ISBN 978-987-46297-6-0


# Evaluación de Desempeño de un Patrón de Software para Multiprocesadores Asimétricos en Sistemas Embebidos

Pedro Ignacio Martos
GPSIC & LSE – Facultad de Ingeniería
Universidad de Buenos Aires
Ciudad Autónoma de Buenos Aires, Argentina
pmartos@fi.uba.ar / pimartos@gmail.com

Alejandra Garrido
LIFIA – Facultad de Informática
Universidad Nacional de La Plata & CONICET
La Plata, Prov.de Buenos Aires, Argentina
garrido@lifia.info.unlp.edu.ar

***Resumen*—Dentro de los sistemas embebidos hay una variante de sistemas multiprocesador (Multicore System on Chip devices – MSoC devices) donde no todos los procesadores son iguales. Estos SoCs se denominan "Dispositivos Multiprocesadores Asimétricos" (Asymmetric Multi Processing Devices – AMP Devices). Para tomar ventaja de las posibilidades que estos dispositivos ofrecen en sistemas embebidos, se han definido patrones de diseño específicos. No obstante ello, en la literatura hay pocos resultados experimentales que muestren los beneficios de aplicar estos patrones. En este trabajo se realiza la medición de desempeño del patrón "Ejecución Optimizada" (Optimized Execution), aplicado a un dispositivo AMP y usando la suite de desempeño "CoreMark". Los resultados muestran que el uso de este patrón de diseño mejora el desempeño del sistema.**

***Palabras Clave— Patrones de Sistemas Embebidos; Patrones para Multiprocesamiento Asimétrico; Software Embebido.***

## I. INTRODUCCIÓN

Los sistemas embebidos (Embedded Systems – E.S), en oposición a los sistemas de propósito general, son sistemas desarrollados con un propósito específico. En algunos casos el usuario final del sistema puede configurarlo (e incluso programarlo), pero el sistema no cambia su propósito. Estos sistemas se denominan "embebidos" porque son parte de un sistema más grande en un dispositivo con una funcionalidad específica [1].

Un Multiprocesador Asimétrico (AMP), es un procesador en el que los elementos de cómputo ("núcleos") tienen diferentes características. Estas diferencias van desde el mismo tipo de procesador ejecutándose a diferentes velocidades de reloj, hasta arquitecturas completamente distintas. Estos procesadores son estudiados porque se encontró que proveen una mejora en la potencia de computo [2], y también en las relaciones (desempeño)/(consumo) y (desempeño)/(área de silicio) [3], por lo que son atractivos para la implementación de E.S. No obstante ello, el diseño de un E.S. que pueda tomar ventaja de la mejora en desempeño de los AMP es difícil. Por ejemplo, la asimetría en desempeño puede afectar adversamente el comportamiento de distintas cargas de cómputo en servidores comerciales y hacer que el sistema sea menos escalable [2].

Con el objetivo de ayudar a los desarrolladores a tomar buenas decisiones de arquitectura de software al implementar sistemas AMP y para tomar ventaja de sus características, hemos trabajado en la especificación de un lenguaje de patrones para sistemas embebidos AMP [4]. Un lenguaje de patrones es una colección interconectada de patrones de diseño o buenas prácticas en un dominio específico [5]. Cada patrón de diseño del lenguaje identifica una solución recurrente a un problema en un contexto específico [6]. De esta manera, un patrón contiene una pequeña porción de conocimiento de diseño y arquitectura para resolver un problema en un cierto contexto y una vez resuelto, deja al sistema en un nuevo contexto, donde hay nuevos problemas a resolver por otros patrones del lenguaje [7].

En otros trabajos hemos realizado mediciones de desempeño de algunos patrones del lenguaje propuesto para sistemas AMP [4]. En particular hemos demostrado que una configuración AMP tiene un desempeño mejor a lo esperado al aplicar el patrón "Mini-Me" también cuando la asimetría de la configuración AMP se debe a procesadores con diferente hardware y con cargas específicas de E.S.[8]

En este trabajo se realiza la medición de desempeño del patrón "Optimized Execution" para evaluar la mejora que produce al aplicarlo. Este patrón, el cual se describe en detalle en la Sección 2, propone que, cuando se tiene un sistema AMP cuyos núcleos tienen distinto ISA (Instruction Set Architecture – ISA), y un ISA es un subconjunto del otro, se debe tomar ventaja de las instrucciones específicas del ISA mas grande, a fin de mejorar el tiempo de ejecución de las tareas. En particular se intenta responder a la siguiente pregunta: "¿Que tan importante es la diferencia de desempeño cuando se usa el ISA específico de cada núcleo en una configuración AMP donde el ISA de un tipo de núcleo es un subconjunto del ISA del otro tipo de núcleo?"

Esta pregunta es importante porque el uso del mismo ISA en núcleos como los descriptos anteriormente tiene una ventaja importante: el sistema puede implementarse con todo el software usando el mismo ISA y la asimetría solo estaría en el hardware, con los resultados descriptos en [8]. Pero, si la diferencia en desempeño es significativa, se necesita un nuevo patrón de software para contemplar esta situación. Para contestar esta pregunta se utilizó una plataforma de cómputo AMP con dos núcleos: un núcleo Cortex M4 y un núcleo Cortex M0, en el que el ISA del núcleo Cortex M0 es un

ISBN 978-987-46297-6-0


subconjunto del ISA del núcleo Cortex M4. Asimismo se uso una suite de desempeño bien conocida para este estudio: CoreMark [9].

El resto de este trabajo se estructura de la siguiente forma: la Sección 2 presenta los trabajos relacionados, la Sección 3 describe la suite de desempeño y detalles de la implementación, la Sección 4 muestra los resultados experimentales, y la sección 5 presenta las conclusiones y trabajos a futuro.

## II. Trabajos Relacionados

Los procesadores AMP se estudian como dispositivos de cómputo por las ventajas que ofrecen. En particular su desempeño es mejor a lo esperado. Como se explicó en la introducción, una limitación del trabajo previo [8] para nuestro propósito es que en el sistema utilizado todos los núcleos tienen el mismo ISA. Nosotros estamos interesados en evaluar el desempeño cuando el ISA de los procesadores es diferente.

Otros trabajos sobre lenguajes de patrones relacionados con el dominio de los E.S. son los patrones para software tolerante a fallas de Hanmer [7] y los patrones para sistemas embebidos de White [10]. Uno de los patrones en el lenguaje de patrones para AMP en E.S es "Optimized Execution" [4], el cual se usa en el contexto en el que un sistema embebido utiliza un procesador AMP donde los núcleos tiene diferentes ISA, y el ISA de un tipo de núcleo, denominado "A", es un subconjunto del ISA del otro tipo de núcleo, denominado "B". Este patrón se aplica al problema de cuando se tienen tareas con requerimientos de tiempo real estricto, cuando hay diferentes ISA disponibles, y un ISA es un subconjunto del otro, es necesario tomar ventaja de las instrucciones específicas del ISA mas grande, para mejorar el tiempo de ejecución de las tareas. Por otra parte este Patrón puede considerarse como opuesto al Antipatrón "Level Down" [4], el cual plantea que el uso del mismo ISA en procesadores asimétricos produce una merma en el desempeño total del sistema. Nuestra intención en este trabajo es evaluar el desempeño de la arquitectura propuesta por el patrón Optimized Execution en el contexto de los E.S.

## III. Medición de Desempeño

La plataforma de hardware utilizada fue la Edu-CIAA, la cual utiliza el procesador de dos núcleos LPC 4337 de NXP, el cual tiene un núcleo complejo (ARM Cortex M4@204MHz) y un núcleo simple (ARM Cortex M0@204MHz) [11]. El ISA del núcleo Cortex M0 (Armv6-M) es un subconjunto del ISA del núcleo Cortex M4 (Armv7-M), lo que hace a esta plataforma apropiada para medir una implementación del patrón Optimized Execution. La suite de desempeño CoreMark fue obtenida del sitio web de EEMBC; compilada con el nivel de optimización –O3; y configurada para ejecutarse desde memoria RAM en configuración "Release".

CoreMark es una suite que mide el desempeño de microcontroladores (MCUs) y unidades centrales de procesamiento (CPUs) utilizadas en sistemas embebidos. CoreMark implementa los siguientes algoritmos: manejo de listas (búsqueda y ordenamiento), manipulación de matrices (operaciones comunes con matrices), maquinas de estado (determinar si una secuencia de entrada contiene números válidos), y chequeo de redundancia cíclica (CRC – cyclic redundancy check). Está diseñada para ejecutarse en dispositivos que van desde microcontroladores de 8-bit hasta microprocesadores de 64-bit. Para asegurar que el compilador no pueda pre-computar los resultados en tiempo de compilación, cada operación de la suite de desempeño genera un valor que solo puede determinarse en tiempo de ejecución. Asimismo todo el código usado en la parte temporizada es parte de la suite en si misma (no se realizan llamadas a librerías).

## IV. Resultados Experimentales

La Tabla 1 muestra los resultados CoreMark absolutos de la plataforma de hardware. Asimismo se muestran los resultados CoreMark para el procesador Cortex M4 LPC 54102[12] para validar nuestra medición respecto a la medición CoreMark realizada por NXP. La Tabla 2 muestra los resultados CoreMark/MHz, los cuales se utilizan para comparar mediciones CoreMark realizadas con procesadores ejecutándose a diferentes velocidades.

Tabla 1: Resultados CoreMark Absolutos

| Configuración | CoreMark |
|---|---|
| LPC 4337 Armv7-M | 488 |
| LPC 4337 Armv6-M | 390 |
| LPC 54102 Armv7-M | 499 |

Tabla 2: Resultados CoreMark/MHz

| Configuración | CoreMark/MHz |
|---|---|
| LPC 4337 Armv7-M | 2.39 |
| LPC 4337 Armv6-M | 1.91 |
| LPC 54102 Armv7-M | 2.45 |

De la Tabla 1 se ve que hay un 25% de diferencia en desempeño usando el ISA Armv6-M, respecto de usar el ISA Armv7-M. Este resultado muestra que el ISA utilizado tiene gran impacto en el desempeño del sistema.

De la Tabla 2 se observa que la medición de NXP es similar a nuestra medición. La diferencia puede explicarse debido a que el procesador LPC 54102 es más nuevo y seguramente mejorado respecto al procesador LPC 4337 usado en nuestras mediciones, por lo que consideramos que nuestras mediciones son válidas y nuestra implementación de la suite CoreMark es correcta.

## V. Conclusiones y Trabajo a Futuro

En este trabajo se realizo la medición de desempeño del patrón de software Optimized Execution, utilizando una suite de desempeño orientada a tareas especificas de E.S. Nuestra pregunta probó ser muy importante y a través de mediciones experimentales se encontró que se puede esperar una diferencia de desempeño de alrededor de 1.25 veces.

Los trabajos a futuro incluyen realizar mediciones de desempeño de otros patrones del lenguaje, extenderlo y descubrir nuevos patrones específicos del dominio.

ISBN 978-987-46297-6-0


## REFERENCIAS

[1] Steve Heath, Embedded Systems Design, 2nd Ed. Elsevier, (2002)

[2] Balakrishnan, Rajwar, Upton, Lai, The impact of performance asymmetry in emerging multicore architectures, Proceedings of the 32nd International Symposium on Computer Architecture (ISCA '05). IEEE. (2005)

[3] Fedorova, Saez,Shelepov, Prieto, Maximizing power efficiency with asymmetric multicore systems, Communications of the ACM, Vol 52 Issue 12 (2009)

[4] P. Martos, Architectural Patterns for Asymmetric Multiprocessing Devices on Embedded Systems, Proceedings of the 11th Latin American Conference on Pattern Languages of Programs (SugarLoaf PLoP '16). Hillside Group (2016)

[5] R. Hanmer, Pattern-Oriented Software Architecture For Dummies, John Wiley & Sons, 2013

[6] Gamma, Help, Johnson, Vlissides. Design Patterns. Addison-Wesley, 1995.

[7] Robert S. Hanmer, Patterns for fault tolerant software, Wiley Series in Software Design Patterns. John Wiley & Sons (2007)

[8] P.Martos & A.Garrido, Software Patterns for Asymmetric Multiprocessing devices on Embedded Systems: a Performance Assessment", Eight Argentine Symposium and Conference on Embedded Systems (CASE). IEEE (2017)

[9] Shay Gal-On and Markus Levy, Exploring CoreMark™ – A Benchmark Maximizing Simplicity and Efficacy. EEMBC (2009)

[10] E. White, "Making Embedded Systems: Design Patterns for Great Software", O'Reilly Media (2011)

[11] "LPC 4337",
https://www.nxp.com/products/processors-and-microcontrollers/arm-based-processors-and-mcus/lpc-cortex-m-mcus/lpc4300-cortex-m4-m0/32-bit-arm-cortex-m4-m0-mcu-up-to-1-mb-flash-and-136-kb-sram-ethernet-two-high-speed-usb-lcd-emc:LPC4337JBD144,
(obtenido en Mayo,2019)

[12] NXP Semiconductors, "AN 11607 LPC 5410x CoreMark Cortex M4 Porting Guide" (2015)