**PLIC**

**Índice:**

1

**1 Introducción** 2

2

**2 Flujo de las Interrupciones** 2

3

**3 Componentes y Adicionales** 2

4

**4 Conclusión** 2

5

**5** **Referencias** 2

**1 Introducción:**

Platform-Level Interrupt Controller o PLIC [1], es la definición de un controlador de interrupciones descrito para el estándar RISC-V [2], donde habilita un control más definido de las interrupciones externas y de los periféricos, o fuentes de interrupción. Sin embargo ,incluso cuando este controlador añade algo de complejidad a la implementación general, todavía es una implementación sencilla en comparación a otras del entorno como CLIC o APLIC.

Este controlador consta de dos partes principales: un Interruption Gateway, y el CORE. Dentro del Interruption Gateway se controla el flujo de interrupción de los periféricos. Mientras que en el Core se maneja el proceso de interrupción y e destino de la propia interrupción en sí misma, dentro del estándar entendemos como destino u objetivo al contexto de ejecución dentro de un mismo HART (Hardware Thread), que a su vez se puede definir como un nivel de privilegios dentro de ese HART: por ejemplo, el modo máquina y modo supervisor.

La imagen 1.1, sacada de la propia definición oficial, muestra una representación del direccionamiento de las interrupciones a los diferentes contextos con el controlador.

Diagrama

El contenido generado por IA puede ser incorrecto.Imagen 1.1 (Figura 1 dentro de la definición oficial [3])

**2 Flujo de las Interrupciones:**

El proceso de trabajo de este controlador se puede seguir de la siguiente manera: Primero el Gateway recibe una interrupción global de un periférico y la procesa, transformándola en una única notificación de interrupción que pasa al CORE. Una vez ha llegado al CORE este la procesa, y manda una notificación de interrupción a todos los objetivos (contextos) cuyo umbral de interrupción quede por debajo del valor de prioridad del periférico que ha interrumpido, entonces, uno y solo uno de los contextos reclamará la interrupción (el primero), siguiendo siempre una jerarquía de prioridades ente distintos periféricos, y cuando haya terminado de procesarla mandará un mensaje al de finalización al Gateway que podrá, en caso de ser necesario, procesar otra señal de interrupción de ese mismo periférico.

En la imagen 2.1, sacada del estándar oficial, podemos ver un pequeño diagrama que el flujo del proceso de una interrupción.

Diagrama

El contenido generado por IA puede ser incorrecto. Imagen 2.1 (Figura 2 dentro de la definición oficial [3])

**3 Componentes y Adicionales:**

Para todo el proceso de interrupción y de gestión de los periféricos necesitaremos una serie de registros o almacenes donde guardar la información de contexto, y para controlar el flujo y señalizar mensajes, por ejemplo: cuando un contexto quiere reclamar una interrupción o señalizar la terminación de la misma.

Interrupt Priority regs -> uno para cada periférico

Interrupt Pending bits -> un bit por cada periférico

Interrupt Enable regs -> por cada contexto

Priority Thresholds -> por cada contexto

Interrupt Claim register ->uno por cada contexto

Interrupt Completion regs -> el mismo que el de claim en muchos casos, igual uno por cada contexto.

**4 Conclusión:**

sdpipfj

# Referencias

|  |  |
| --- | --- |
| [1] | A. Chang, «riscv/riscv-plic-spec,» [En línea]. Available: https://github.com/riscv/riscv-plic-spec. |
| [2] | RISC-V, «Ratified RISC-V Specifications,» [En línea]. Available: https://lf-riscv.atlassian.net/wiki/spaces/HOME/pages/16154769/RISC-V+Technical+Specifications. |
| [3] | P. S. H. Committee, «RISC-V Platform-Level Interrupt Controller Specification,» [En línea]. Available: https://drive.google.com/file/d/1at94PNJl4v2eAsKIwKOsZWBxsVcY2U2F/view. |