Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computadores

Programa de Licenciatura en Ingeniería en Computadores

Curso: CE-4302 Arquitectura de Computadores II

Especificación Proyecto I: Modelado de un *Interconnect* para un sistema Multi-Procesador (MP)

Profesores:

Luis Alonso Barboza Artavia

Ronald García Fernández

Fecha de entrega: 03-04 de Abril, 2025

Semestre: I 2025

Objetivo general

Aplicar los conceptos de arquitectura de computadores II en el diseño e implementación de un modelo de simulación de un *Interconnect* parte de un sistema multi-procesador (MP), con el fin de

Atributos relacionados: Habilidades de Comunicación (HC).

Se comunica de manera efectiva e inclusiva sobre actividades de ingeniería complejas con la comunidad de ingenieros y con la sociedad en general, es capaz de comprender y escribir informes efectivos y documentación de diseño, hacer presentaciones efectivas, tomando en cuenta las diferencias culturales, de idioma y de aprendizaje.

Motivación

Los sistemas computacionales modernos se basan en sistemas multiprocesadores. Un elemento fundamental es el *Interconnect/Fabric* el cual se encarga de realizar la distribución de los mensajes de forma adecuada.

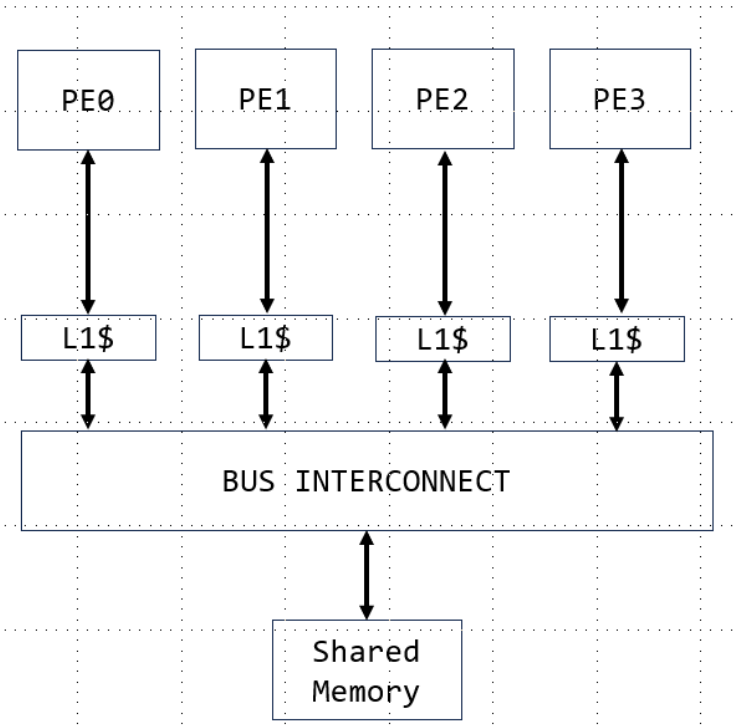

Figura 1. Ejemplo de un sistema MP con 4 PEs y su respectivo *Interconnect*.

Este proyecto tiene como objetivo diseñar e implementar un modelo simplificado que pueda simular un *Interconnect* y sistema MP, el cual permita analizar de forma cuantitativa las capacidades de diferentes esquemas de enrutamiento.

Características generales del modelo solicitado

1- El sistema debe ser desarrollado mediante algún lenguaje de programación (**Python no es permitido)** que permita *multithreading* con el fin de modelar los diferentes elementos de hardware (por ejemplo, C++20, Rust, o SystemVerilog*).

2- La memoria del sistema es compartida y tienen al menos 4096 posiciones para el almacenamiento de datos, el ancho de la memoria es de 32 bits, dicha memoria esta alineada por 4-bytes.

3- El sistema debe soportar como mínimo 8 PEs.

4- Cada PE tiene un cache privado de 128 bloques de 16 bytes.

5- El *Interconnect* soporta los siguientes mensajes:

Tabla 1. Mensajes soportados por el *Interconnect*

Mensaje/Operación	**Campos mínimos**	**Propósito**	**Notas**
`WRITE_MEM`	`SRC, ADDR, NUM_OF_CACHE_LINES START_CACHE_LINE, QoS`	Escribir en memoria los datos enviados por PE SRC la dirección ADDR	Los datos por escribir vienen del caché del PE SRC
`READ_MEM`	`SRC, ADDR, SIZE, QoS`	Leer de memoria un bloque de tamaño `SIZE` a partir de la dirección ADDR	El PE SRC almacena los datos en su caché
`BROADCAST_INVALIDATE`	`SRC CACHE_LINE, QoS`	Invalida la línea de cache `CACHE_LINE` de todos los PEs del sistema	
`INV_ACK`	`SRC, QoS`	Respuesta a la invalidación de un PE	
`INV_COMPLETE`	`DEST, QoS`	Respuesta que indica todas las invalidaciones fueron completadas	El PE que genera `BROADCAST_INVALIDATE` es quien recibe esta confirmación

READ_RESP	DEST, DATA, QoS	Datos de la Memoria principal correspondientes a READ_MEM	La cantidad de bytes debe ser consistente con el mensaje de READ_MEM
WRITE_RESP	DEST, STATUS, QoS	Respuesta a WRITE_MEM	STATUS es un campo de 1- byte indicando si la escritura fue exitosa: 0x1: OK 0x0: NOT_OK

QoS: representa un valor de prioridad asignado a cada PE de forma estática (antes de iniciar la simulación), dicho valor esta entre 0x00-0xFF

SRC: PE fuente del mensaje y tiene un valor único por PE asignado de forma estática SRC 0x00-0x07

SIZE: corresponde al número de bytes a leer de memoria principal.

ADDR: dirección destino/fuente

Note que los mensajes poseen campos o atributos mínimos, de ser necesario agregar otro este tiene que estar justificado (por ejemplo, agregar STATUS a READ_RESP).

6- Cada PE tiene una "memoria" de instrucciones privada y esta debe ser capaz de ser cargada con el código (mensajes) que se desea ejecutar, es decir el usuario puede definir el código que cada PE ejecuta ver figura 2:

```
1   PE2
2   WRITE_MEM 0xF004, 0x01, 0x02
3   WRITE_MEM 0xF014, 0x04, 0x01
4   WRITE_MEM 0xF024, 0x01, 0x04
5   READ_MEM  0xF034
6   BROADCAST_INVALIDATE 0x12
7   BROADCAST_INVALIDATE 0x11
8   BROADCAST_INVALIDATE 0x10
9   READ_MEM   0xF004
10  READ_MEM   0xF014
11  READ_MEM   0xF024
12  WRITE_MEM  0xF034, 0x02, 0x1f
```

Figura 2 ejemplo de "memoria"

Al ejecutar el sistema cada una de las instrucciones se traduce en un mensaje que contiene la información mínima necesaria según la tabla 1, dichos mensajes deben ser procesados por el *Interconnect*.

7- El *Interconnect* debe soportar 2 diferentes esquemas de arbitraje/calendarización FIFO (el valor de QoS es ignorado) y uno determinado por el valor QoS parte del mensaje.

8- Debe modelar el tiempo de transferencia de cada mensaje en función de su tamaño (en bytes) y además debe modelar el tiempo acceso a la memoria principal.

9- Está prohibido el uso de funciones como `sleep()`, o similares que consuman CPU time.

10- Al final de la ejecución de la simulación del sistema, se debe mostrar las estadísticas del *Interconnect* y los PEs esto con el fin permitir hacer un análisis de la ejecución en términos de eventos ocurridos, transmisión de datos, a través del tiempo, se deja la opción a cada grupo de trabajo el definir la forma de mostrar la información (gráficos, tablas, etc).

11- El sistema debe permitir la ejecución en pasos (*stepping*).

Requisitos de software

1- Utilizando las instrucciones soportadas por cada PE, diseñe 2 pruebas (workloads) para cada esquema con diferentes escenarios para el Interconnect. Todos los PEs deben utilizarse y como mínimo cada PE debe ejecutar 10 instrucciones.
2- Con cada una de las pruebas, genere gráficos de ancho de banda, tráfico en cada uno de los esquemas y otra estadística establecida por el grupo. Las gráficas se deben mostrar después de ejecutar la prueba. Para **visualizar** la parte gráfica pueden utilizar cualquier lenguaje de programación.

Notas:

1- Es obligatorio entregar el código fuente y un README, junto con la documentación e instrucciones para compilar y ejecutar el sistema (no se aceptan binarios como entregables)
2- Se recomienda definir una métrica/peso para evento para determinar el desempeño del sistema (estos tienen que estar basados en aspectos de sistemas computacionales reales)
3- Se recomienda una interfaz gráfica debe ser simple, esta es un medio para la interacción del usuario, pero no es el objetivo principal de proyecto.

Evaluación

El proyecto se desarrollará en grupos de 3 integrantes

La evaluación del proyecto se da bajos los siguientes rubros:

1- Presentación Funcional (**60%**) todos los miembros del grupo deben estar presentes en una sesión demostrativa (previa cita con el profesor) la funcionalidad del sistema, en la cual se realizarán preguntas sobre cualquier etapa del sistema, según la rúbrica correspondiente.

2- Artículo científico (**20%**) con la descripción del proceso de diseño del sistema y análisis de resultados.

3- Presentación del paper y resultados (**20%**): Cada grupo deberá grabar y entregar un video de 4:30 a 5:30 minutos presentando la idea de su solución (puede usar diapositivas u otro medio de apoyo), considerando que el público meta no tiene necesariamente el *background* técnico, por lo cual se debe exponer de forma clara el trabajo de realizado y los resultados más relevantes y conclusiones puede utilizar de referencia el siguiente formato. Por motivos de acreditación se recomienda vehemente que el enlace proporcionado esté disponible para cualquier persona por 1 año o más después de entregada la evaluación.

4- Artículo científico y presentación en inglés (10 %): se podrá optar por 10% extra si se realiza el articulo y la presentación en inglés (todo el contenido del video debe estar en inglés). Se otorgará únicamente si ambos entregables están en inglés.

Si tiene dudas puede contactar al profesor por medio de correo electrónico, la entrega debe ser realizada mediante el Tec Digital en la pestaña de evaluaciones no se aceptarán entregas después de las 11:59PM del 8 de mayo (grupo 2) y 9 de mayo (grupo 1) 2025.

Referencias

[1] John L Hennessy y David A Patterson. Computer Architecture: A Quantitative Approach. Elsevier, 2017

[2] Greaves, D. J. (n.d.). Modern System-on-Chip Design on Arm. Arm Education Media. ISBN 978-1-911531-36-4.