Configuración y Optimización de Sistemas de Cómputo
Interconexiones

Master Universitario en Ingeniería Informática
Depto. de Informática de Sistemas y Computadores (DISCA)
Universidad Politécnica de Valencia

Contenidos

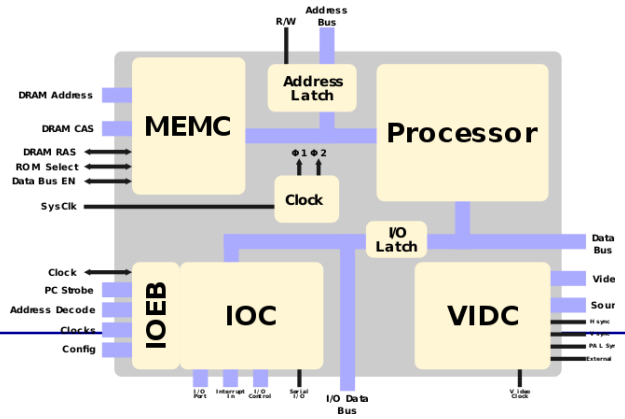
- Interconexiones en el Chip
  - AMBA, TileLink
- Interconexiones entre chips
  - NVLink, CCIX
  - Interconexiones en la placa
    - PCIexpres
    - North/South-Bridge
- Interconexiones entre nodos
  - Ethernet
  - Infiniband

Interconexiones en el Chip

- Aspectos generales de los Sistemas en Chip (SoC)
  - Mapa de memoria
  - Tipos de IPs (maestros/esclavos, procesadores/memorias, I/O)
  - Comunicaciones (Coherencia, no coherencia)
- Interfaces de comunicación
  - AMBA : AHB, APB, AXI, CHI
  - Otras especificaciones: TileLink, Whisbone, Core…
- Ejemplos Reales de Arquitecturas de Interconexión en SoC
  - Centric (bus-based/crossbar based), distribuidas, Jerarquicas

System-on-Chip (SoC)

- Integran muchas funcionalidades dentro de un mismo chip
  - Algunas de estas funcionalidades estaban anteriormente fuera del chip
  - Mayores niveles de integración permiten integrar más funcionalidades dentro del chip
    - Porque no metemos la memoria dentro?
  - Estas funcionalidades se integran en módulos hardware (referidos como IP blocks)



ARM250 Primer SoC basado en ARM. (@wikichip.org)

Comunicaciones en un SoC

- Son iniciadas generalmente por CPUs
  - Accesos a memoria
    - Frecuentes → Altas prestaciones
  - Configuración de registros
    - Ocasionales → No necesitan altas prestaciones
- E/S (I/O)
  - Datos de entrada/salida de periféricos
    - Altas prestaciones (Ethernet)
    - Bajas prestaciones (UART)
  - Transferencias a memoria a través de DMA
- En SoCs coherentes
  - Tráfico de coherencia de Cache
    - Comunicaciones entre las diferentes caches y niveles

System-on-Chip (SoC)

- ¿Como cooperan los diferentes IPs en un SoC?
  - Mapa de memoria
    - Define las coordenadas
  - Arquitectura de interconexión (bus, red en chip, etc)
    - Es el navegador GPS del SoC. Dada una dirección (coordenadas) te lleva al destino
  - Interfaces de comunicación
    - Establecen el "idioma" en el que se hablan los IPs

Mapa de memoria

- Define donde se mapean todos los IP blocks del

Core		Address range	Description
AHBROM		0xc0000000 - 0xc0100000	AHB boot ROM
AHBRAM		0x40000000 - 0x40100000	AHB RAM area
MIG		0x00000000 - 0x40000000	DDR4 memory
APBBRIDGE	APBUART	0xfc001000 - 0xfc001100	APB UART registers
	GRVERSION	0xfc081000 - 0xfc081100	Version and Revision register
	GPTIMER	0xfc000000 - 0xfc000100	Timer unit registers
	GRGPIO	0xfc083000 - 0xfc083100	General Purpose I/O port registers
	GRETH	0xfc084000 - 0xfc084100	Megabit/Gigabit Ethernet MAC registers
	GRSPW0	0xfc000600 - 0xfc000700	SpaceWire serial link 0 registers
	GRSPW1	0xfc000700 - 0xfc000800	SpaceWire serial link 1 registers
	AHBUART	0xfc000300 - 0xfc000400	AHB UART registers
	AHBSTAT	0xfc082000 - 0xfc082100	AHBSTAT registers
PLIC		0xf8000000 - 0xfc000000	Platform Level Interrupt Controller area
DEBUG MODULE		0xfe000000 - 0xff000000	Debug module area
CLINT		0xe0000000 - 0xe0100000	Core Local Interrupt Controller area
L2CACHE		0xff000000 - 0xff400000	L2 cache configuration registers
GRIOMMU		0xfffa0000 - 0xfffa0200	GRIOMMU configuration registers
AHBTRACE		0xfff00000 - 0xfff20000	AHB trace buffer area

DISCA

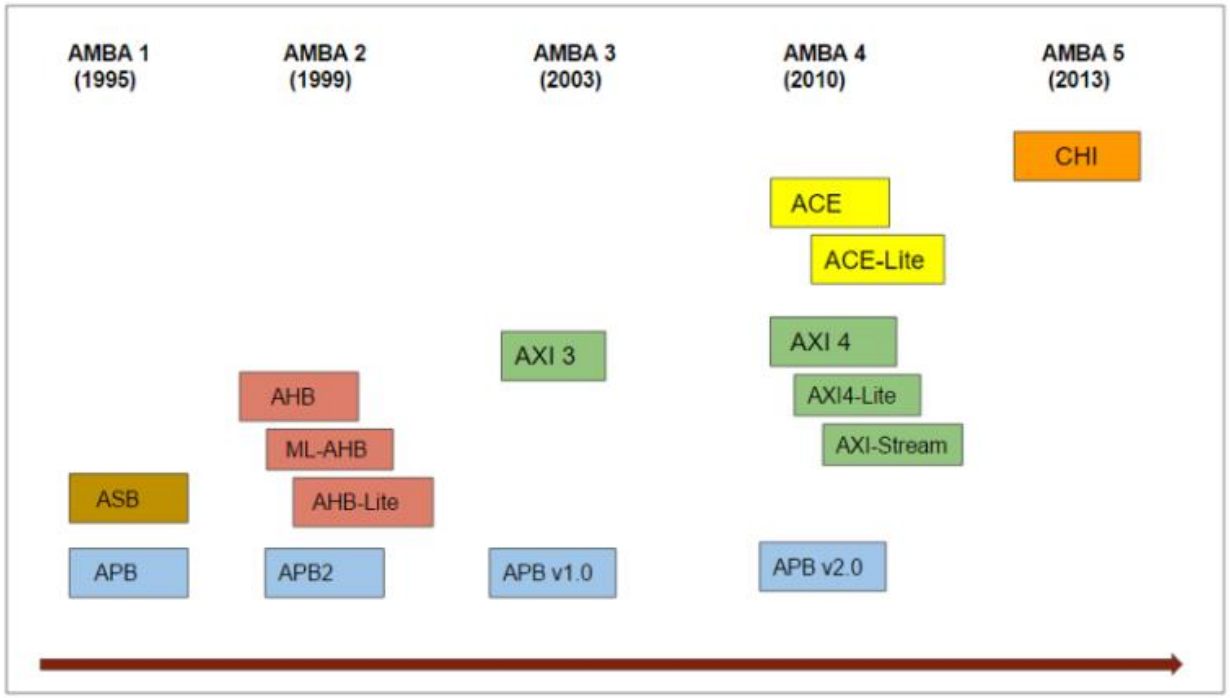
Architecturas de Interconnexión

- Se ajustan al tipo de comunicaciones
  - SoC con IPs no-coherentes (o parcialmente coherentes)
    - Bus, crossbar (no muchos Ips/cores)
    - Soluciones jerárquicas (número elevado de cores)
  - SoCs coherentes
    - Malla o Toro 2D
    - Soluciones híbridas: Malla + Bus, Malla + Anillo, Crossbar + Anillo
    - Otras topologías más complejas sufren de problemas de escalabilidad física
      - Enlaces largos, elevado número de puertos
      - Limitaciones máxima frecuencia, consumo y área

Interfaces de Comunicaciones

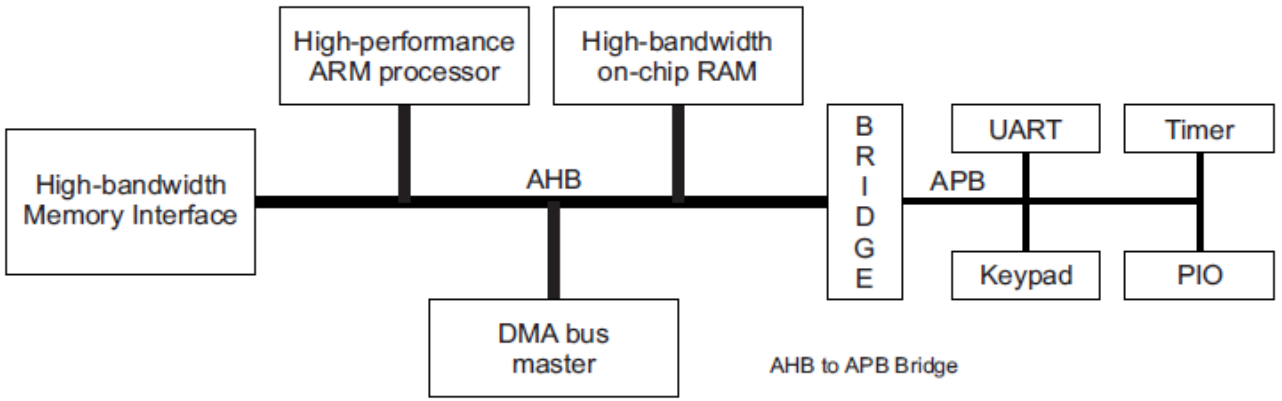
- Definen la interacción entre los IP blocks
- Permiten conexiones punto a punto (Maestro → Esclavo)
- Existen especificaciones estandarizadas
  - AMBA
  - TileLink
  - Whisbone
- Algunas especificaciones son abiertas
  - Permite la integración "sencilla" de IPs de varios fabricantes
  - Son compatibles con diferentes implementaciones que pueden ser patentadas
    - No revelan detalles de la arquitectura interna

Evolución del interfaz AMBA



AMB AHB

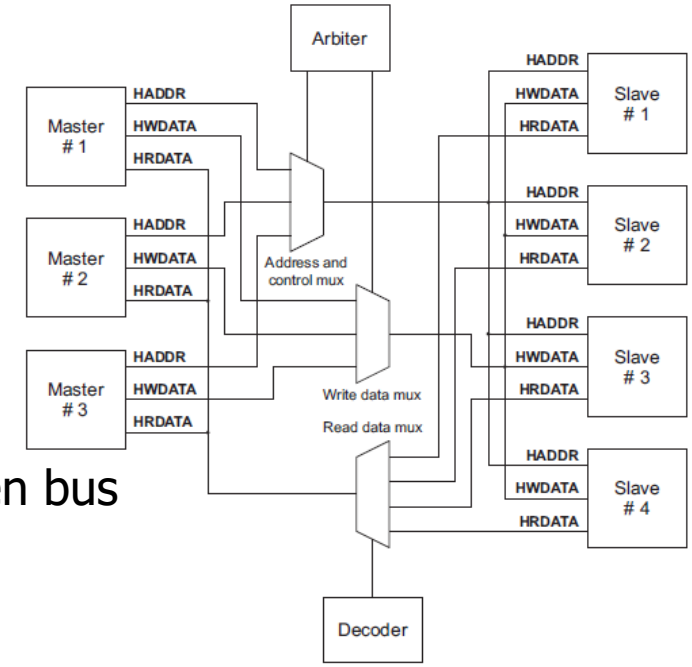
- Bus diseñado para conectar CPUs con memorias.
  - Fue diseñado cuando las CPUs de ARM no tenían mucha complejidad
    - No más de una transacción concurrente
  - Limitación de rendimiento para CPUs o IPs complejos
- Baja complejidad (parte positiva)
  - Elevada frecuencia de operación
  - Bajos requerimientos de área



AHB to APB Bridge

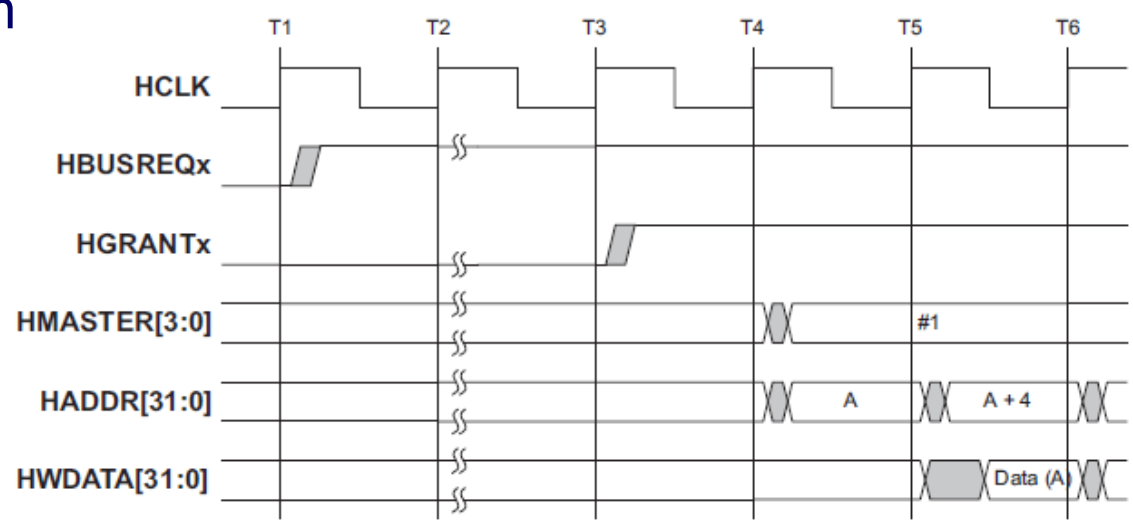
AHB Arquitectura

- Maestros
  - Interfaces representando componentes que inician transacciones
- Esclavos
  - Interfaces receptores de peticiones
- Decodificador
  - Dada una dirección determina el destino
- Arbitro
  - Selecciona el maestro que puede utilizar en bus



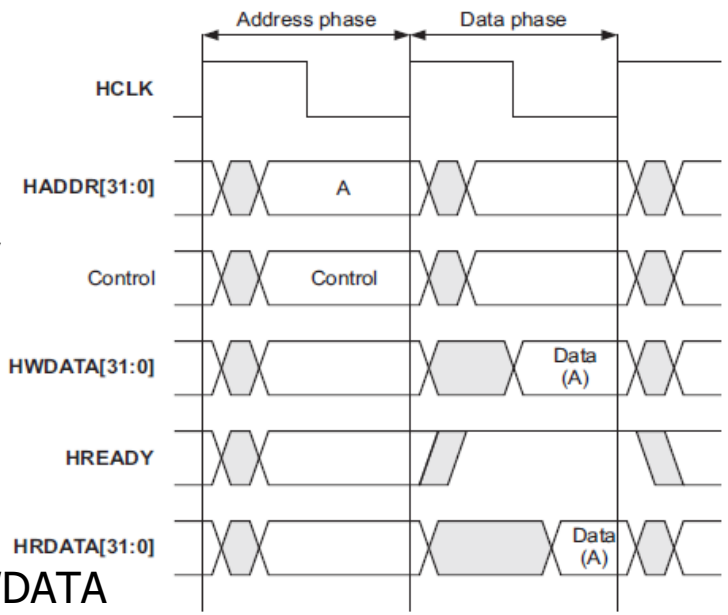
AHB Operación básica

- El arbitro decide quien accede al bus (HMASTER)
- Los maestros solicitan acceso con la señal de HBUSREQx
- Los maestros reciben notificación de acceso al bus con HGRANTx
- Una vez un maestro tiene concedido el acceso al bus puede iniciar la transacción



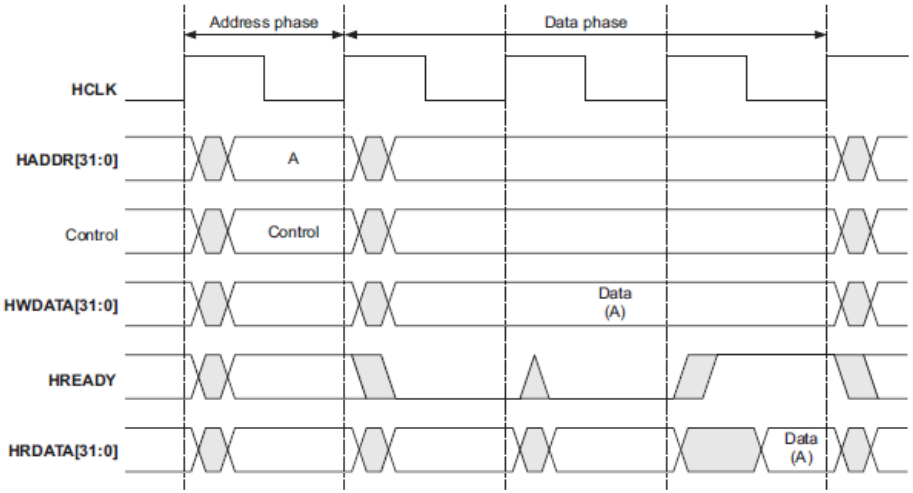
AHB Operación básica

- Transacción en 2 fases
  - Fase de Direcciones
  - Fase de Datos
- Fase Direcciones
  - Master proporciona la dirección y señales de control
    - Tamaño y tipo de transferencia
    - Propiedades (lock, cacheable)
- Fase de Datos
  - Operaciones de escritura
    - Master proporciona datos en HWDATA
    - El esclavo los acepta (HREADY=1)
  - Operaciones de lectura
    - El esclavo responde a la petición con los Datos en HRDATA



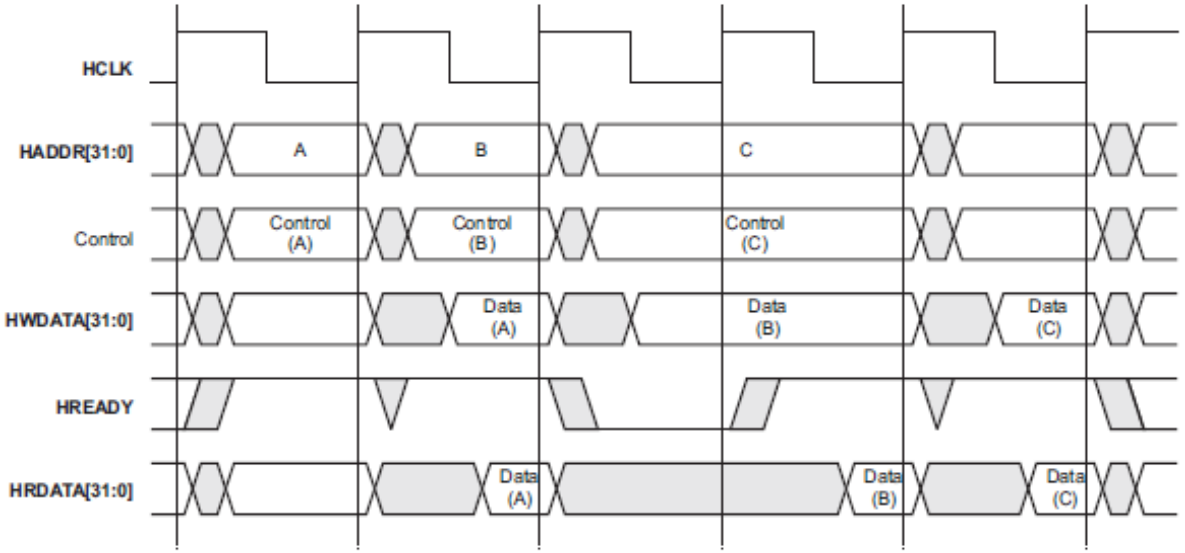
AHB transferencia con estados de espera

- Un esclavo puede tardar un tiempo en dar una respuesta
  - Ejemplo: La memoria necesita de varios ciclos para procesar una petición de lectura y proporcionar un dato
- HREADY
  - Señal que utiliza el esclavo para indicar que está listo para procesar la petición
  - HREADY =0 sirve para retrasar la respuesta
- CONTROL
  - Maestro → HTRANS (SEQ, NONSEQ, IDLE)
  - Esclavo → HRESP (OK, ERROR, RETRY, SPLIT)



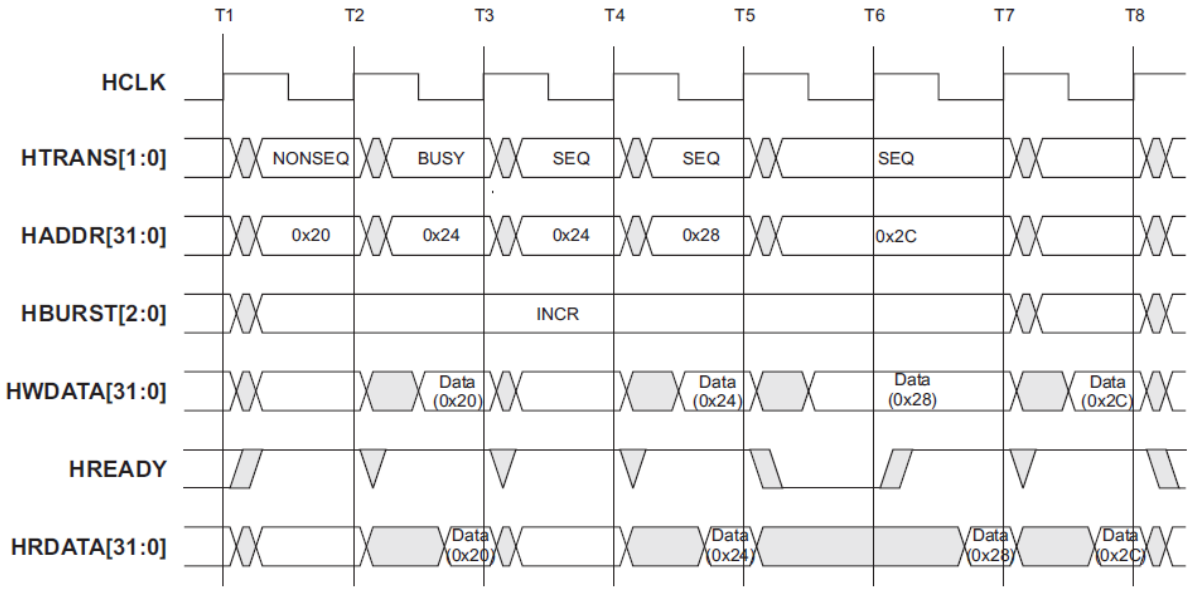
AHB múltiples transferencias

- Múltiples transferencias pueden ser entrelazadas sin perder ciclos
- AHB permite el solapamiento de fases de direcciones
- El esclavo puede alargar la respuesta
  - El master (transacción B en este caso) tiene ocupado el bus durante más tiempo



AHB ráfagas de datos

- Las ráfagas (burst) son útiles para transmitir bloques de datos
  - A parte de comunicaciones de tamaños soportados por el ISA de una CPU (byte, half, Word, etc) se transfieren bloques de datos (p.ej. Línea de cache)
- AHB soporta diferentes tipos de ráfagas
  - SINGLE
  - INCR
  - WRAP4
  - INCR4
  - …
  - WRAP16
  - INCR 16

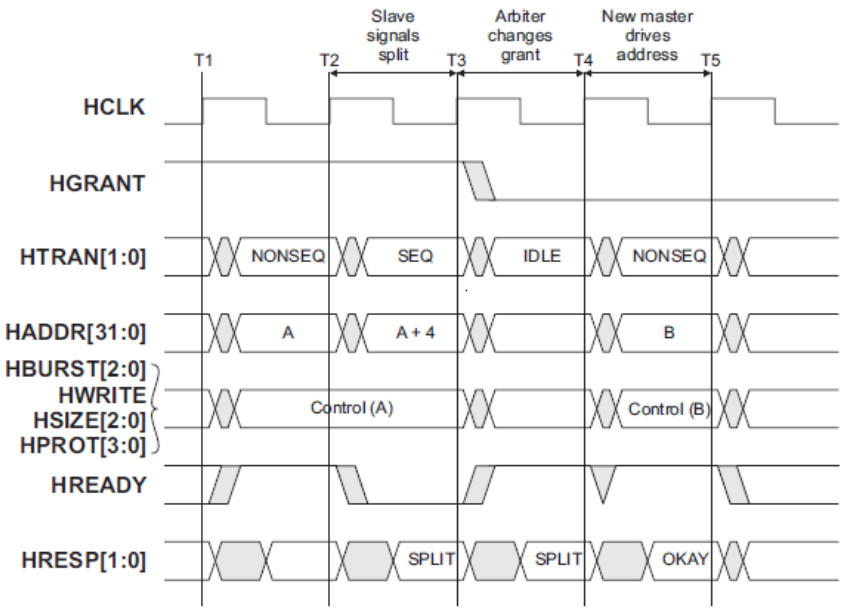


Preguntas

- ¿ Porque es mejor hacer una transacción tipo ráfaga que varias simples, si hemos visto que las simples se pueden solapar sin meter ciclos de espera?
- ¿Cuál es la utilidad de WRAP?

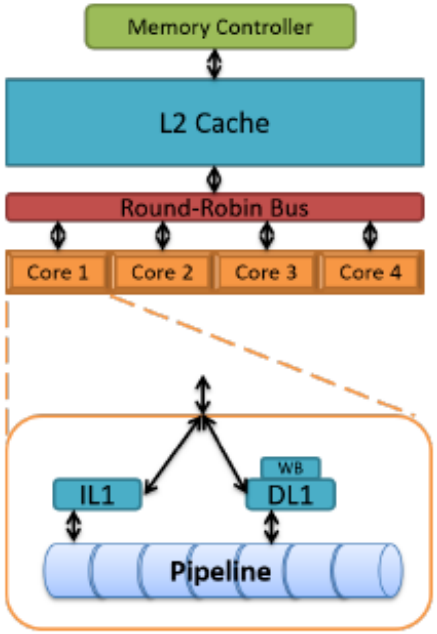
AHB Split Transactions

- Cuando un esclavo no puede proporcionar un dato en un tiempo razonable (HREADY=0) tiene ocupado el BUS impidiendo que otro maestro pueda acceder
- SOLUCIÓN: Utilizar la funcionalidad de "SPLIT transaction"



AHB Ejercicio

- Calcular el tiempo la **contención máxima** que puede sufrir una tarea en el core1 (T1) debido a la interferencia en el Bus.

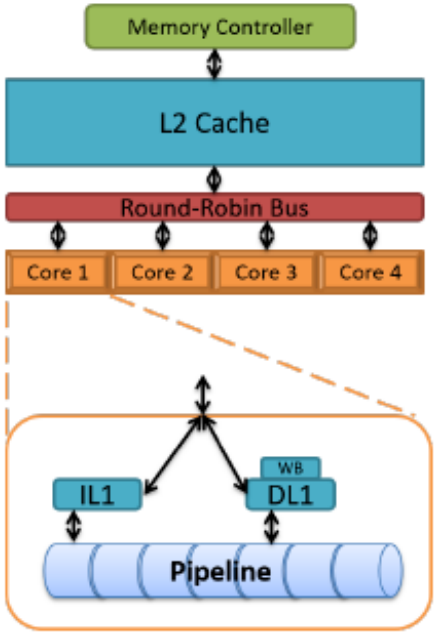


Características del sistema:

- La tarea T1 realiza 100 peticiones a memoria que se resuelven como “hit” en la L2 Cache
- El tiempo de acierto en L2 es 5 ciclos
- El tiempo de fallo en L2 es 20 ciclos (tiene en cuenta el tiempo a memoria)
- El bus es round-robin y **no soporta SPLIT** transactions

AHB Ejercicio

- Calcular el tiempo la **contención máxima** que puede sufrir una tarea en el core1 (T1) debido a la interferencia en el Bus.



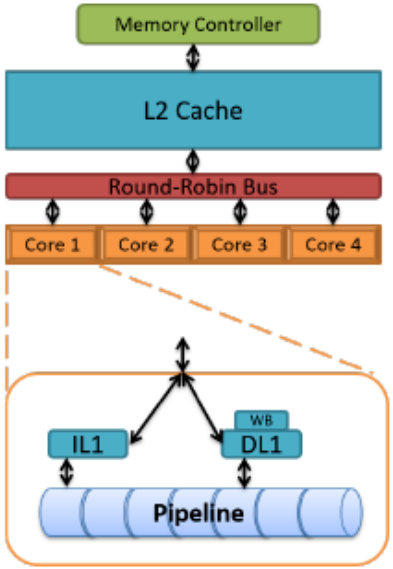
Características del sistema:

- La tarea T1 realiza 100 peticiones a memoria que se resuelven como “hit” en la L2 Cache
- El tiempo de acierto en L2 es 5 ciclos
- El tiempo de fallo en L2 es 20 ciclos (tiene en cuenta el tiempo a memoria)
- El bus es round-robin y **no soporta SPLIT** transactions

Solución: 100*3 *(20) = 6000 ciclos

AHB Ejercicio

- Calcular el tiempo la **contención máxima** que puede sufrir una tarea en el core1 (T1) debido a la interferencia en el Bus.

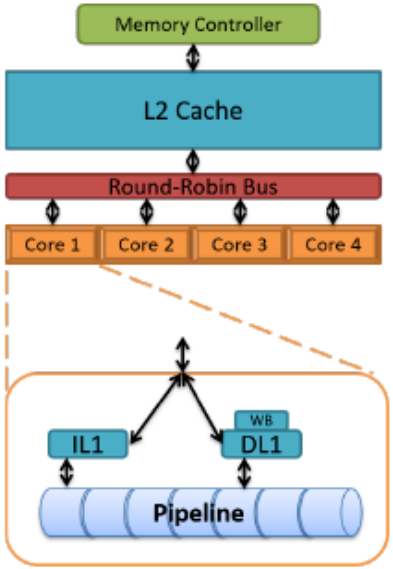


Características del sistema:

- La tarea T1 realiza 100 peticiones a memoria que se resuelven como "hit" en la L2 Cache
- El tiempo de acierto en L2 es 5 ciclos
- El tiempo de fallo en L2 es 20 ciclos (tiene en cuenta el tiempo a memoria)
- El bus es round-robin y la L2 señaliza **SPLIT** cuando determina que es un fallo de L2 (5 ciclos)

AHB Ejercicio

- Calcular el tiempo la **contención máxima** que puede sufrir una tarea en el core1 (T1) debido a la interferencia en el Bus.



Características del sistema:

- La tarea T1 realiza 100 peticiones a memoria que se resuelven como "hit" en la L2 Cache
- El tiempo de acierto en L2 es 5 ciclos
- El tiempo de fallo en L2 es 20 ciclos (tiene en cuenta el tiempo a memoria)
- El bus es round-robin y la L2 señaliza **SPLIT** cuando determina que es un fallo de L2 (5 ciclos)

Solución: 100*3 *(5) = 1500 ciclos

AHB Lock

- AHB da soporte a operaciones atómicas habilitando una señal del lock
- Cuando un master realiza una petición con la señal de LOCK=1 esta petición no se puede interrumpir
  - No se pude hacer SPLIT
  - Esto garantiza la consistencia de operaciones atómicas tipo LDST (Load&Store)
  - El coste en prestaciones es elevando

APB Conectando Periféricos

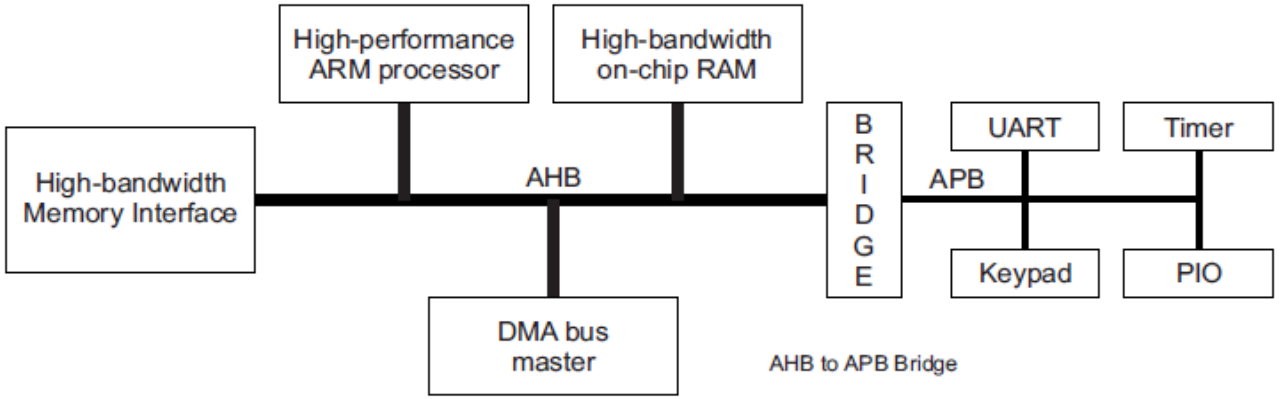
- Muchos de los componentes de un SoC no necesitan tanto ancho de banda
  - Utilizar AHB resulta ineficiente
    - Elevado consumo
    - Elevados recursos de área
- AMBA define el standard APB para conectar este tipo de dispositivos
- Ejemplos de periféricos que se conectan con un APB
  - UART
  - Timer

AMBA APB

- APB (Advanced Peripheral Bus)
  - Diseñado para conectar la CPU a periféricos con bajos requerimientos de ancho de banda
- AHB Implementa las siguientes funcionalidades que APB no tiene
  - Burst transfers
  - Split transactions
  - Single cycle bus master handover
  - Single clock edge operation
  - wider data bus configurations (64/128 bits).

APB/AHB coexistencia

- Es normal encontrar SoCs donde CPU se comunica con los periféricos con APB y a la memoria con AHB
- Para que puedan coexistir necesitamos traducir de AHB a APB
  - Generalmente se utiliza un "bridge"
  - Este componente actúa como esclavo AHB y como maestro APB

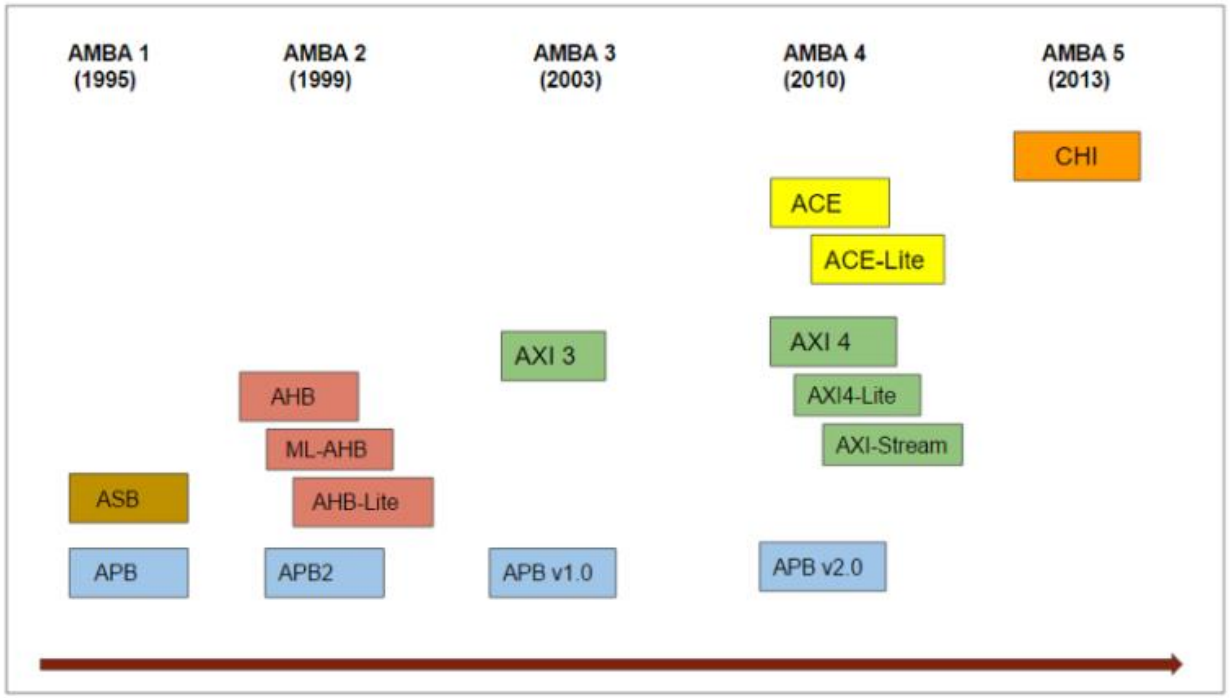


AHB to APB Bridge

Limitaciones AHB

- AHB tiene limitaciones
  - Protocolo (handshake) es lento (3 ciclos desde petición a respuesta)
  - Elevado tiempo de bloqueo
    - Se puede limitar con el uso de SPLIT (aumenta la complejidad del arbitro)
  - Paralelismo limitado
- Estas limitaciones son más importantes cuando tenemos:
  - Elevado número de IPs (masters y slaves)
  - IPs capaces de soportar operaciones concurrentes
    - Procesadores superescalares
    - Controladores de Memoria (DDR3/4/5)
  - Requerimientos de tiempo real
    - Ráfagas de tamaño indefinido (arbitrariamente largas)

Interfaz AXI (Avanced Extensible Interface)

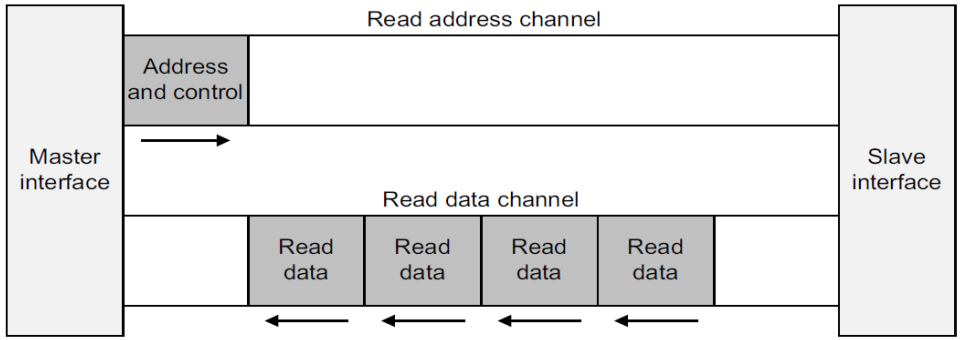


AXI Funcionalidades (no soportadas en AHB)

- Fase de Direcciones y de Datos Desacoplada
- Accessos de datos no alineados utilizando señales de strobe
- Ráfagas a partir de una dirección inicial (no hay que pasar todas las direcciones)
- Canales de Escritura y Lectura Separados
- Multiples peticiones en “vuelo”
- Posibilidad de procesar las peticiones fuera de orden

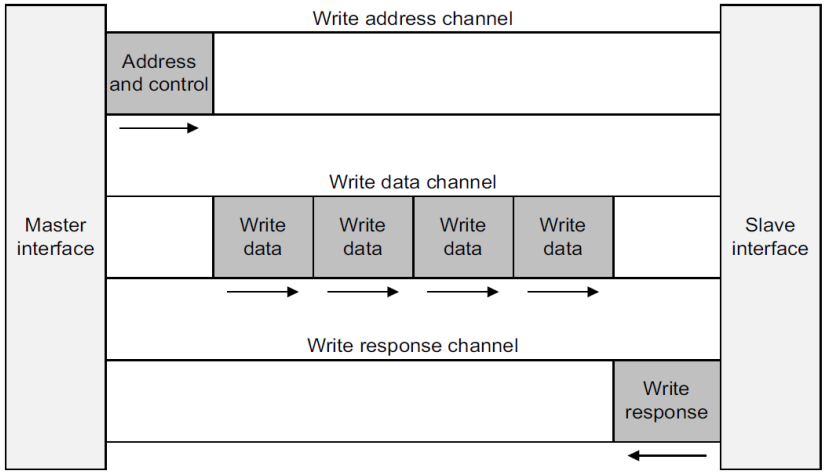
AXI Lecturas de datos

- Para completar una transacción de lectura AXI gasta dos canales
  - AREAD (Address Read): Canal donde el maestro realiza peticiones de lectura
  - RDATA (Read Data): Canal que utiliza el esclavo para proporcionar los datos requeridos (siempre como respuesta a una petición AREAD).
  - Ambos canales utilizan protocolos (handshake) de intercambio de datos/control independientes
    - Señales ready/valid independientes



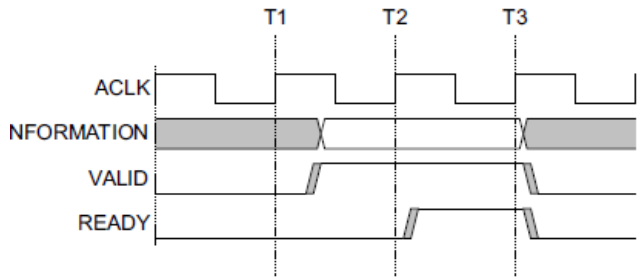
AXI Escrituras de Datos

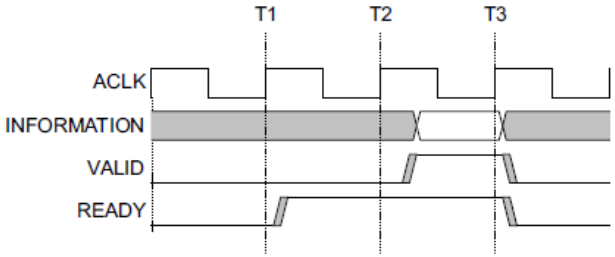
- Para completar una transacción de escritura AXI gasta tres canales
  - AWRITE (Address Write): Canal donde el maestro realiza peticiones de escritura
  - WDATA (Write Data): Canal que utiliza el maestro para proporcionar los datos requeridos (siempre asociada a una petición en AWRITE).
  - B (Write response): El esclavo utiliza este canal para notificar que se ha procesador la escritura
  - Todos los canales utilizan protocolos de Sincronización independientes

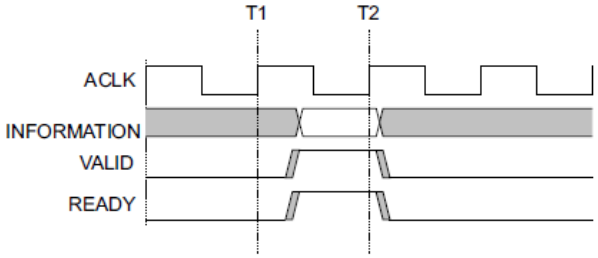


AXI Transacciones, control de flujo

- Control de flujo entre transmisor y receptor tipo “backpressure”
  - La fuente (maestro) genera la señal de valid
  - El destino genera la señal de ready
  - La transacción ocurre cuando ready=‘1’ & valid =‘1’

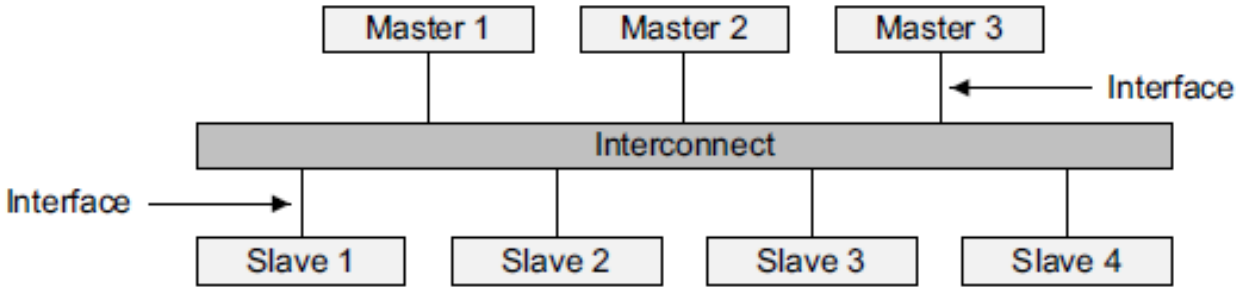






AXI Arquitectura

- La arquitectura de un SoC basado en AXI es similar a la de uno basado en AHB
- AXI no obstante favorece la posibilidad de utilizar NoCs (Redes en el Chip) como arquitectura de interconexión
  - Las peticiones son NO bloqueantes por defecto
    - Envio petición y libero el recurso
  - El negociado de canales (handshake) es equivalente al control de flujo de las NoC (wormhole)



DISCA

AXI Atributos Transacciones de Memoria

- AXI expande el tipo de atributos en las operaciones de acceso a memoria
  - Los esclavos no necesitan conocer el mapa de memoria y/o los atributos

ARCACHE[3:0]	AWCACHE[3:0]	Memory type
0b0000	0b0000	Device Non-bufferable
0b0001	0b0001	Device Bufferable
0b0010	0b0010	Normal Non-cacheable Non-bufferable
0b0011	0b0011	Normal Non-cacheable Bufferable
0b1010	0b0110	Write-Through No-Allocate
0b1110 (0b0110)	0b0110	Write-Through Read-Allocate
0b1010	0b1110 (0b1010)	Write-Through Write-Allocate
0b1110	0b1110	Write-Through Read and Write-Allocate
0b1011	0b0111	Write-Back No-Allocate
0b1111 (0b0111)	0b0111	Write-Back Read-Allocate
0b1011	0b1111 (0b1011)	Write-Back Write-Allocate
0b1111	0b1111	Write-Back Read and Write-Allocate

DISCA

AXI Concurrencia

- Las transacciones de AXI utilizan un Identificador (ID)
- Los IDs permiten:
  - Partir una transacción y asegurarse que se procesa en orden. Transacciones con un mismo ID tienen que devolverse en orden
  - Procesamiento paralelo de transacciones. Un master puede enviar transacciones con más de un ID sin necesidad de que termine transacciones anteriores
- El otro mecanismo básico de concurrencia que permite AXI lo proporcionan los canales
  - Canales separados de Lectura y escritura
  - Procesamiento concurrente de Direcciones y Datos.

AXI Ordenamiento de Transacciones

- AXI soporta un ordenamiento de transacciones basados en los IDs
- Las transacciones con el mismo ID:
  - Con destino un periférico deben llegar al periférico en el mismo orden en el que fueron emitidas
  - Con destino un dispositivo de memoria que utilizan la mismas direcciones (o con solapamiento) deben llegar a la memoria en el mismo orden en el que fueron
- El cumplimiento de este principio requiere que los maestros, esclavos y la red se diseñen para que esto pueda ser garantizado

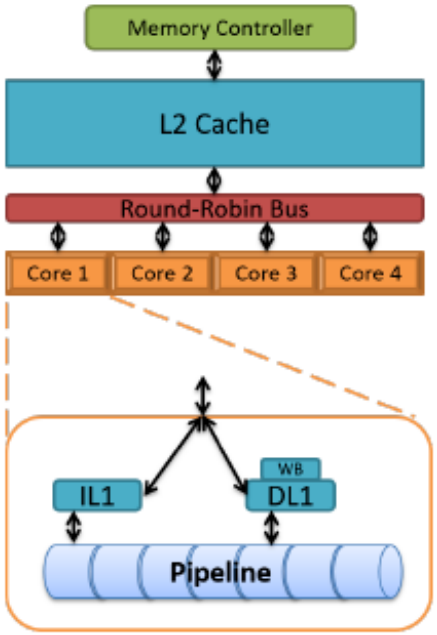
AXI Extensiones de coherencia (ACE)

- El protocol ACE extiende el interfaz AXI4 proporcionando soporte para coherencia de cache. El protocol ACE implementa:
  - Modelo de cache de 5 estados (atributo a linea de cache).
    - El estado determina que acciones se deben realizer cuando se accede a esta linea
  - Nuevas señales en el interfaz de AXI (AXI4 en particular)
    - Permitir nuevas transacciones
    - Definición de dominio de coherencia
  - Canales adicionales para permitir la comunicación entre diferentes maestros de líneas de cache compartidas (snoops)

AXI Extensiones de Coherencia (ACE)

- Tres nuevos canales
  - Snoop Address
    - Entrada a Master (con cache) que proporciona la direccion y las señales de control asociadas
  - Snoop Response Channel
    - Salida de un master (con cache) que proporciona la respuesta a una transaccion de snoop
  - Snoop Data Channel (opcional)
    - Canal para que un master pueda proporcionar el dato al que se hace "snoop"

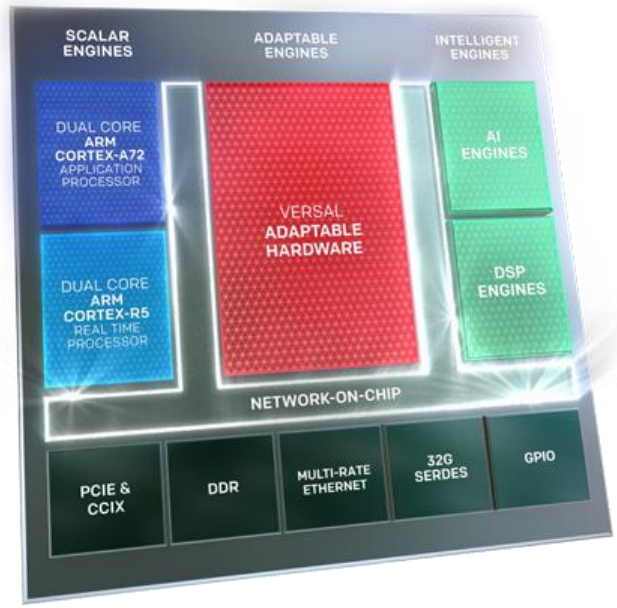
AXI Extensiones de Coherencia (ACE)

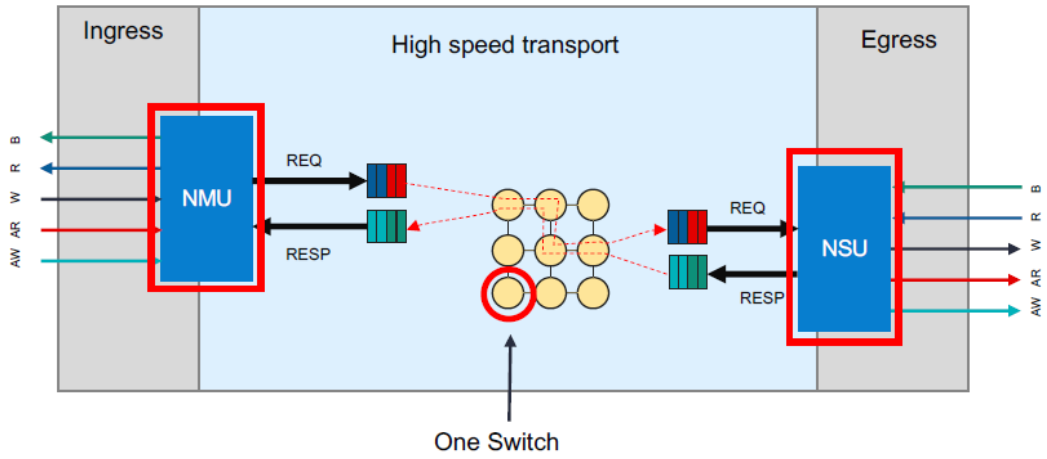


Ejemplo:

- Multicore con L1 write-through
  - Utiliza Snoop Address y Snoop Response Channel
  - Datos Actualizados siempre en la L2
  - Snoop Address Channel para notificar escrituras de los otros cores y invalidar líneas de cache
- Multicore con L1 write-back
  - Utiliza los 3 canales
  - El canal "Snoop Data" se utiliza por ejemplo cuando se hace una petición de lectura para devolver el dato cuando coincide con la "Snoop Address"

SoCs basados en AXI

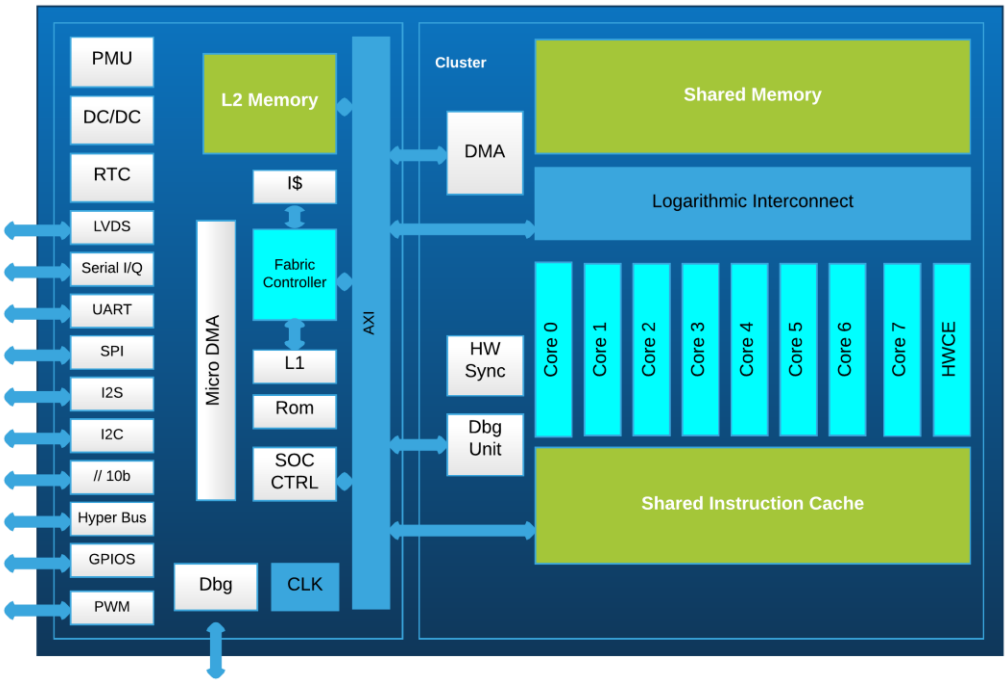




XILINX VERSAL (https://www.xilinx.com/products/silicon-devices/acap/versal-ai-core.html)

HotChips2020_FPGA_Xilinx_Versal_Premium_MV Presentation

SoCs basados en AXI



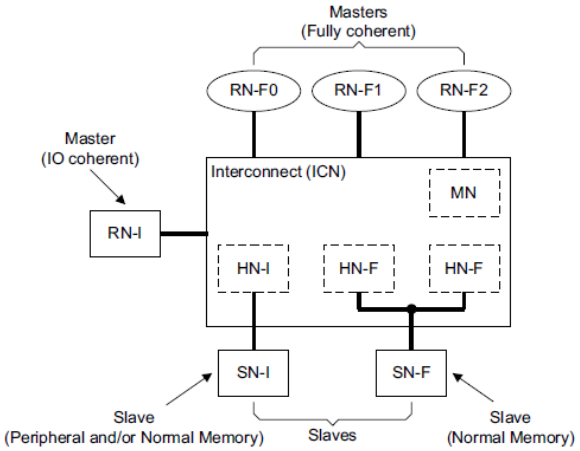
Greenwaves RISC-V GAP8 Open-source processor (https://greenwaves-technologies.com/)

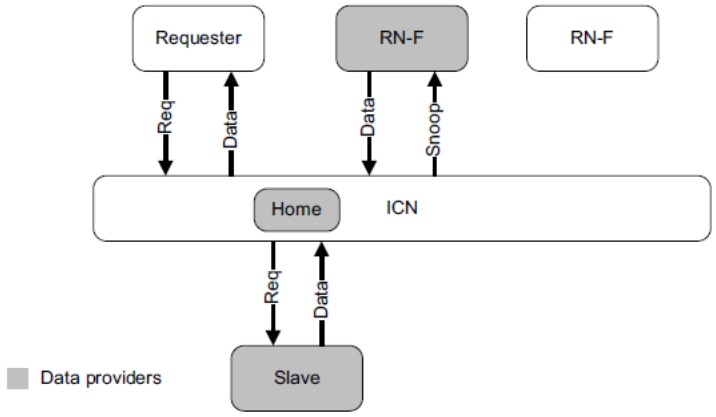
AMBA CHI (Coherent Hub Interface)

- Arquitectura escalable para SoCs coherentes
  - Amplio rango desde sistemas pequeños/medios a grandes
- Abstrae y proporciona funcionalidades de comunicación en distintas capas
  - Protocolo (de coherencia)
  - Red
  - Enlace
- Comunicación basada en paquetes (NoC "pura")
- Transacciones orquestadas por un "home-node" que cordina los snoops, accesos a cache y memorias
- CHI soporta protocolos de coherencia:
  - Basados en snoop filter y directory based (para que pueda escalar).
  - MESI y MOESI

CHI Tipos de Nodos

- RN (request node) nodo que genera peticiones a la red (lecturas/escrituras)
  - Atributos F/I/D (Fully coherent, IO coherent, I/O coherent with VM support)
- SN (slave node) nodo que recibe peticiones y proporciona respuestas
  - Atributos F/I (Fully coherent, I/O coherent)
- HN (home node)
  - Atributos F/I (Fully coherent, No coherent)

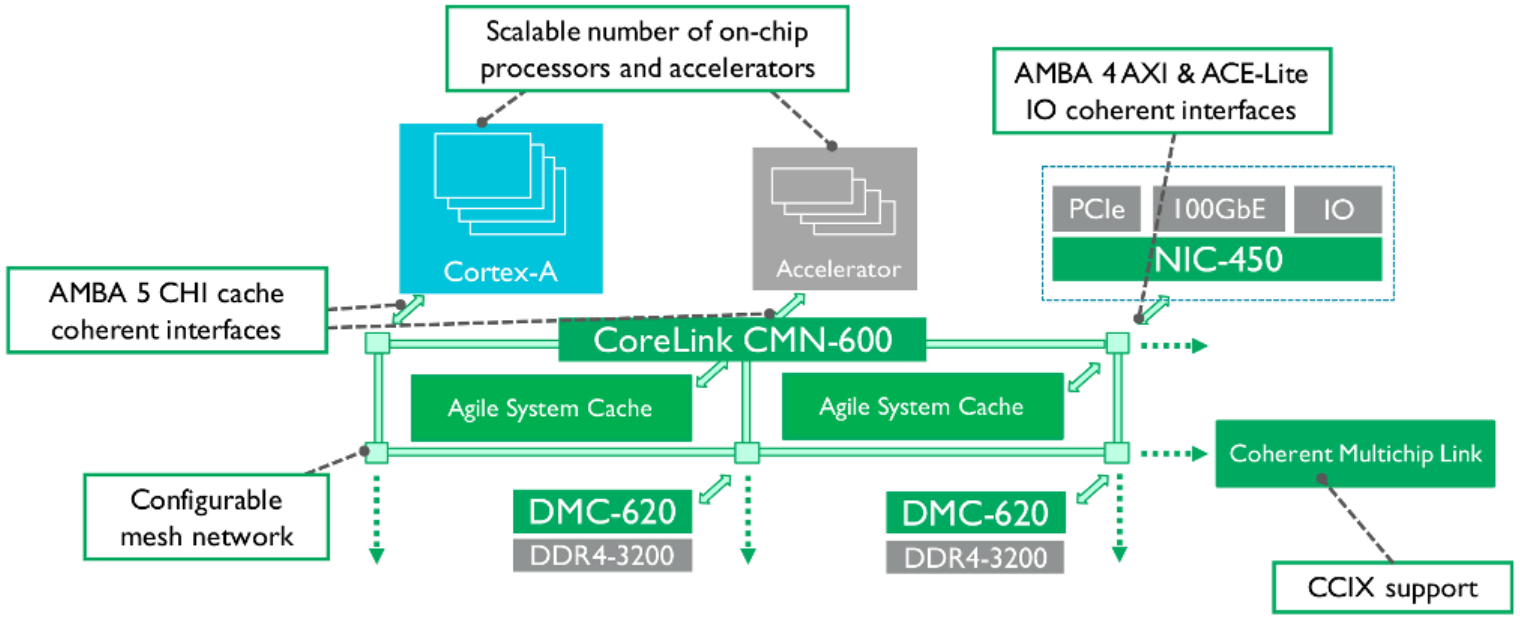




CHI

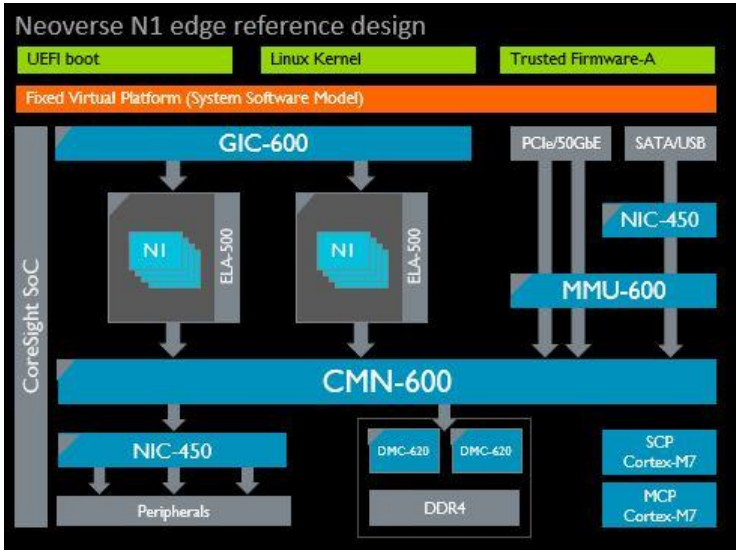
- Facilita la interconexión coherente de CPUs
- La implementación es compleja
  - Proporciona todo lo necesario para tener sistema coherente de forma estandarizada
  - Limita su adopción
    - A dia de hoy las implementaciones de CHI son todas de ARM
      - Neoverse platforms
    - No todos los protocolos de coherencia se ajustan bien (insuficiente o demasiado)
    - Muchos procesadores manycore basados en ARM implementan su propia NoC (sin compatibilidad CHI)
      - CAVIUM ThunderX
      - Fugaku

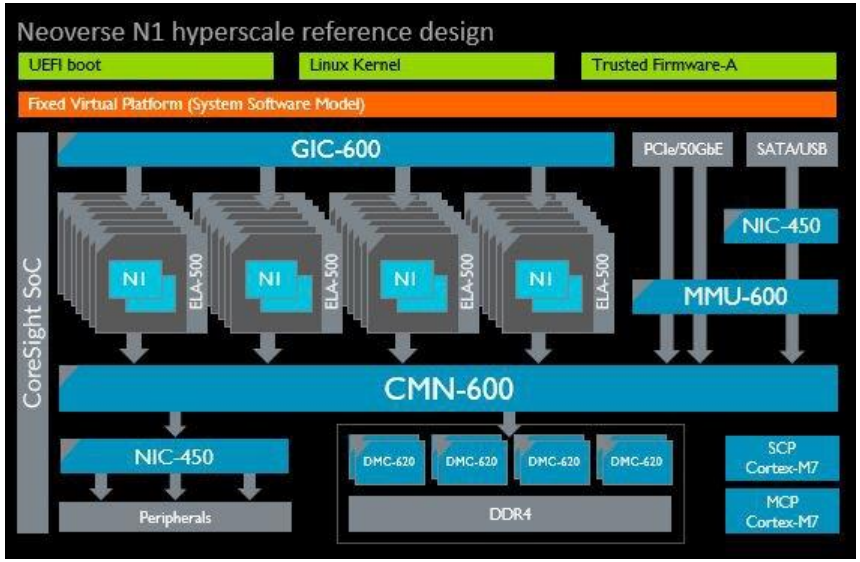
NoCs basadas en CHI: CoreLink CMN-600



Ejemplos de SoCs basados en CHI

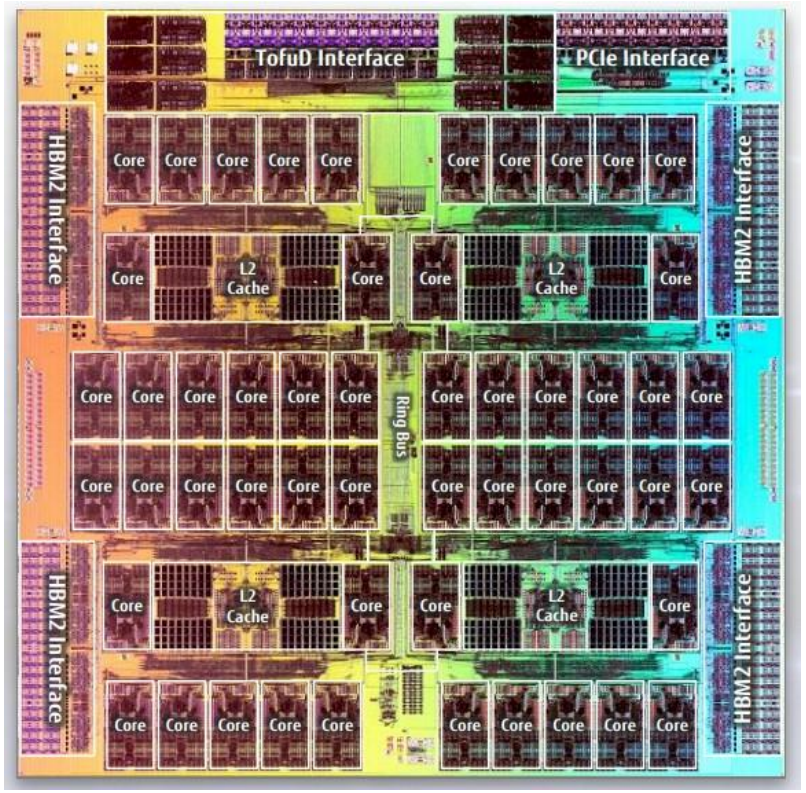
- Neoverse Platforms
  - Amazon Graviton Server se basa en NEOVERSE
  - Implementa la NoC CMN-600





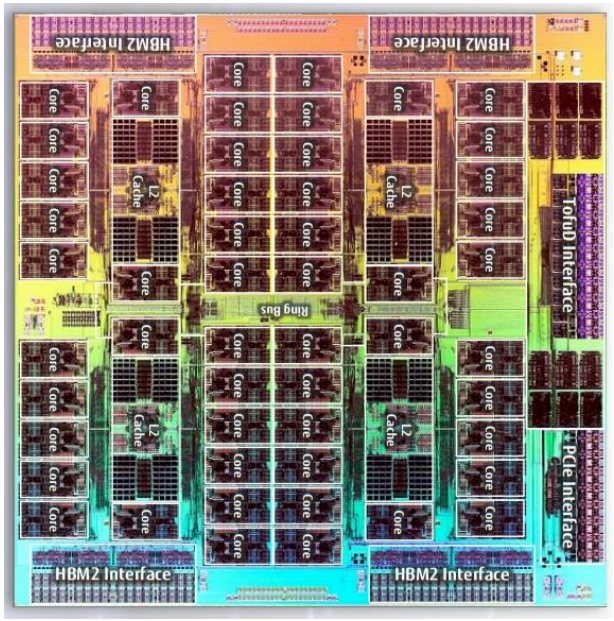
SoCs para HPC

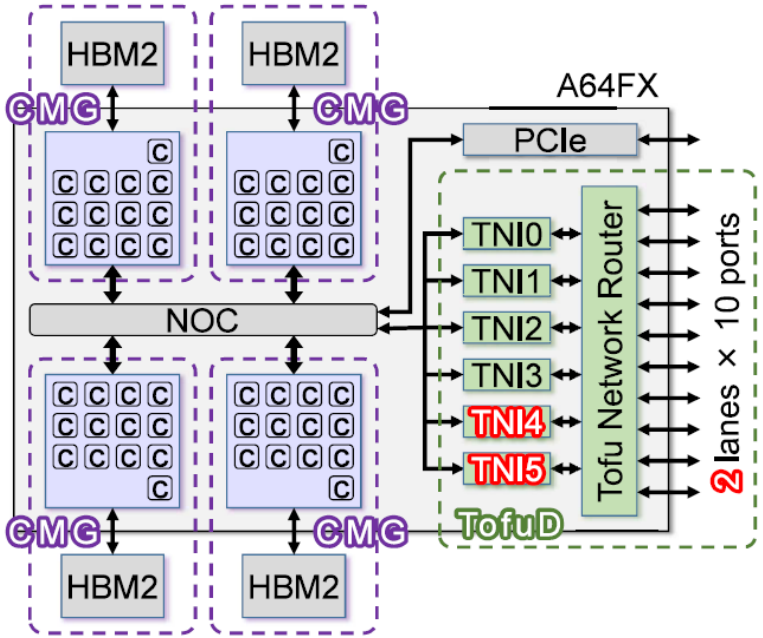
- Fugaku A64FX (7nm)
  - N1 en el TOP500 (Junio 2020)
  - Basado en ARMv8
  - Implementa SVE de 512bits
- On-chip Network
  - Topologia?



SoCs para HPC

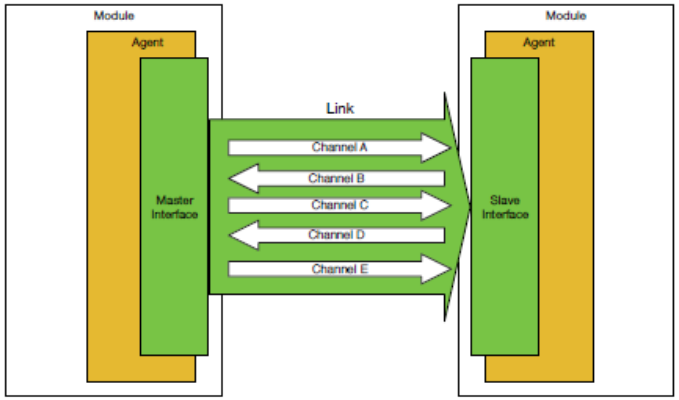
- Fugaku A64FX (7nm) On-chip Network





Otras specificaciones

- **TileLink** es un standard de comunicaciones on-chip originalmente desarrollado por la universidad de Berkley
  - Actualmente se utiliza fundamentalmente for Sifive (procesadores RISC-V)
- Originalm etamente coherentes



- Posteriormente sacaron una versión para sistemas no coherentes
- Se puede hacer compatibles con AXI (muchas similitudes)

Bibliografia

AMBA® 5 CHI Architecture Specification

AMBA® AXI™ and ACE™ Protocol Specification

AMBA™ Specification (Rev 2.0)

https://hotchips.org/

https://developer.arm.com/ip-products/system-ip/corelink-interconnect/corelink-coherent-mesh-network-family

AN INTRODUCTION TO CCIX™ WHITE PAPER

https://www.fujitsu.com/global/Images/the-tofu-interconnect-d-for-supercomputer-fugaku.pdf

SiFive TileLink Specification SiFive, 2019