos algunas de las diferencias con las de la competencia:

- **Microsoft**: En la Xbox 360, se diseña una arquitectura gracias a IBM que tiene ciertas similaridades con el Cell. Por empezar, el procesador de la consola era el *Xenon*, de 3 núcleos, que eran PPE's como los del Cell. Sin embargo, no se encontraba en la arquitectura ningún SPE. Los PPE's, tanto como el resto de modulos, se interconectaban mediante un bús *XBAR*. A grandes rasgos, es la otra compañía del momento la cual también busca un acercamiento al paralelismo.

- **Nintendo**: En la Wii, también se recurre a IBM. Resulta que el procesador que termina ocupando esta consola (Broadway), es un *rebranding* de la arquitectura ya usada por el producto anterior (la Gamecube). Esto es así ya que IBM modifica la arquitectura que le pide Nintendo, haciéndola correr a un clock mayor.

  Por otro lado, la Wii tambien poseía un coprocesador llamado Starlet, el cual se encargaba de tareas menores como el manejo de periféricos o retrocompatibilidad con juegos de consolas anteriores, o hasta del firmware de la consola. Sin embargo, el manejo de estos dos CPU's se alejaba bastante de la computación heterogénea que requería el Cell.

4. Inconveniencias

Debido a que programar los distintos SPE's requerían mucho tiempo, una gran cantidad de desarrolladores pasaron por alto los programas apartes que requerian los mismos, y se centraban únicamente en el PPE, que terminaba realizando todas las tareas. Esto provocaba bajos rendimientos en los programas que corría el CELL, ya que sus SPE's quedaban inutilizados.

Muchas tareas que se podían paralelizar, no se estaban ejecutando como debían, ni por los sub-núcleos especializados para eso. Además, los SPE's tampoco tenían *branch prediction*, por lo que sentencias como if/else eran supervisadas por el PPE, asignándole más tareas.

Esto provocaba una performance mediocre en la mayoría de programas realizados por desarrolladores con poco conocimiento y experiencia en este paradigma. El Cell tenía potencial, pero requería un código extremadamente conciso, con pocos condicionales y que mantuviese ocupado a los 9 núcleos en todo momento.

Por si no era suficiente, el costo de fabricación de la consola, durante parte de los primeros 3 años del lanzamiento, rondaba alrededor de los US$900, mientras que el precio para el consumidor final era de US$500; sumado a que Microsoft ya había lanzado su consola hace un año atrás, ganando gran parte del mercado. Finalmente, los consumidores a los que se destinó el producto, lo rechazaban en gran medida.

5. Aplicaciones del CELL

Sin importar el resultado de la aplicación de la arquitectura, al Cell se lo utilizó para, por ejemplo:

- **Clima**: Ya que para replicar efectos meteorológicos como las nubes, y para procesar el inmenso volumen de información que llega desde los instrumentos usados para las distintas mediciones, requería de un sistema que pudiese realizar estas tareas intensivas. Se decidió experimentar con esta arquitectura para diseñar más adelante una solución apropiada (que solucionase también los problemas que había con el ancho de banda limitado de memoria).

- **Modelos de simulación**: Un gran ejemplo de esto es el *RoadRunner*, una supercomputadora que llegó a operar a 1 petaflop (mil billones de operaciones de punto flotante por segundo), capaz de simular explosiones nucleares y demás cuestiones armamentísticas del gobierno de los Estados Unidos.

- **Medicina**: Aquí también fue útil el Cell, ya que facilita procesar la salida de las tomografías computarizadas de un uso intensivo de recursos, para reconstruir imágenes médicas en 3D.

Otras aplicaciones de la arquitectura fueron en la computarización masiva de datos, en servidores, y hasta incluso en la industria petrolera y gasífera.

6. Remanentes del CELL en la Actualidad

Viendo en retrospectiva, hoy se sigue aplicando hasta cierto punto parte de lo que se promovía en la arquitectura, pero a gran escala.

Con el diseño de celdas, ejemplos serían las telecomunicaciones, los datos móviles; siendo las distintas células las porciones de terreno con cobertura móvil, mediante las torres. O también como las plataformas funcionando en la nube, o en los sistemas como computadoras personales, consolas, etcétera. Tanto en la memoria diviéndose en celdas, o las tareas que realiza un microprocesador, encargando a los distintos núcleos distintas tareas. Por lo visto, no solo se expandió a arquitecturas, sino a sistemas que abarcan muchas más cosas.

Este tipo de diseño arquitectural nos permite escalar el sistema con facilidad, pudiendo aumentar la carga de trabajo sin tener que reestructurarlo por completo. También, al suceder fallas, estas se aíslan sobre el módulo erróneo y encapsula el error, para poder modularizar, y reemplazarlo con sencillez. Además facilita el paralelismo, enfocando las celdas a propósitos simples/únicos y eficientando la utilización de recursos.

Por otro lado, la computación heterogénea no se vio tan masificada en el diseño de arquitecturas, pero sí en el uso de sistemas que los abarquen. Hoy en día, interesa más el hecho de poder realizar tareas eficientemente, con arquitecturas especializadas, para después poder complementarlas en un

mismo equipo, no así como se hacía en el Cell, donde se encontraban núcleos con propósitos distintos.

7. Análisis a futuro

Habiendo visto las distintas aplicaciones que hay hoy en día con las arquitecturas basadas en celdas, creemos que seguirá al alza. No solo se seguirá aplicando a los diseños de microprocesadores y sistemas, sino que se expandirá a muchos de los rubros en los que hoy no se lo toma en cuenta.

Un ejemplo sería en la agricultura, dividiendo en parcelas los cultivos o ganados, y controlando las cosechas o animales, para revisar en cuál se debe focalizar una atención especial. O tambien en los hospitales, buscando aquellos pacientes que requieran un cuidado particular.

Ahora, creemos que la computación heterogénea sí tiene lugar en el futuro, pero algo similar a como pasa en la actualidad. No que se centre en el interior de los distintos diseños minimales, sino de la manera más abstracta posible.

8. Conclusión

Creemos que, si bien el enfoque que tomó Sony con su arquitectura (originalmente compartida por dispositivos varios del mundo del entretenimiento) no tuvo un éxito considerable. Fue útil para aproximarse al mundo de la computación heterogónea, y para darle otro foco a la capacidad de conseguir un mayor rendimiento en los procesadores venideros, no solo aumentando la frecuencia del *clock* de los mismos.

Además, tuvo bastante importancia su experimento con las arquitecturas basadas en celdas, ya que le permitió a las distintas industrias que hoy en día lo aplican, ver sus beneficios.

Referencias

[1] *Laboratorio de Arquitecturas Avanzadas con Cell y PlayStation 3*, Universitat de València, por Fernando Pardo y Jose A. Boluda.

[2] *The PlayStation 3 for High Performance Scientific Computing*, University of Tennessee: Jakub Kurzak, Alfredo Buttari, Piotr Luszczek, Jack Dongarra

[3] *The impact of IBM Cell technology on the programming paradigm in the context of computer systems for climate and weather models*, Shujia Zhou, Daniel Duffy, Thomas Clune, Max Suarez, Samuel Williams, Milton Halem.

[4] *Architecture of Consoles: A practical analysis by Rodrigo Copetti*, https://www.copetti.org/writings/consoles/

[5] *Cell Technology Tackles 3D Medical Imaging Reconstruction Challenges*, https://www.techbriefs.com/component/content/article/6088-10976-400

[6] *Breaking the petaflop barrier*, https://www.ibm.com/history/petaflop-barrier

[7] *Introducing the IBM/Sony/Toshiba Cell Processor ? Part I: the SIMD processing units*, https://arstechnica.com/features/2005/02/cell-1/

[8] *The PlayStation Supercomputer*, https://www.datacenterdynamics.com/en/analysis/the-playstation-supercomputer/

[9] *Console wars: A rare bright spot in the gloomy technology industry, video games are growing up*, https://www.economist.com/business/2002/06/20/console-wars

[10] *Console wars*, https://www.gainesville.com/story/news/2006/11/11/waging-console-war/31502430007/

[11] *PlayStation 3, Console Wars and the Costs of Complexity*, https://techliberation.com/2006/09/07/playstation-3-console-wars-the-costs-of-complexity/

[12] *PlayStation 3, Console Wars and the Costs of Complexity*, https://arstechnica.com/features/2005/02/cell-1/

[13] *Cell-Based Architecture: The Backbone of Scalable Systems*, https://medium.com/@jishanrandika/cell-based-architecture-the-backbone-of-scalable-systems-2b5ea50cbe1c