SISTEMI EMBEDDED
AA 2013/2014

Nios II processor:

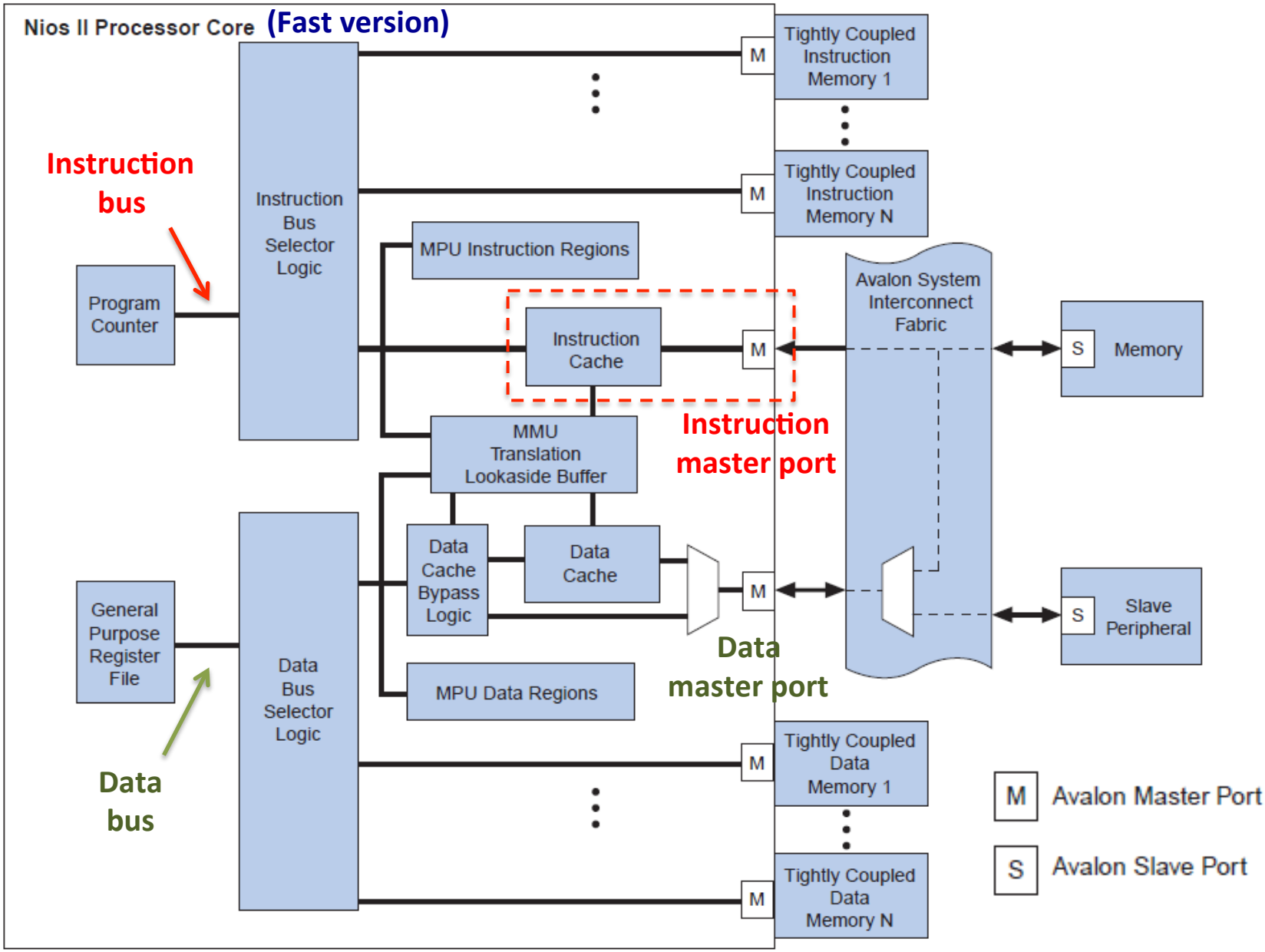
Memory organization and access

Federico Baronti

Memory and I/O access (1)

- **Instruction master port**: An Avalon Memory-Mapped (Avalon-MM) master port that connects to instruction memory via system interconnect fabric
  - **Instruction cache**: Fast cache memory internal to the Nios II core
- **Data master port**: An Avalon-MM master port that connects to data memory and peripherals via system interconnect fabric
  - **Data cache**: Fast cache memory internal to the Nios II core
- **Tightly-coupled instruction or data memory port:** Interface to fast on-chip memory outside the Nios II core

Memory and I/O access (2)



Memory and I/O access (3)

- Separate instruction and data busses, as in Harvard architecture
- Both data memory and peripherals are mapped into the address space of the data master port
  - The Nios II uses little-endian byte ordering
- Quantity, type, and connection of memory and peripherals are system-dependent
  - Typically, Nios II processor systems contain a mix of fast on-chip memory and slower off-chip memory
  - Peripherals typically reside on-chip, although interfaces to off-chip peripherals also exist

Instruction master port

- 32-bit pipelined Avalon-MM master port
- Used to fetch instructions to be exectued by the processor
- The pipeline support increases the throughput of synchrounous memory w/ pipeline latency
- The Nios II can prefetch sequential instructions and perform branch prediction to keep the instruction pipe as active as possible
- Always retrieves 32-bit data thanks to dynamic bus sizing logic embedded in the system interconnect fabric

Data master port

- The Nios II data bus is implemented as a 32-bit Avalon-MM master port. The data master port performs two functions:
  - Read data from memory or a peripheral when the processor executes a load instruction
  - Write data to memory or a peripheral when the processor executes a store instruction

Cache (1)

- The Nios II architecture supports cache memories on both the instruction master port (instruction cache) and the data master port (data cache)
- Cache memory resides on-chip as an integral part of the Nios II processor core
- The cache memories can improve the average memory access time for Nios II processor systems that use slow off-chip memory such as SDRAM for program and data storage
- The instruction and data caches are enabled perpetually at run-time, but methods are provided for software to bypass the data cache so that peripheral accesses do not return cached data

Cache (2)

- The cache memories are optional. The need for higher memory performance (and by association, the need for cache memory) is application dependent
- Cache use improves performance if:
  - Regular memory is located off-chip, and access time is long compared to on-chip memory
  - The largest, performance-critical instruction loop is smaller than the instruction cache
  - The largest block of performance-critical data is smaller than the data cache

Cache bypass methods

- The Nios II architecture provides the following methods for bypassing the data cache:
- **I/O load and store instructions**
  - The load and store I/O instructions such as ldio and stio bypass the data cache and force an Avalon-MM data transfer to a specified address
- **Bit-31 cache bypass**
  - The bit-31 cache bypass method on the data master port uses bit 31 of the address as a tag that indicates whether the processor should transfer data to/from cache, or bypass it

Load instructions

ldw / ldwio — **load 32-bit word from memory or I/O peripheral**

Operation:	rB ← Mem32[rA + σ (IMM14)]
Assembler Syntax:	`ldw rB, byte_offset(rA)` `ldwio rB, byte_offset(rA)`
Example:	`ldw r6, 100(r5)`
Description:	Computes the effective byte address specified by the sum of rA and the instruction's signed 16-bit immediate value. Loads register rB with the memory word located at the effective byte address. The effective byte address must be word aligned. If the byte address is not a multiple of 4, the operation is undefined.
Usage:	In processors with a data cache, this instruction may retrieve the desired data from the cache instead of from memory. Use the `ldwio` instruction for peripheral I/O. In processors with a data cache, `ldwio` bypasses the cache and memory. Use the `ldwio` instruction for peripheral I/O. In processors with a data cache, `ldwio` bypasses the cache and is guaranteed to generate an Avalon-MM data transfer. In processors without a data cache, `ldwio` acts like `ldw`.

Store instructions

stw / stwio — **store word to memory or I/O peripheral**

Operation:	Mem32[rA + σ (IMM16)] ← rB
Assembler Syntax:	`stw rB, byte_offset(rA)`
	`stwio rB, byte_offset(rA)`
Example:	`stw r6, 100(r5)`
Description:	Computes the effective byte address specified by the sum of rA and the instruction's signed 16-bit immediate value. Stores rB to the memory location specified by the effective byte address. The effective byte address must be word aligned. If the byte address is not a multiple of 4, the operation is undefined.
Usage:	In processors with a data cache, this instruction may not generate an Avalon-MM data transfer immediately. Use the `stwio` instruction for peripheral I/O. In processors with a data cache, `stwio` bypasses the cache and is guaranteed to generate an Avalon-MM bus cycle. In processors without a data cache, `stwio` acts like `stw`.

Use load and store IO in C

- Use IORD_32DIRECT(), IOWR_32DIRECT(),… macros defined in io.h
- Use compiler flags:
  - -mbypass-cache, -mno-bypass-cache
    Force all load and store instructions to always bypass cache by using I/O variants of the instructions. The default is not to bypass the cache.
  - -mno-cache-volatile, -mcache-volatile
    Volatile memory access bypass the cache using the I/O variants of the load and store instructions. The default is not to bypass the cache.

Characteristics of the Nios II cores

Feature		Core		
		Nios II/e	Nios II/s	Nios II/f
Instruction Bus	Cache	–	512 bytes to 64 KB	512 bytes to 64 KB
	Pipelined Memory Access	–	Yes	Yes
	Branch Prediction	–	Static	Dynamic
	Tightly-Coupled Memory	–	Optional	Optional
Data Bus	Cache	–	–	512 bytes to 64 KB
	Pipelined Memory Access	–	–	–
	Cache Bypass Methods	–	–	■ I/O instructions ■ Bit-31 cache bypass ■ Optional MMU
	Tightly-Coupled Memory	–	–	Optional
⋮				
External Address Space		2 GB	2 GB	2 GB without MMU 4 GB with MMU

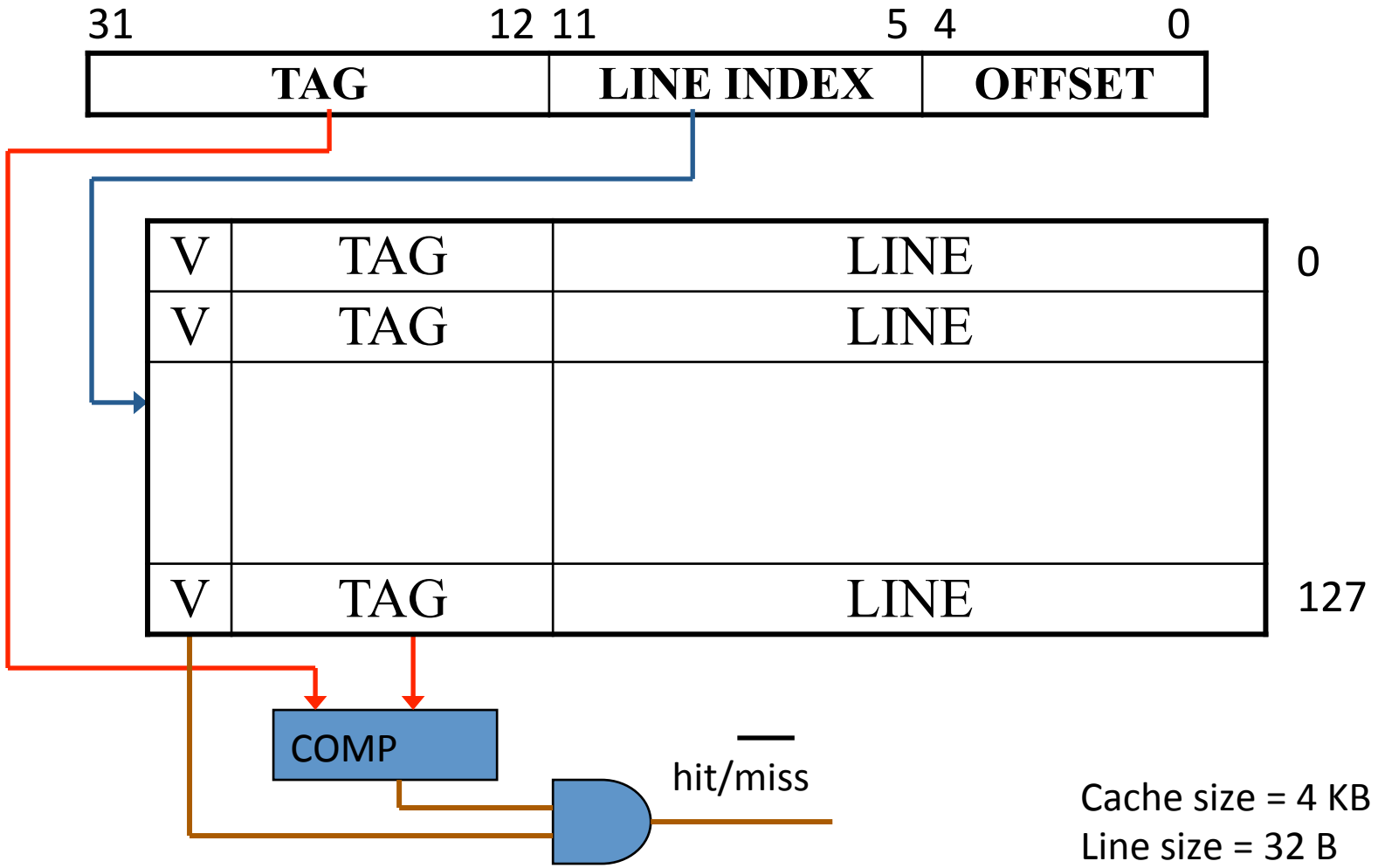
Nios II/f: instr and data cache (1)

- **Instruction Cache**
  - Direct-mapped cache implementation
  - 32 bytes (8 words) per cache line
  - The instruction master port reads an entire cache line at a time from memory, and issues one read per clock cycle

Nios II/f: instr and data cache (2)

- **Data Cache**
  - Direct-mapped cache implementation
  - Configurable line size of 4, 16, or 32 bytes
  - The data master port reads an entire cache line at a time from memory, and issues one read per clock cycle
  - Write-back
  - Write-allocate (i.e., on a store instruction, a cache miss allocates the line for that address)

Cache implementation



Cache size = 4 KB
Line size = 32 B

Tightly-coupled memories (1)

- Tightly-coupled memory provides guaranteed low-latency memory access for performance-critical applications. Compared to cache memory, tightly-coupled memory provides the following benefits:
  - Performance similar to cache memory
  - Programmer can guarantee that performance-critical code or data is located in tightly-coupled memory
  - No real-time caching overhead, such as loading, invalidating, or flushing memory

Tightly-coupled memories (2)

- Physically, a tightly-coupled memory port is a separate master port on the Nios II processor core, similar to the instruction or data master port
- A Nios II core can have zero, one, or multiple tightly-coupled memories
- The Nios II architecture supports tightly-coupled memory for both instruction and data access
- Each tightly-coupled memory port connects directly to exactly one memory with guaranteed low, fixed latency
- **The memory is external to the Nios II core and is located on chip**

Tightly-coupled memories (3)

- Tightly-coupled memories occupy normal address space, the same as other memory devices connected via system interconnect fabric
  - The address ranges for tightly-coupled memories (if any) are determined at system generation time
- Software accesses tightly-coupled memory using regular load and store instructions
- From the software's perspective, there is no difference accessing tightly-coupled memories compared to other memories

References

- Altera, “Nios II Processor Reference Handbook,” *n2cpu_niiv1.pdf*
  - 2. Processor architecture/Memory and I/O organization