University of Thessaly
Department of Electrical and Computer Engineering

RISC -V VECTOR EXTENSION (RVV)

Presented by: Charalampos Zachariadis,
Filippos Markovitsis,
Stefanos Ziakas,
Eleni Athanailidi

ECE338 - Parallel Computer Architecture

CONTENTS

PROJECT AIM

- Explore **vector instructions** for **RISC-V** architectures.
- Implement hardware to execute custom instructions using **Verilog**.
- **Validate** our implementation.

Ultimate Goal: 🎯

Integrate our designed hardware to an existing RISC-V implementation used currently only for scalar data.

TERMINOLOGY

Vector Register	A register that holds multiple elements with varying size (e.g., 8-bit, 16-bit, 32-bit, 64-bit)
VLEN	The length of the vector registers in bits
VL	Determines the number of elements a vector instruction will operate on during a single execution.

TERMINOLOGY

AVL	Indicates the vector length (VL) the program wants to use (vl <= avl <= 2 vlmax)
SEW (Standard Element Width)	Specifies the width of individual elements in the vector register (e.g., 8-bit, 16-bit, 32-bit, 64-bit)
LMUL (Length Multiplier)	Determines the grouping of vector registers to allow different vector lengths

DESIGN ACCEPTANCES

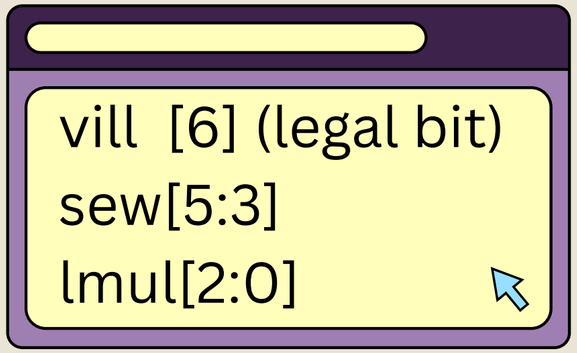
Minimal usage of **CSR** registers

Use only the CSR registers needed to our instrunction set (vtype, VL, AVL)

DESIGN ACCEPTANCES

Minimal usage of **CSR** registers

Use only the CSR registers
needed to our instrunction set
(vtype, VL, AVL)



DESIGN ACCEPTANCES

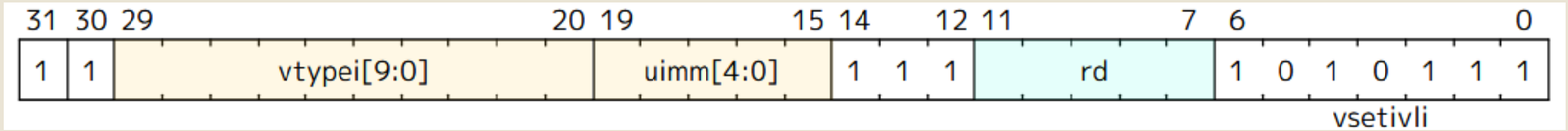
Minimal usage of **CSR** registers

The are accesed only in the vsetivli instrunction (**no need for addresses**)

DESIGN ACCEPTANCES

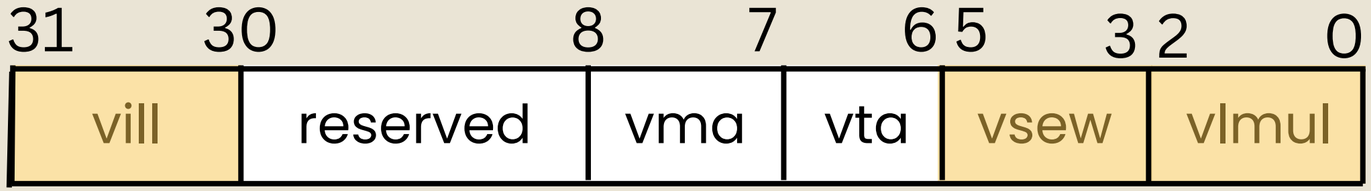
- vsetvli, vsetvl instrunction not supported

- AVL is set from the uimm from vsetivli (5 bits)



DESIGN ACCEPTANCES

- truncated vtype register



DESIGN ACCEPTANCES

- Masking Exclusively Done with zeros

SEW = 16 SEW = 16 SEW = 16 SEW = 16

LMUL = 1, VL = 3

DESIGN ACCEPTANCES

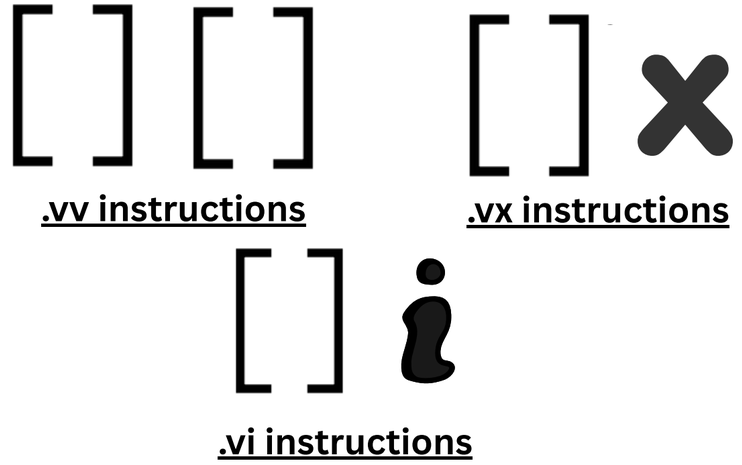
- Masking Exclusively Done with zeros

SEW = 16	SEW = 16	SEW = 16	SEW = 16
rvs	rvs	rvs	0000

LMUL = 1, VL = 3

INSTRUCTIONS

- **add, sub, mul, and, or, xor :** Arithmetic and Logical vv, vx, vi operations
- **vsetivli (rd, uimm, vtype):** Sets the vector length (VL) and vector type (vtype) based on an immediate AVL (Application Vector Length) and vtype.
- (premature) **vsb.v v3, imm(s1):** Stores single bytes in memory from vRegs



.vv instructions

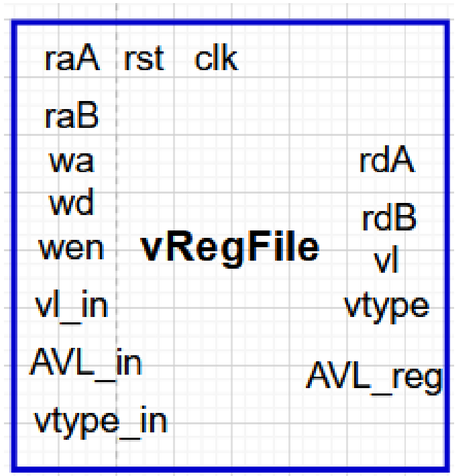
.vx instructions

.vi instructions

HARDWARE IMPLEMENTATION

Vector Register File

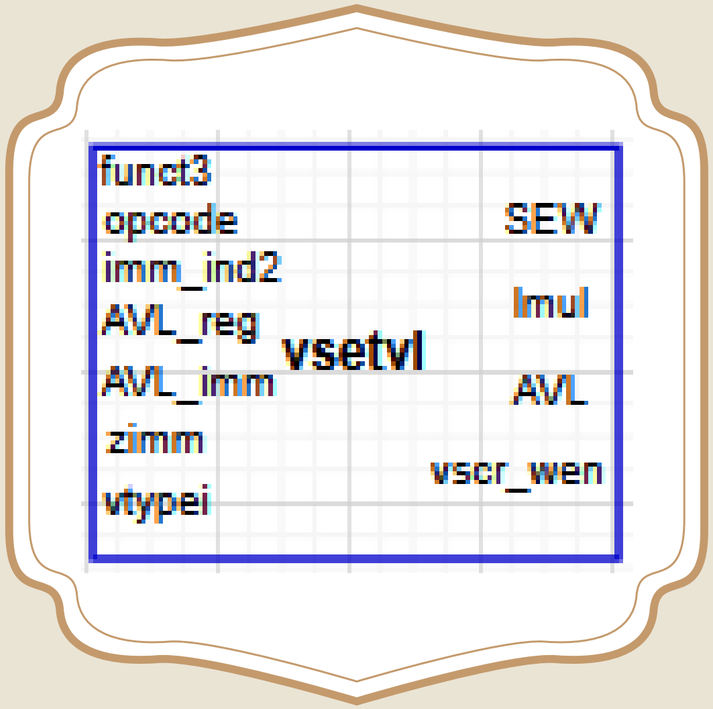
- 32 64-bit Registers
- vtype, vl, avl CSR logic



HARDWARE IMPLEMENTATION

vsetvl

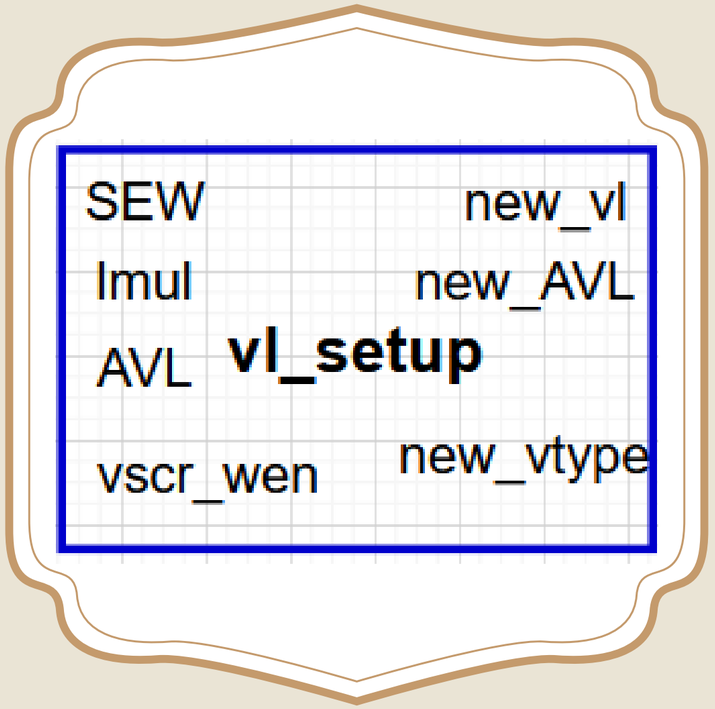
- Extracts vector config parameters: SEW, LMUL, and AVL from setup instructions
- Activates only when opcode = VR FORMAT and funct3 = VC FORMAT
- Asserts vcsr_wen to enable updates to vector control registers (vl, vtype)
- Outputs defaults and blocks changes if instruction is not a valid config command



HARDWARE IMPLEMENTATION

vl_setup

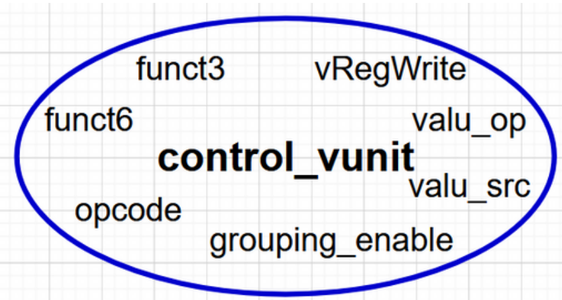
- Determines vector length (VL) based on SEW, LMUL, and requested AVL
- Calculates vlmax using current configuration and limits VL to min(AVL, vlmax)
- Validates SEW and LMUL; vector config only allowed if vcsr_wen == 1
- Disables vector execution if config is invalid or not permitted



HARDWARE IMPLEMENTATION

control_vunit

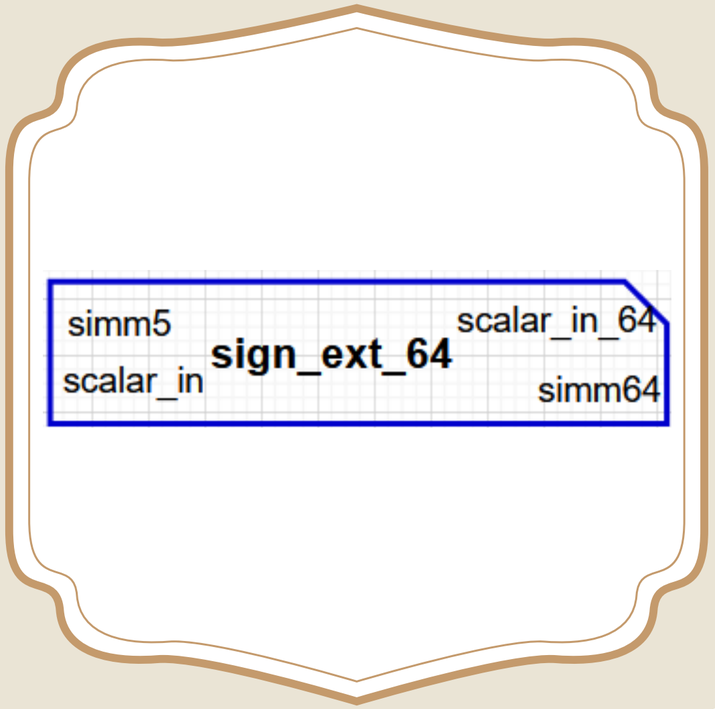
- Decodes vector instructions and generates control signals for the vector ALU (vALU)
- Uses opcode, funct3, and funct6 to determine operation type and control behavior
- Sets signals for operand source, write enable, operation type, and grouping
- Disables vector operations if instruction is not recognized or not in vector format



HARDWARE IMPLEMENTATION

sign_ext_64

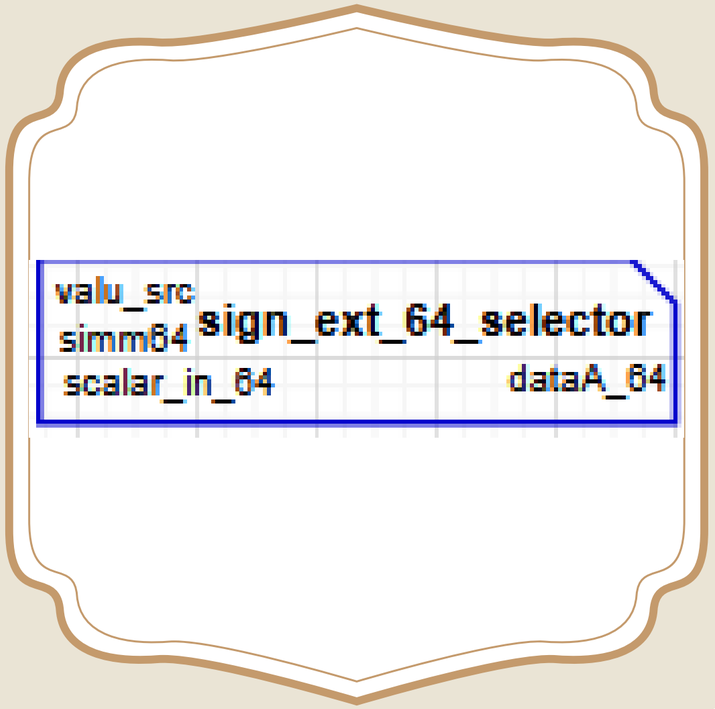
- Extends 5-bit and 32-bit signed inputs to 64-bit values
- simm5: Sign bit replicated 59 times to form simm64
- scalar in: Sign bit replicated 32 times to form scalar_in_64
- Ensures correct signed interpretation for vector operations



HARDWARE IMPLEMENTATION

sign_ext_64_selector

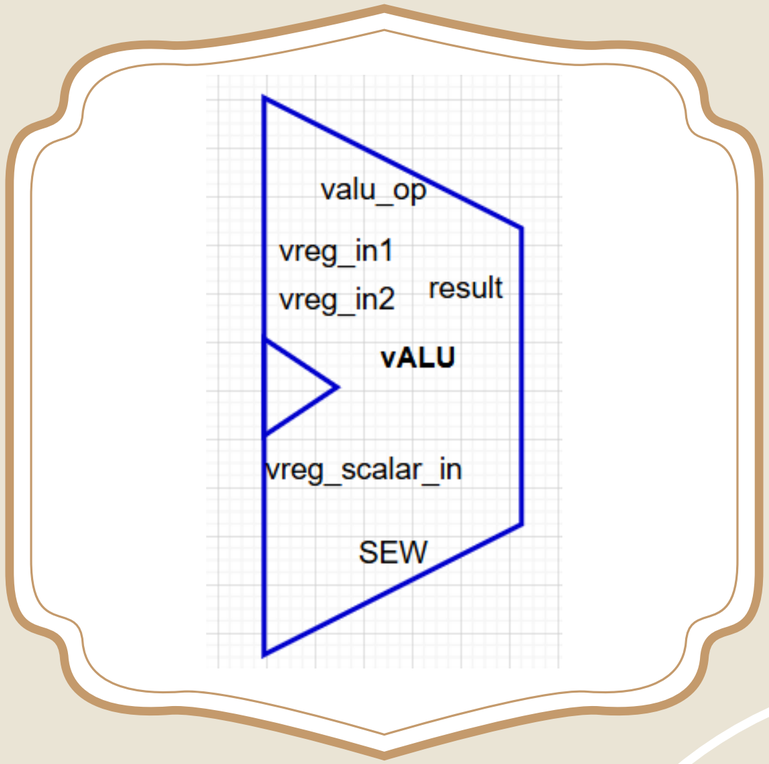
- Selects first vALU operand for vector-scalar operations
- Chooses between scalar_in_64 and simm64, based on valu_src control signal
  - valu_src = 0: select scalar input
  - valu_src = 1: select immediate
- Supports flexible operand handling for different instruction types



HARDWARE IMPLEMENTATION

vALU

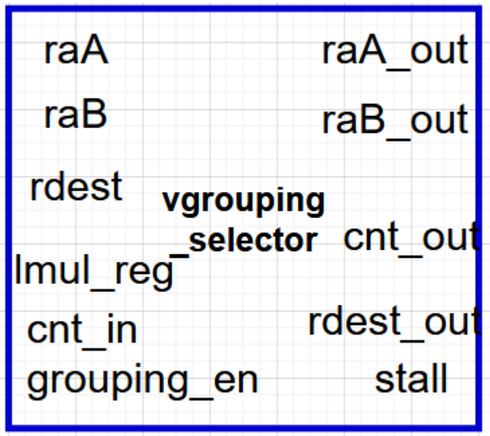
- executes the logical and arithmetic vector operations
- The operations rely on the SEW
- It executes executes all the instrunctions between single vector registers within one cycle



HARDWARE IMPLEMENTATION

vgrouping_selector

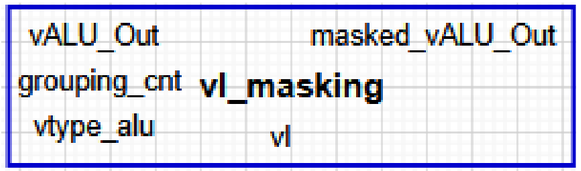
- Handles register grouping when LMUL > 1 by splitting operations across multiple physical registers
- Uses a counter to track sub-iterations and update register indices (raA, raB, rdest) accordingly
- Asserts a stall while grouped operations are in progress to prevent premature instruction fetch
- Counter and Stall logic



HARDWARE IMPLEMENTATION

vl_masking

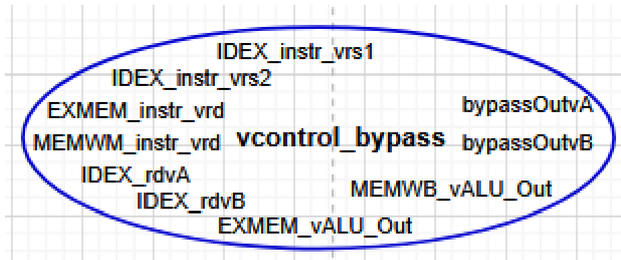
- Applies Tail Agnostic Zero Masking to elements beyond given vl
- Provides a clean separation of useless and useful bits inside the vector register
- Dynamic, based on given VL



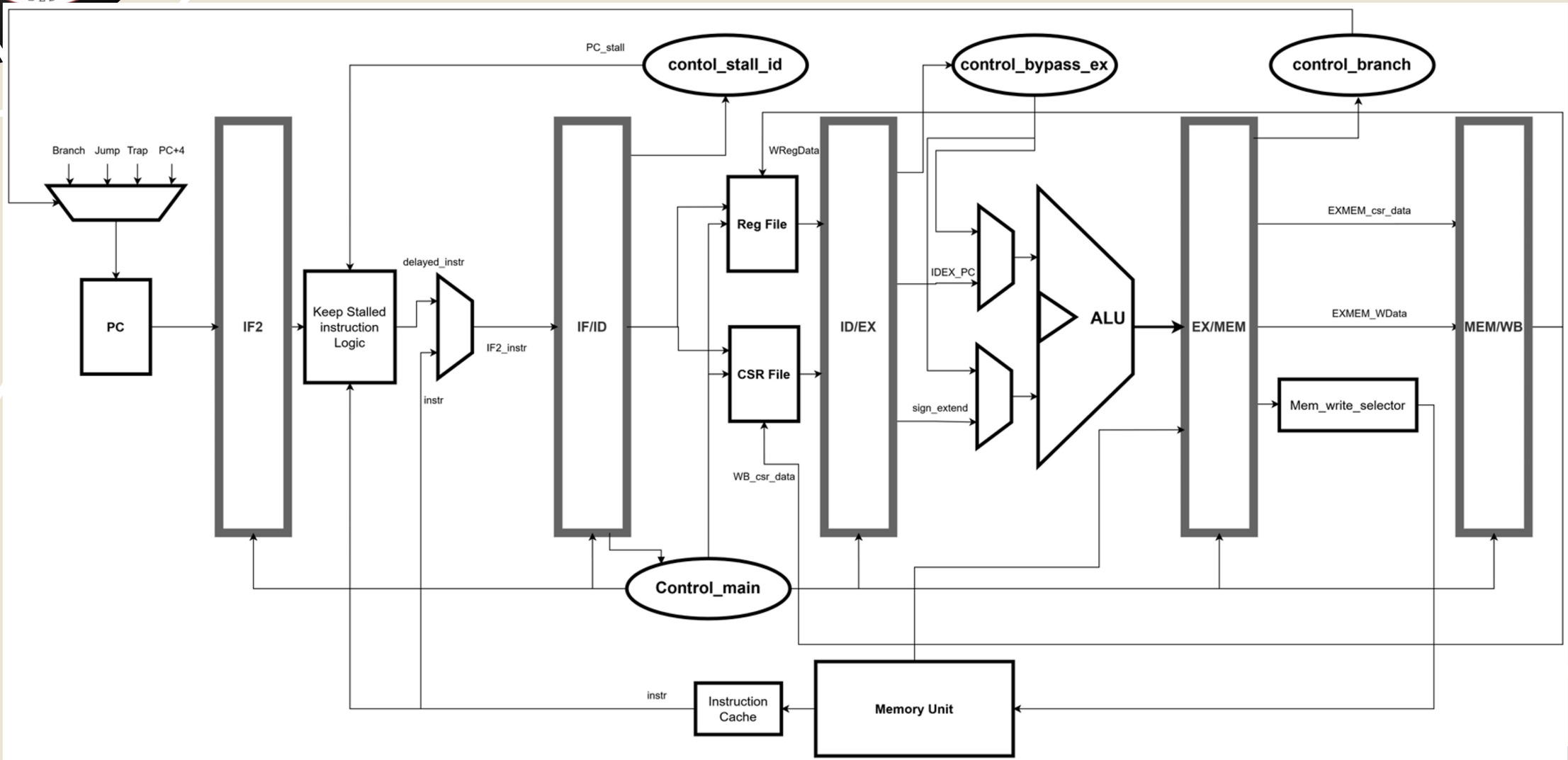
HARDWARE IMPLEMENTATION

vcontrol_bypass

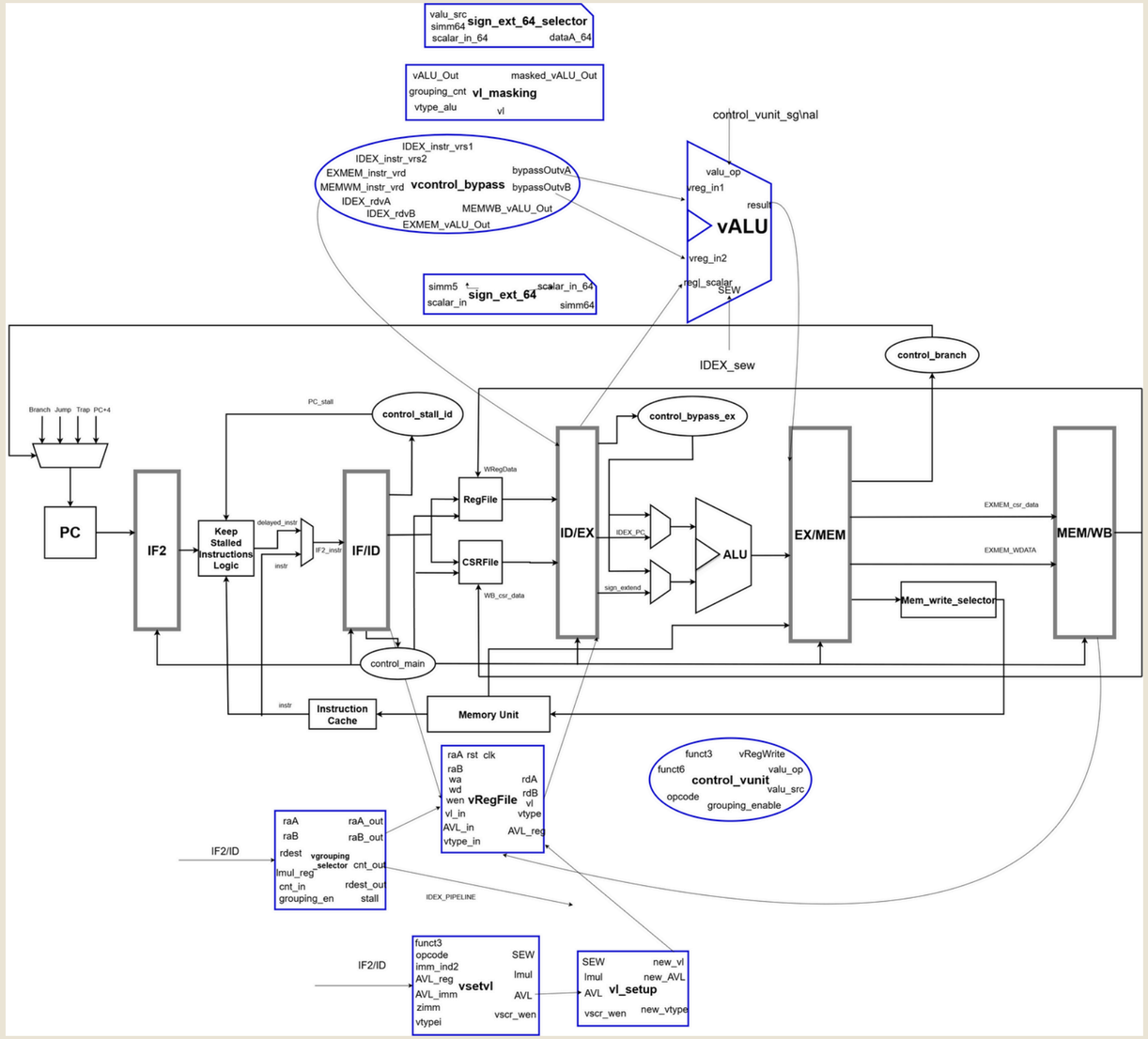
- Resolves data hazards in the vector pipeline using data forwarding logic
- Monitors register dependencies between pipeline stages (IDEX, EXMEM, MEMWB)
- Forwards latest ALU results to avoid stalls when source and destination registers match
- Uses the scalar RISC-V bypass unit to handle dependencies involving the base RegFile



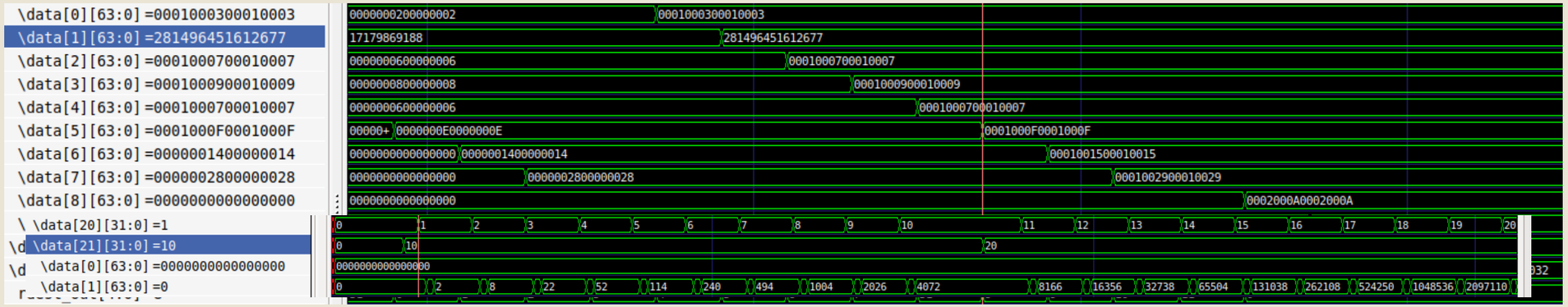
INTEGRATION



INTEGRATION



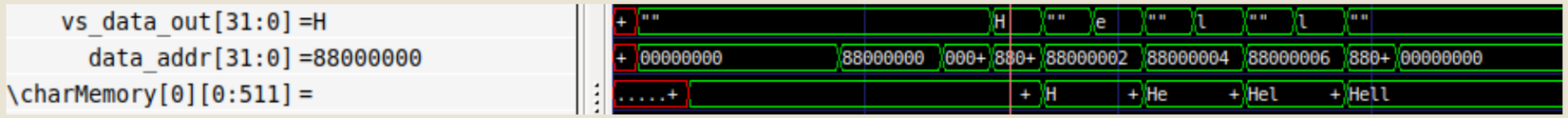
TESTING & VALIDATION



Testing AVL, LMUL, SEW & Read-After-Write Dependencies

Infinite Loop Combination of regular regs with vectors

TESTING & VALIDATION



Vector Store in memory

FUTURE WORK

- VLUL < 1 (e.g. 1/2, 1/4, 1/8)
- vsetvl and vsetvli implementation
- Support loads from memory and more efficient stores in it
- More instructions supported → complete set of CSR registers in order to adjust their functionality

VALU

EVALUATION

Instructions Exploration ★★★★★

Vector Architecture ★★★★

Testing & Validation ★★★★

Zedboard Execution

University of Thessaly
Department of Electrical and Computer Engineering

THANK YOU

Presented by: Charalampos Zachariadis,
Filippos Markovitsis,
Stefanos Ziakas,
Eleni Athanailidi