Cuotas hardware para el despliegue de aplicaciones de alta criticidad en sistemas multiprocesador

Pablo Andreu[1],Carles Hernández[1] y Pedro Lopez[1]

Resumen— **Los procesadores multinúcleo están presentes en todas las plataformas de computo de alto rendimiento de hoy en día debido a su mayor eficiencia. Dicho incremento en eficiencia respecto a los procesadores mononúcleo se consigue gracias a la compartición de recursos. Sin embargo, dicha compartición de recursos genera interferencia entre núcleos y complica la validación temporal del software que se ejecuta en el sistema. En este articulo presentamos un sistema de cuotas hardware para controlar dicha interferencia entre núcleos y preservar las garantías funcionales del sistema.**

Nuestro mecanismo de cuotas actúa sobre la contención entre núcleos. Con este mecanismo conseguimos controlar la interferencia entre núcleos, garantizar tiempos de ejecución para tareas prioritarias y maximizar el rendimiento garantizado del sistema.

Palabras clave— **Criticidad mixta, procesadores multinúcleo, calidad de servicio.**

I. Introducción

EL uso de procesadores multinúcleo es prácticamente un requerimiento en la mayoría de sistemas modernos. Sin embargo, la complejidad de los sistemas multinúcleo afecta gravemente a la habilidad de certificar dichos sistemas con los niveles de seguridad funcional necesarios para determinadas aplicaciones. Por ejemplo, los coches autónomos requieren una certificación de los elementos de computo de acuerdo con los niveles de criticidad mas altos, dado que un fallo en dicho sistema puede tener consecuencias catastróficas.

Los procesadores multinúcleo con muchos recursos compartidos aumentan la complejidad de la verificación de la temporización del sistema. La verificación temporal es un requisito definido en los estándares de certificación de los sistemas críticos. Cuanto mayor es la criticidad, mayor es la evidencia necesaria para garantizar el cumplimiento de los limites temporales asignados a cada tarea del sistema. La razón por la que los procesadores multinúcleo aumentan la complejidad del proceso de verificación reside en las dificultades para cuantificar de forma precisa y segura las interferencias que pueden producirse debido a las interacciones con otras tareas que se ejecutan conjuntamente en los diferentes recursos compartidos. Esto lleva a una sobre-estimación de la interferencia que sufren estas tareas y por tanto una reducción efectiva de las capacidades de computo del sistema.

Varios trabajos recientes han propuesto la asignación de cuotas de utilización de recursos compartidos a los diferentes núcleos para controlar las interferencias temporales entre estos. Los núcleos que agotan sus cuotas son detenidos por dichos mecanismos para garantizar que las tareas críticas del núcleo pueden cumplir sus requisitos de sincronización. Los sistemas de cuotas basados en software necesitan una tarea de monitorización que maneje la situación, normalmente a través de interrupciones. Lamentablemente, en sistemas con un número moderado de núcleos, esto puede llevar a estimaciones de tiempo pesimistas debido al tiempo necesario para procesar todas las posibles violaciones de cuota. De hecho, en las aplicaciones relacionadas con la seguridad, las tareas de monitorización también controlan las propiedades funcionales del sistema, como la posible aparición de errores que deben ser tratados a tiempo.

En este trabajo proponemos un mecanismo de cuota basado en hardware (MCH) que actúa sobre la contención entre núcleos, limitando así la interferencia máxima entre estos sin la necesidad de intervención del software. El mecanismo propuesto utiliza la información de contención proporcionada por la unidad de monitorización de rendimiento del sistema para tomar las decisiones de arbitraje.

La evaluación del rendimiento del mecanismo propuesto muestra que los patrones de acceso patológicos pueden limitar el rendimiento global del sistema. Por lo tanto, también ampliamos el MCH con un mecanismo de relleno de cuotas automático que aumenta el rendimiento del sistema sin comprometer las garantías funcionales de este.

II. Sistemas de criticidad mixta

En los sistemas embebidos la certificación según ciertos criterios de seguridad funcional es necesaria. Estas normas de seguridad funcional definen varios niveles de criticidad asociados a las diferentes funcionalidades de un sistema. Por ejemplo, según la nomenclatura de automoción definida en la norma ISO26262 [1] ASIL-D corresponde al nivel mas alto de criticidad mientras que *Quality Managed* (QM) corresponde a tareas que no son relevantes para la seguridad del sistema.

Si un sistema tiene tareas de ambas criticidades se considera un sistema de criticidad mixta. En dicho sistema, las aplicaciones de máxima criticidad deberán de tener unas garantías funcionales mucho mas estrictas a las QM. Por ejemplo, las tareas ASIL-D deberán de responder a una entrada en un periodo de 1ms. Para cumplir dicho requisito funcional, el di-

[1]Dpto. de Informática de Sistemas y Computadores, Universidad Politécnica de Valencia, e-mail: {pabance,carherlu,plopez}@upv.es

señador del sistema deberá de tener en cuenta las interferencias entre núcleos en los recursos compartidos y garantizar que estas no sobrepasen un determinado limite.

Sin embargo, los beneficios de tener sistemas de criticidad mixta son importantes. Estos sistemas permiten reducir la complejidad del sistema total y los costes asociados, permitiendo agrupar varios diferentes subsistemas en el mismo chip. Reduciendo el tamaño del sistema final, su peso, su área y su consumo energético. No obstante, pese a las ventajas que ofrecen los sistemas de criticidad mixta, para tener un sistema de este tipo hace falta aislar de manera temporal o espacial las tareas de la misma o diferentes criticidades [2]

Para poder controlar la interferencia entre diferentes tareas de un mismo sistema este articulo propone el usar nuestro MCH. Permitiendo dicho sistema controlar de manera determinista la interferencia entre tareas en un bus compartido.

III. Reduciendo la contención en un bus compartido

En este articulo se estudia el caso de un bus compartido entre los diferentes núcleos. En dicho bus testeamos diferentes políticas de arbitraje que ofrecen diferentes características funcionales. Dichas políticas son:

- Round Robin.
- Prioridades estáticas.

Con Round Robin, cada ciclo de arbitraje se le da paso a un núcleo de los que piden acceso al bus compartido. Si varios núcleos piden acceso al bus compartido se decide el orden entre estos de manera que el siguiente núcleo al ultimo al que se le dio paso sea el que es arbitrado. De esta manera, si los núcleos 1 y 3 piden acceso al bus al mismo tiempo, el bus se le concederá al núcleo 1 si el núcleo 0 fue el ultimo que accedió y se le concederá al núcleo 3 si el núcleo 1 o 2 fueron los últimos que accedieron al bus.

De esta manera Round Robin reparte el bus compartido de manera equitativa entre todos los actores del bus. Pero esto presenta un problema cuando tenemos un sistema heterogéneo, donde un núcleo ejecuta tareas de mayor criticidad que el resto. En este sistema, el núcleo critico podrá esperar como máximo tantos ciclos de arbitraje como núcleos hay en el sistema hasta que su petición sea resuelta. De hecho, en trabajos previos se ha observado un efecto de sincronía en un bus compartido con Round Robin [3], donde un bus arbitrado con dicha política se comporta como un bus multiplexado en el tiempo . Estas características de los buses con Round Robin dificultan el análisis de la interferencia que la tarea critica sufre por tareas no criticas.

De ahí nace la segunda política de arbitraje que exploramos en este trabajo, esta política da prioridad máxima al núcleo que tiene la tarea critica. De esta manera conseguimos reducir el peor caso del núcleo prioritario a que las peticiones del bus de dicho núcleo tarden un ciclo de arbitraje en resolverse. Si extendemos esta política de prioridades estática de manera que el núcleo cero tenga mas prioridad que el uno, que el núcleo uno tenga mas prioridad que el dos y el dos mas prioridad que el tres conseguimos reducir la interferencia entre núcleos.

Sin embargo, y como veremos en el apartado de resultados, mediante la política estática se mejora el rendimiento y las garantías de las tareas prioritarias respecto a las no prioritarias, pero, calcular la inflación temporal que dichas tareas sufren debido a la compartición de recursos sigue siendo complicado, ya que, en el peor caso deberemos asumir que cada vez que el núcleo principal pide acceder al bus hay un núcleo secundario accediendo con una petición de longitud máxima.

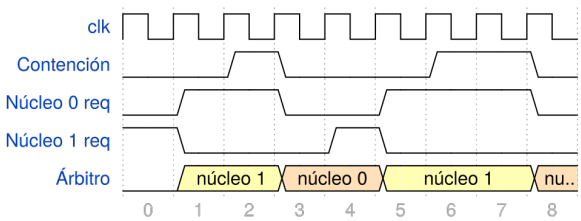


Fig. 1: Contención que el núcleo cero sufre con políticas estáticas

En la Figura 1 se representa un sistema con dos núcleos en un bus compartido con un árbitro que emplea la política de arbitraje de prioridades descrita previamente. En dicha figura mostramos la contención que el núcleo 0 sufre debido a las peticiones del núcleo 1. De esta manera podemos observar como en este ejemplo el núcleo 0 sufre 3 ciclos de contención, siendo la contención máxima que puede sufrir el núcleo 0 en este ejemplo igual a la longitud máxima de petición que se pueda hacer en dicho bus multiplicado por la cantidad de veces que el núcleo 0 accede a dicho bus.

De esta manera podemos determinar que la política de arbitraje del bus decrementa la variabilidad sufrida por el núcleo prioritario pero no consigue reducir el pesimismo en el calculo de la inflación temporal a un valor realista, por lo que si nuestra aplicación necesita poner un limite máximo en la contención sufrida en el núcleo prioritario necesitamos un sistema que actúe junto a la política de arbitraje para conseguirlo.

IV. Propuesta

El mecanismo de arbitraje por cuotas de contención software presenta desventajas claras, como la necesidad de uso de interrupciones, que, a su vez generan contención que debemos tener en cuenta para nuestro calculo.

En este articulo presentamos el MCH, un mecanismo que mitiga el efecto de las interrupciones en la validación temporal del sistema efectivamente eliminándolas. Dicho mecanismo consigue este efecto gracias a la interconexión entre la MCCU y el arbitro del sistema. Gracias a la información proporcionada al arbitro desde la MCCU dicho arbitro puede tomar

decisiones teniendo en cuenta la contención generada por cada núcleo.

De esta manera nuestra solución elimina la problemática causada por las interrupciones y genera un sistema de arbitraje que toma decisiones mas informadas.

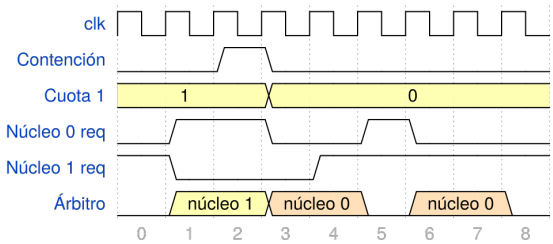


Fig. 2: Arbitraje en un sistema con MCH

El funcionamiento del MCH puede ser observado en la Figura 2 donde se observa que, tras agotarse la cuota de contención disponible el arbitro enmascara las peticiones de acceso al bus del núcleo 1. Gracias al MCH podemos garantizar para este ejemplo que la contención en ciclos que puede sufrir el núcleo 0 es igual al numero configurado en la MCCU. De hecho, si comparamos la Figura 2 y Figura 1 podemos observar como, tras agotarse la cuota de contención, el núcleo 0 accede al bus con su segundo acceso en el ciclo 6 en el sistema con MCH y en el ciclo 8 en un sistema sin MCH.

Otro de los puntos a tener en cuenta a la hora de diseñar el MCH es el agotamiento temprano de la cuota. En un sistema con periodos de arbitraje definidos, cada vez que se cambia de contexto en el núcleo critico se rellena la cuota para los núcleos secundarios. Este proceso de relleno se produce para todos los núcleos secundarios a la vez, por lo tanto, después del periodo de arbitraje se producen periodos de alta saturación del bus compartido junto a periodos de muy baja ocupación de dicho bus compartido debido a los núcleos secundarios habiendo agotado su cuota.

Para mitigar dicho efecto, implementamos el mecanismo de rellenado de cuotas hardware o rellenado hardware. Gracias a este mecanismo podemos repartir una cierta cantidad de cuota en distintos periodos de tiempo y aumentar la ocupación media del bus sin perder garantías funcionales del sistema. Este proceso evita que los núcleos secundarios se queden sin cuota rápidamente y aumente su latencia.

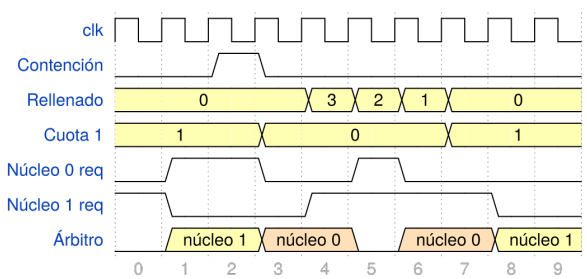


Fig. 3: Muestra del mecanismo de rellenado de cuotas

En la Figura 3 se muestra el ejemplo de la Figura 2 con el sistema de relleno de cuota añadido. En este caso el valor de relleno de cuota esta configurado a tres ciclos, y el valor de cuota rellenada por relleno esta configurado a un ciclo. De esta manera, en el ciclo 7 se rellena la cuota del núcleo 1 a un ciclo de contención y se desbloquea dicho núcleo. Pudiendo ser arbitrado en el ciclo 8 gracias al sistema de relleno de cuotas, pero no pudiendo ser arbitrado en el ciclo 5 para preservar las propiedades funcionales del sistema.

El sistema de rellenado tiene los siguientes parámetros configurables mediante un registro físico en el sistema:

- Rellenado activado.
- Ciclos de rellenado: Ciclos que tarda la cuota en ser rellenada desde que se agota.
- Cantidad de cuota rellenada: En el caso que se use el sistema de rellenado para prevenir la inanición de los núcleos secundarios mientras se garantiza los requerimientos temporales del sistema.

V. Resultados experimentales

En esta sección validamos el correcto funcionamiento del MCH, mostrando que cumple la función de controlar la interferencia entre núcleos. Para dicha prueba sintetizamos nuestro diseño del MCH en una Xilinx-VCU118. Dicho diseño está disponible en RTL en el repositorio publico del proyecto europeo al que pertenece este trabajo [4]. Conteniendo dicho repositorio el RTL del sistema RISC-V completo, junto a la MCCU en su versión mas reciente y los subsistemas específicos de este proyecto.

Listing 1: Código de la métrica progreso

```
1 volatile int progreso[4];
2 progreso[coreId] = 0;
3 while(1){
4     progreso[coreId]++;
5 }
```

En esta sección se describe el rendimiento de los núcleos secundarios basándonos en una métrica llamada progreso, dicha métrica la obtenemos ejecutando el código 1. Dicho código genera contención tanto de lectura como de escritura en el bus compartido para cada núcleo, siendo al final de la ejecución leída la métrica por el núcleo principal para calcular el progreso de los núcleos secundarios.

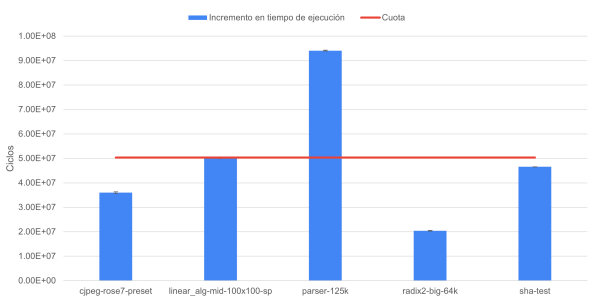


Fig. 4: Incremento de tiempo de ejecución de la tarea primaria cuando se ejecuta sola y con tareas secundarias limitadas mediante el MCH (Intervalo de confianza 99 %)

Como podemos observar en la Figura 4 el incremento de tiempo de ejecución de la tarea critica se mantiene generalmente por debajo de la cuota total

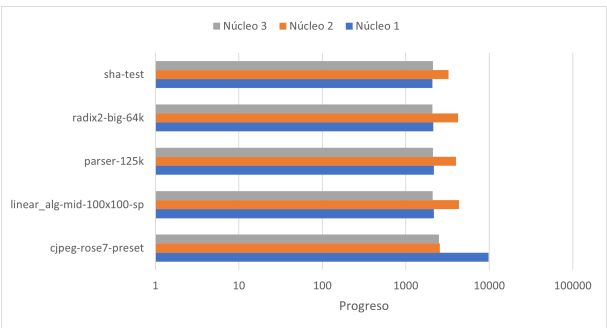
del sistema. De hecho, entre todas las ejecuciones, la máxima infracción de cuota producida ha sido de 68 ciclos. Estas infracciones de cuota son posibles debido a que la cuota se aplica en el periodo de arbitraje del bus, mientras que la contención se genera una vez arbitrado, por lo tanto, el desbordamiento máximo de la cuota seria igual a la longitud máxima de petición del bus.

Sin embargo, pese a que la cuota ha prevenido el exceso de contención en el bus, la Figura 4 muestra que, para parser-125k, la inflación temporal ha sido muy superior a la cuota asignada a los núcleos secundarios. Esto es debido a que la cuota únicamente actúa sobre la contención en el bus compartido, no en el resto de subsistemas compartidos del sistema, por lo que un patrón de accesos desafortunado de parser-125k puede haber causado mucha contención en el controlador de memoria, causando esta inflación del tiempo de ejecución superior a la esperada.

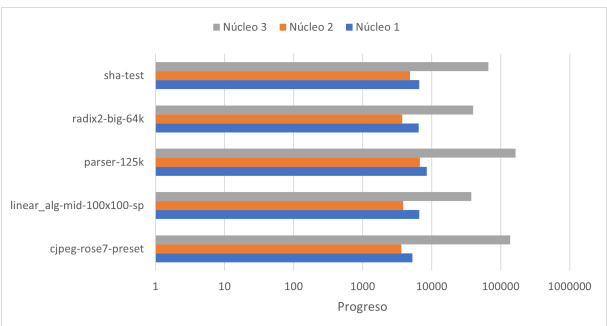
No obstante, pese a este caso mostrado con parser-125k, hay benchmarks que presentan el caso contrario, como radix2-big-64k. Este caso, si bien los núcleos secundarios han alcanzado su limite de cuota, este limite no se ha reflejado en una inflación significativa del tiempo de ejecución del núcleo primario. Esto puede ser debido a la mayor latencia de acceso del núcleo 0 siendo enmascarada por el write-buffer o por accesos favorables al controlador de memoria.

A. *Mecanismo de Cuotas Hardware*

Una vez probado el correcto funcionamiento de las cuotas sobre el incremento de tiempo de ejecución del núcleo principal podemos analizar el efecto de las cuotas sobre los núcleos secundarios.



(a) Round Robin



(b) Estático

Fig. 5: Progreso de los núcleos secundarios para un sistema con MCH y un valor de cuota de 0xFFFF

La Figura 5 representa el progreso de los núcleos secundarios para los diferentes *benchmarks* de la suite EEMBC CoreMark® [5] funcionando en el núcleo primario. En especifico, podemos observar como dicho progreso interactúa con las dos políticas de funcionamiento del bus presentadas en la sección III. En especifico nos interesan las diferencias de rendimiento entre ambas y su comportamiento patológico. Las diferencias de rendimiento entre ambas son importantes debido a que ambas ofrecen las mismas garantias de contención para el núcleo primario, por lo que un mayor rendimiento de un núcleo secundario representa un mayor rendimiento del sistema.

Si comparamos la Figura 5a y la Figura 5b podemos observar como el progreso para la política estática es muy superior que el progreso para la política Round Robin. Esto es debido a que la política estática da prioridad de arbitraje al núcleo prioritario, reduciendo así contención que podría ser prevenida en Round Robin dando prioridad a dicho núcleo. Sin embargo podemos observar que el progreso no es uniforme entre núcleos para ninguna de las políticas estudiadas. Esto es debido a la aparición sistemática de casos patológicos en el bus, donde hay determinados patrones que se repiten, un ejemplo de estos patrones es el efecto de sincronismo en el bus que se puede observar en [3], donde para un bus Round Robin concurrido se puede observar como este actúa como si fuera TDM.

Debido a estos casos patológicos podemos ver que, para la política estática, el núcleo 1 es favorecido sobre el núcleo 2 y el núcleo 3 es favorecido sobre estos dos. La ventaja del núcleo 1 sobre el núcleo 2 puede ser debida a que este tiende a ser arbitrado tras el núcleo 0, reduciéndose la probabilidad de que el núcleo 0 realice dos peticiones seguidas. Finalmente, la ventaja de un orden de magnitud que experimenta el núcleo 3 sobre el núcleo 1 y el núcleo 2 es debida a que este, al ser el ultimo arbitrado, es el que entrará al bus en los periodos que el bus tenga una ocupación menor, reduciéndose de esta manera la probabilidad de que el núcleo 0 requiera acceso al dicho bus. Sin embargo, en un sistema de prioridades esperamos que el núcleo 1 tenga una mayor prioridad que el núcleo 3, pero, debido a los casos patológicos descritos este recibe un orden de magnitud menos de rendimiento que el núcleo 3.

B. *Rellenado hardware de cuotas*

Para subsanar estos casos patológicos que reducen el rendimiento del sistema, homogeneizar el rendimiento de los núcleos secundarios y evitar la inanición de estos proponemos el sistema de rellenado hardware de cuotas presentado en la Sección IV. Este mecanismo rellena la cuota de los núcleos que la han agotado, repartiendo la cuota disponible en un periodo de arbitraje entre sus diferentes instantes. De esta manera los núcleos secundarios agotarán su cuota en periodos donde el núcleo 0 este muy activo, reduciendo la presión sobre el bus, para ser esta cuota rellenada mas tarde en un momento donde el núcleo 0 este menos activo.

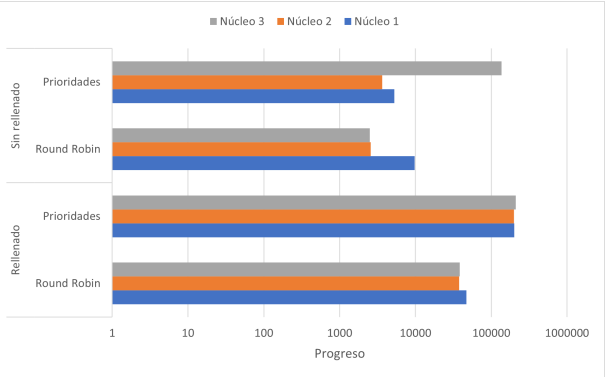


Fig. 6: Cjpeg-rose7-preset diferencia de progreso para un sistema con MCH con rellenado y sin rellenado de cuotas

De esta manera se obtienen los resultados presentados en la Figura 6, donde podemos observar que al añadir el rellenado, para la misma cuota final conseguimos un progreso marcadamente superior. Este sistema de rellenado mitiga los casos patológicos y redistribuye de manera mas acertada la cuota, consiguiendo un progreso similar entre todos los núcleos secundarios y un progreso marcadamente superior a un sistema sin redistribución de cuotas hardware.

VI. Trabajo relacionado

Si las garantías funcionales que una política de arbitraje estáticas garantiza no son suficientes para la tarea prioritaria, hace falta reducir la tasa de acceso del bus de manera artificial.

Hay trabajos que se centran en repartir el bus compartido en intervalos de tiempo, actuando dichos intervalos de tiempo como buses privados de cada núcleo para acceder al recurso compartido. Estos trabajos son conocidos como basados en *Time Division Multiplexing*(TDM). Mediante TDM se evita la contención a costa del rendimiento final del sistema, ya que si un actor es asignado un intervalo de tiempo en el bus debe consumirlo, necesite transmitir o no. H

En algunos artículos como en el de la BRU [6] se propone la introducción de un mecanismo hardware de control de contención mediante *rate-limiting*, es decir, se reduce la contención que sufre el núcleo primario reduciendo la tasa máxima de inyección al bus de los secundarios. De esta manera se aumenta la predecibilidad, pero se reduce el rendimiento total del sistema.

La MCCU [7], en la que se basa este trabajo, afronta el problema del control de contención en un bus compartido de una manera drásticamente diferente a lo propuesto en la BRU. Mientras que la BRU reduce la tasa de inyección máxima de los actores del bus por la contención que estos puedan causar, la MCCU monitoriza la contención real que los núcleos secundarios producen al primero y genera interrupciones una vez un cierto numero de ciclos de contención es alcanzado. De esta manera, la MCCU toma decisiones respecto a la contención real, y no respecto a la contención especulada que cierto ratio de inyección puede generar en el bus.

De esta manera, la MCCU consigue, para unos valores de cuota adecuados, un mayor rendimiento del sistema, ofreciendo la garantía de que no se sobrepasara una determinada contención en el bus, y por lo tanto, permitiendo que el sistema funcione a máximo rendimiento hasta que dicha contención ocurra. Sin embargo, el mecanismo de acción de la MCCU es criticado por la BRU [6] debido a su dependencia de las interrupciones para limitar la contención. De acuerdo a ciertos estándares de seguridad funcional [1], [8] es recomendable reducir al máximo el uso de interrupciones, que dificultan la certificación temporal del sistema.

VII. Conclusiones

En este artículo, proponemos un mecanismo de cuotas de hardware como herramienta eficaz para limitar las interferencias multinúcleo. De este modo, ofrecemos garantías a las plataformas multinúcleo para aplicaciones relacionadas con la seguridad. Demostramos la viabilidad del mecanismo propuesto con una implementación RTL en un procesador multinúcleo RISC-V industrial. La evaluación del mecanismo de cuota hardware confirma que dicho mecanismo es adecuado tanto para limitar la contención máxima en el núcleo crítico como para mejorar el rendimiento y su calidad de servicio en los núcleos secundarios

Agradecimientos

Este trabajo ha recibido financiación del proyecto SELENE del programa de investigación e innovación Horizon 2020 de la Unión Europea bajo el acuerdo de subvención nº871467.

Referencias

[1] ISO, "Iso 26262 - road vehicles functional safety," `https://www.iso.org/standard/68383.html`, 2021, (acceso: 2 de Septiembre de 2021).

[2] Alfons Crespo et al., "Mixed criticality in control systems," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 12261–12271, 2014, 19th IFAC World Congress.

[3] Gabriel Fernandez et al., "Increasing confidence on measurement-based contention bounds for real-time round-robin buses," in *Proceedings of the 52nd Annual Design Automation Conference*, New York, NY, USA, 2015, DAC '15, Association for Computing Machinery.

[4] "Repositorio de selene," `https://gitlab.com/selene-riscv-platform/selene-hardware`, (acceso: 27 de junio de 2022).

[5] Embedded microprocessor benchmark consortium, "Coremark-pro. an eembc benchmark," `https://www.eembc.org/coremark-pro/`, (acceso: 25 de Junio de 2021).

[6] Farzad Farshchi, Qijing Huang, and Heechul Yun, "Bru: Bandwidth regulation unit for real-time multicore processors," in *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2020, pp. 364–375.

[7] Jordi Cardona et al., "Maximum-contention control unit (MCCU): resource access count and contention time enforcement," in *Design, Automation & Test in Europe Conference & Exhibition, DATE 2019, Florence, Italy, March 25-29, 2019*, Jürgen Teich and Franco Fummi, Eds. 2019, pp. 710–715, IEEE.

[8] *IEC 61508 Ed 2, Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 1: General requirements*, 2010.