DESIGN SPEC DOCUMENT

ECE-593: Fundamentals of Pre-Silicon Validation
Maseeh College of Engineering and Computer Science
Winter, 2025

Project Name: Multi-Processor System with Memory Controller and Shared Bus
Members: Riley Cox, Nathan Fraly, Jeff McHale
Date:1/24/26

Project Name	Multi-Processor System with Memory Controller and Shared Bus
Location	Portland State University
Start Date	1/14/26
Estimated Finish Date	3/4/26
Completed Date	

Prepared by: Team 1	
Prepared for: Prof. Venkatesh Patil	
Team Member Name	Email
Nathan Fraly	nfraly@pdx.edu
Riley Cox	rilco@pdx.edu
Jeff McHale	jmchale@pdx.edu

Design Features:
Three cores with identical architecture composed of an ALU, Instruction Unit and Memory Interface Unit
Round Robin Arbiter using Two Priority Arbiters and a Mask
Per-core private cache, 128B capacity, 4-way set associative, 4B line size
Snooping MESI protocol, write-back, write-allocate with random replacement (LRU under consideration)
Shared memory architecture - all cores access a single 2KB byte addressable pool of memory (2048 x 1 Byte)
16 opcodes supported by ISA (ALU - arithmetic/logical/shifts/special functions, MIU - load/store)
Each core has its own register file of 32 x 32-bit general purpose registers
All arithmetic is 32-bit signed with wraparound to handle overflow. Shift operations are logical (zero fill on LSR, zero insert on LSL)
Fixed 32-bit instruction encoding with two formats: I-type (loads/stores), R-type (register/register ALU ops)
Modular SystemVerilog design utilizing packages, interfaces and parameterized widths, promoting maintainability and extensibility
Raise flag + NOP undefined opcodes

Project Description:
A three core multiprocessor system with round-robin arbiter and private per-core caches. Design follows a simplified ISA, utilizing a load-store architecture with fixed encoding and immediate addressing. Each core implements identical architecture composed of an Instruction Unit, an ALU and a Memory Interface Unit (MIU). The Instruction Unit decodes 32-bit fixed instructions to determine whether to service ALU or MIU operations. The ALU supports signed arithmetic, logical, shift and special function behavior operating on 32-bit operands with wraparound on overflow; shift operations are purely logical. The MIU services all load/store instructions using immediate addressing by issuing requests to the private cache controller. On a miss, the cache fetches the missing line from the shared 2KB byte addressable main memory through the arbiter. The ISA defines two instruction layouts: I-type for load/store operations and R-type for register/register ALU operations. Each core possesses its own register file of 32 x 32-bit general purpose registers. Reserved bits are held for future expansion of the system. Unimplemented opcodes are flagged and treated as a NOP to avoid undefined behavior, ensuring deterministic execution. Cache organization consists of a private cache per core, each 128B capacity, 4-way set associative with 4B line size. The caches implement a snooping MESI invalidation protocol to maintain coherence, with write-back and write-allocate policies. A random replacement policy will be used (LRU under consideration).

Important Signals/Flags	Purpose
core_ready_flag	Core can accept a new instruction
alu_busy_flag	ALU still performing operation, stall
mem_busy_flag	Load/store in progress, stall
instruction_done_flag	Indicates current instruction has completed
illegal_opcode_flag	Inputted opcode is not defined
cache_miss_flag	Used to debug cache behavior
addr_fault_flag	Raised for out of range address

Design Signals	Instruction Unit & Register File
rf_addr_a[4:0]	Read address for selected register (a)
rf_addr_b[4:0]	Read address for selected register (b)
rf_data_a[31:0]	Read data from rf_addr_a
rf_data_b[31:0]	Read data from rf_addr_b
rf_wen	Write enable
rf_write_addr[4:0]	Destination register address
rf_write_data[31:0]	Write data to destination register

Design Signals	Instruction Unit & ALU
alu_op[3:0]	Select ALU operation
alu_a[31:0]	Operand A
alu_b[31:0]	Operand B
alu_result[31:0]	ALU output
alu_done	Computation complete

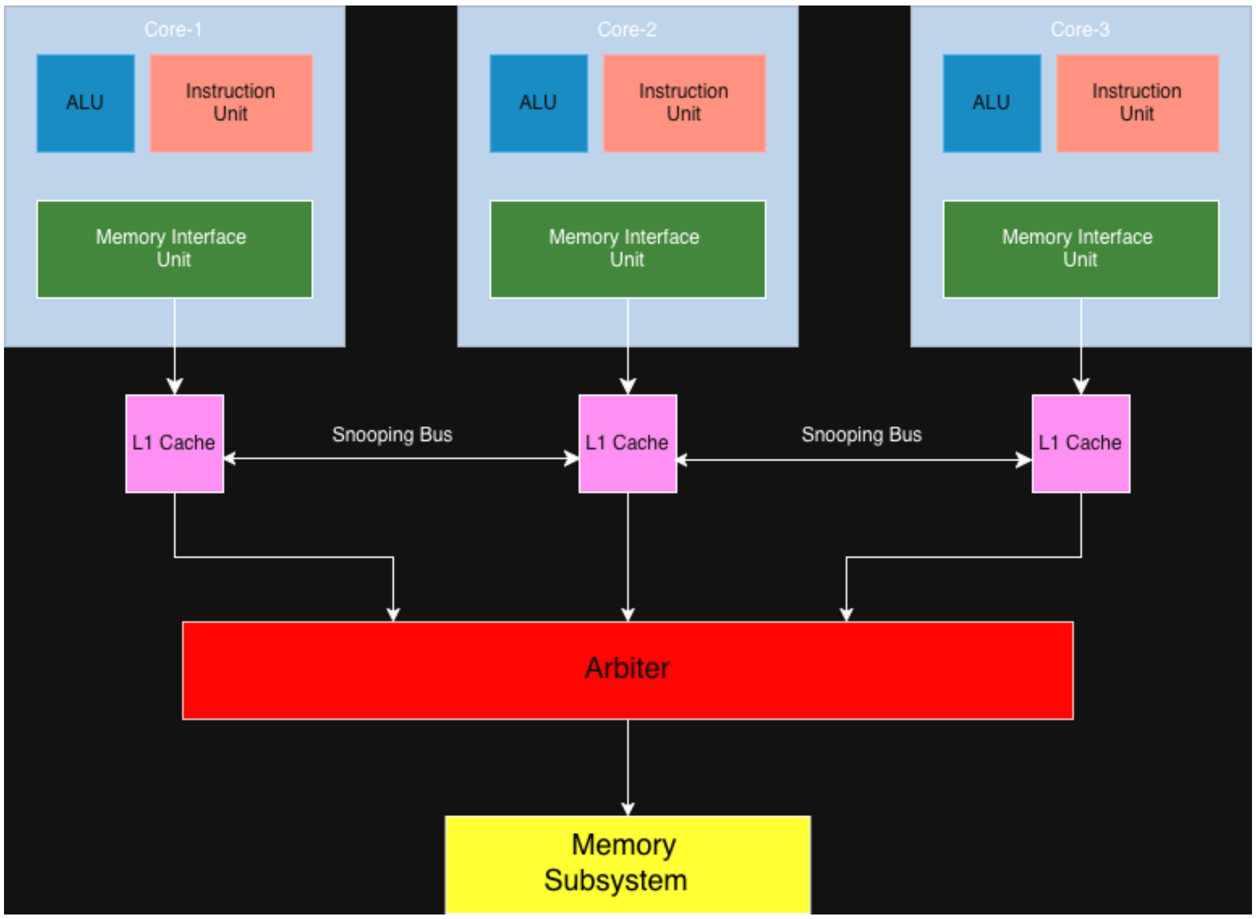
Design Signals	Instruction Unit & MIU
mem_req	Start load/store
mem_we	Load = 0, Store = 1
mem_addr[10:0]	Address to access
mem_read[7:0]	Load byte
mem_write[7:0]	Store byte
mem_done	Transaction complete

Design Signals	MIU & Cache
cache_req_valid	Request is valid
cache_req_ready	Cache is ready to accept request
cache_req_we	Load = 0, Store = 1
cache_req_addr[10:0]	Address to target
cache_req_write[7:0]	Store data byte
cache_resp_data[7:0]	Response data
cache_resp_valid	Response is valid, request completed

Design Signals	Coherence Snoop Bus
snoop_valid	Transaction is valid
snoop_core[1:0]	Core that initiated transaction
snoop_cmd	Read request = 0, RFO = 1
snoop_addr[10:0]	Target cache line base address

Design Signals	Cache & Arbiter & Main Memory
mem_req_valid	Memory request valid
mem_req_ready	Memory accepts (via arbiter)
mem_req_we	Read = 0, Write = 1
mem_req_addr[10:0]	Address to target
mem_req_write[7:0]	Store data byte
mem_resp_data[7:0]	Read response data byte
mem_resp_valid	Response is valid, request completed

Block Diagram



References/Citations
Computer Architecture 5th edition, Hennessy & Patterson, Chapter 5
Cache Coherence - https://en.wikipedia.org/wiki/Cache_coherence
MESI Protocol - https://en.wikipedia.org/wiki/MESI_protocol
Directory-based Coherence - https://en.wikipedia.org/wiki/Directory-based_coherence
Bus Snooping - https://en.wikipedia.org/wiki/Bus_snooping