CORE-V-WALLY RISC-V Processor Project Concept and Launch

David Harris, James Stine, Ross Thompson and Sarah Harris
Harvey Mudd College, Oklahoma State University, University of Nevada Las Vegas

OpenHW Group Technical Working Group Meeting
23 January 2023

Overview

- Introduction
- Background: Wally Overview
- Implementation and Verification
- Team & OpenHW repository
- Conclusion

Project Concept and Launch

- We have put information on Project Concept and Project Launch in the documentation GitHub site for the OpenHW Group
  - https://github.com/openhwgroup/programs/tree/master/Project-Descriptions-and-Plans/CORE-V-WALLY
- This presentation is designed to give details of these proposals and integration into the OpenHW group.
- It is designed as CORE-V-WALLY and is an open-source configurable RISC-V microprocessor and System-on-Chip (SoC) project including SystemVerilog files, test suites, benchmarking, peripherals, Linux boot, and a design flow for an implementation on FPGA boards and for implementations as a SoC targeting 28nm.

Importance

This project is important in that it is associated with a textbook and that it is highly configurable.

No existing SystemVerilog RISC-V cores have either of these attributes.

In the maximum configuration, it meets all application processor requirements.

The SystemVerilog code is written considering performance and readability.

The floating-point and muldiv units target two cycle latency.

OpenHW Collaboration Goals

Increase Wally's visibility and credibility

Improved test suites

Feedback from collaborators

Become the primary RISC-V processor used in upper division / intro grad courses in higher education

Industry adoption

- Presently collaborating with Imperas for test development

Goals

RISC-V System-on-Chip Design textbook

- David Harris, James Stine, Ross Thompson, Sarah Harris
- To be published by Elsevier 2024
- Follow-on to the *Digital Design and Computer Architecture RISC-V Ed*.

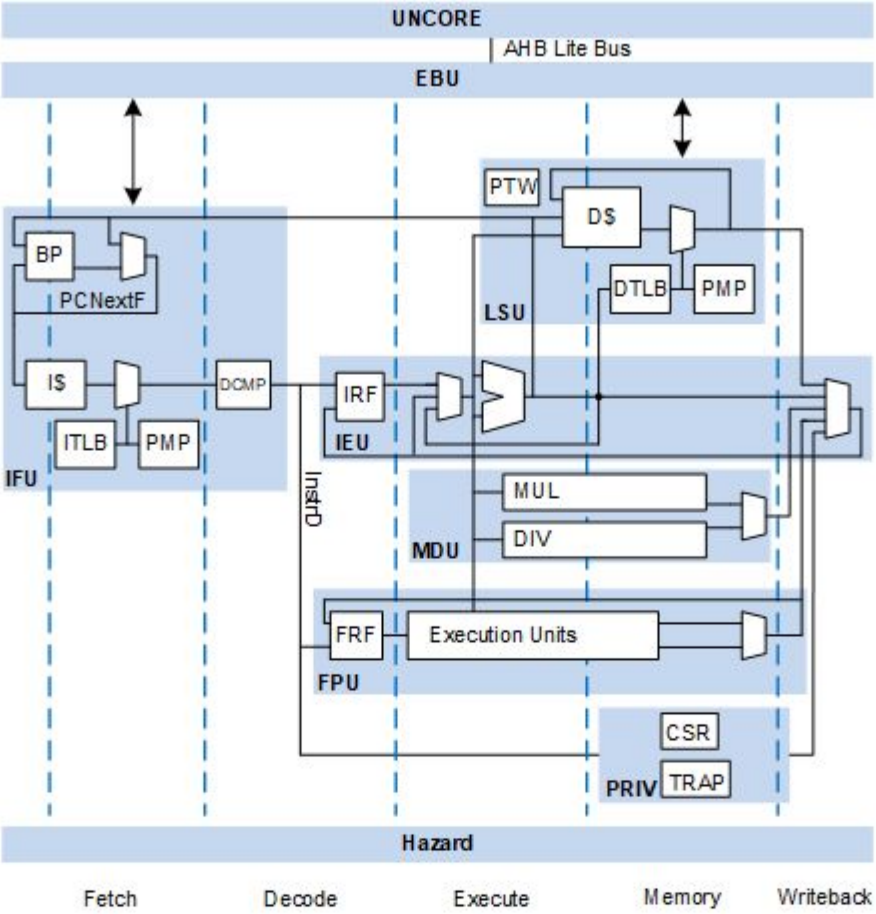
Upper undergraduate / intro graduate-level textbook

- Focus on design issues implementing a complete processor
- Pedagogy for those interested in detailed computer architecture details.
- Complete repository for simulation, benchmarks, HDL, and software.

Wally Overview

Open-source configurable RISC-V processor

- Single-issue 5-stage pipeline
- Configurable System-Verilog supporting standard extensions
- Simulation: Siemens Questa
- Synthesis with SNPS Design Compiler and Xilinx FPGA tools



Standard Configurations

Configuration	Config	XLEN	DTIM / IROM Size	Bus	Periph	I$/D$ Size	Privilege Modes	Virt Mem
Embedded	rv32e	32 bits	n/a	YES	NO	n/a	none	NO
Simple CPU	rv32i	32 bits	2K / 2K	NO	NO	n/a	none	NO
Microcontroller	rv32ic	32 bits	4K / 16K	YES	YES	n/a	MU	NO
Apps Proc	rv32gc	32 bits	n/a	YES	YES	16K	MSU	YES
Simple CPU	rv64i	64 bits	2K / 2K	NO	NO	n/a	none	NO
Apps Proc	rv64gc	64 bits	n/a	YES	YES	16K	MSU	YES

Features

- RV32 and RV64
- Optional Extensions:
  - A: Atomic
  - C: Compressed
  - M: Multiply/Divide
  - E: Embedded
  - F/D/Q: Single/Double/Quad Floating-Point
  - S/U: Supervisor / User Mode
  - Zicsr: Control/Status Registers
  - Zfencei: Instruction synchronization
  - Counters
  - Virtual Memory
  - Physical Memory Protection
- Optional Microarchitectural Features
  - L1 I$ and D$
  - GSHARE branch predictor
  - Vectored interrupts
- Optional Peripherals
  - CLINT, PLIC, UART, GPIO, TIM

Parameter	Meaning	rv32e	rv32ic	rv32gc	rv64ic	rv64gc
XLEN	Architecture width	32	32	32	64	64
MISA	Instruction set	E	IC	IMAFDC	IC	IMAFDC
ZICSR_SUPPORTED	CSRs supported	0	1	1	1	1
ZIFENCEI_SUPPORTED	FENCE.I supported	0	0	1	0	1
ZICOUNTERS_SUPPORTED	Performance counters	0	0	1	0	1
COUNTERS	# of counters	n/a	n/a	32	n/a	32
VECTORED_INTERRUPTS_SUPPORTED	Vectored Interrupts	0	1	1	1	1
DMEM	Data Memory Type	BUS	TIM	CACHE	TIM	CACHE
IMEM	Instruction Memory Type	BUS	TIM	CACHE	TIM	CACHE
VIRTMEM_SUPPORTED	Supports virtual memory	0	0	1	0	1
ITLB_ENTRIES	Instruction TLB size	n/a	n/a	32	n/a	32
DTLB_ENTRIES	Data TLB size	n/a	n/a	32	n/a	32
DCACHE_NUMWAYS		n/a	n/a	4	n/a	4
DCACHE_WAYSIZEINBYTES		n/a	n/a	4096	n/a	4096
DCACHE_LINELENINBITS		n/a	n/a	256	n/a	256
ICACHE_NUMWAYS		n/a	n/a	4	n/a	4
ICACHE_WAYSIZEINBYTES		n/a	n/a	4096	n/a	4096
ICACHE_LINELENINBITS		n/a	n/a	256	n/a	256
PMP_ENTRIES	Phys Mem Protection: 0, 16, or 64	0	0	64	0	64
DIV_BITSPERCYCLE		n/a	n/a	4	n/a	4

Implementation: Synthesizable SystemVerilog

Wally illustrates complex design in SystemVerilog

Configuration options to optionally support each feature

Uses industry-standard EDA design tools

No abstraction gap between coding and debugging

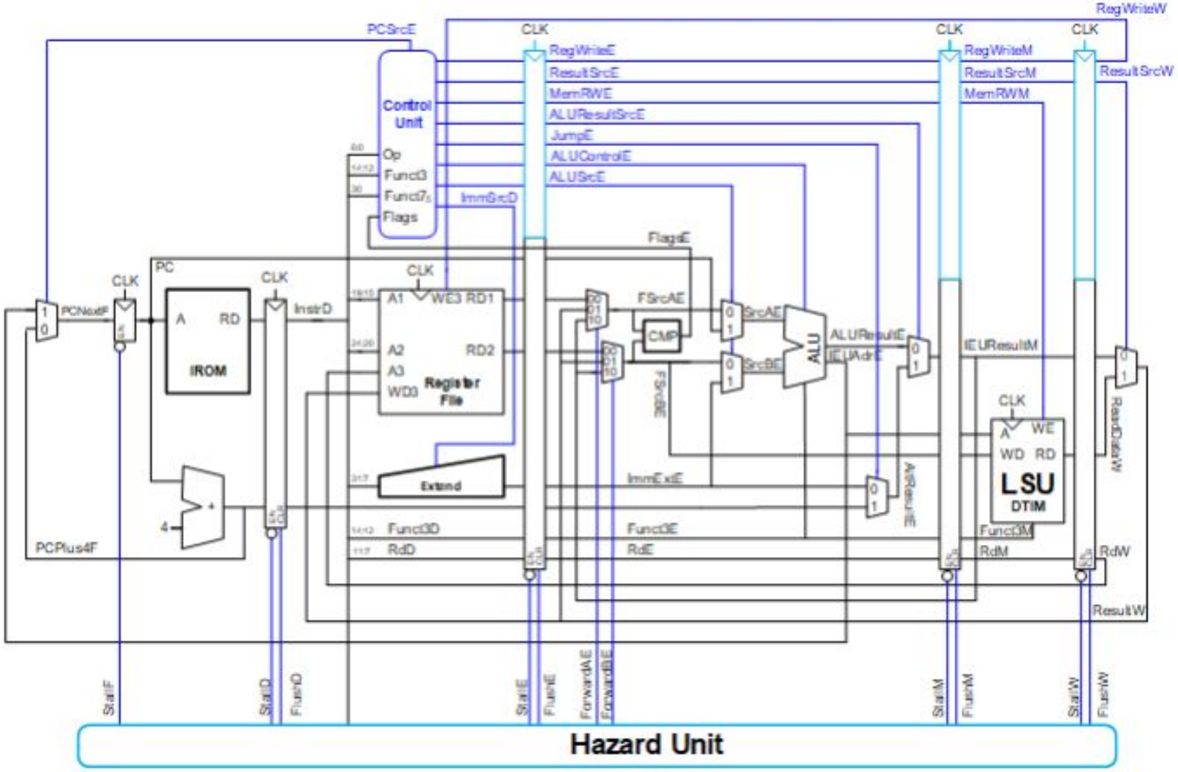
Efficient synthesis

```
if (`XLEN==32) begin:shifter // RV32
  always_comb  // funnel mux
    if (Right)
      if (Arith) z = {{31{A[31]}}, A};
      else       z = {31'b0, A};
    else         z = {A, 31'b0};
  assign amttrunc = Amt; // shift amount
end else begin:shifter  // RV64
  always_comb  // funnel mux
    if (W64) begin // 32-bit shifts
      if (Right)
        if (Arith) z = {64'b0, {31{A[31]}}, A[31:0]};
        else       z = {95'b0, A[31:0]};
      else         z = {32'b0, A[31:0], 63'b0};
    end else begin
      if (Right)
        if (Arith) z = {{63{A[63]}}, A};
        else       z = {63'b0, A};
      else         z = {A, 63'b0};
    end
  assign amttrunc = W64 ? {1'b0, Amt[4:0]} : Amt; // 32 or 64-bit shift
end
```

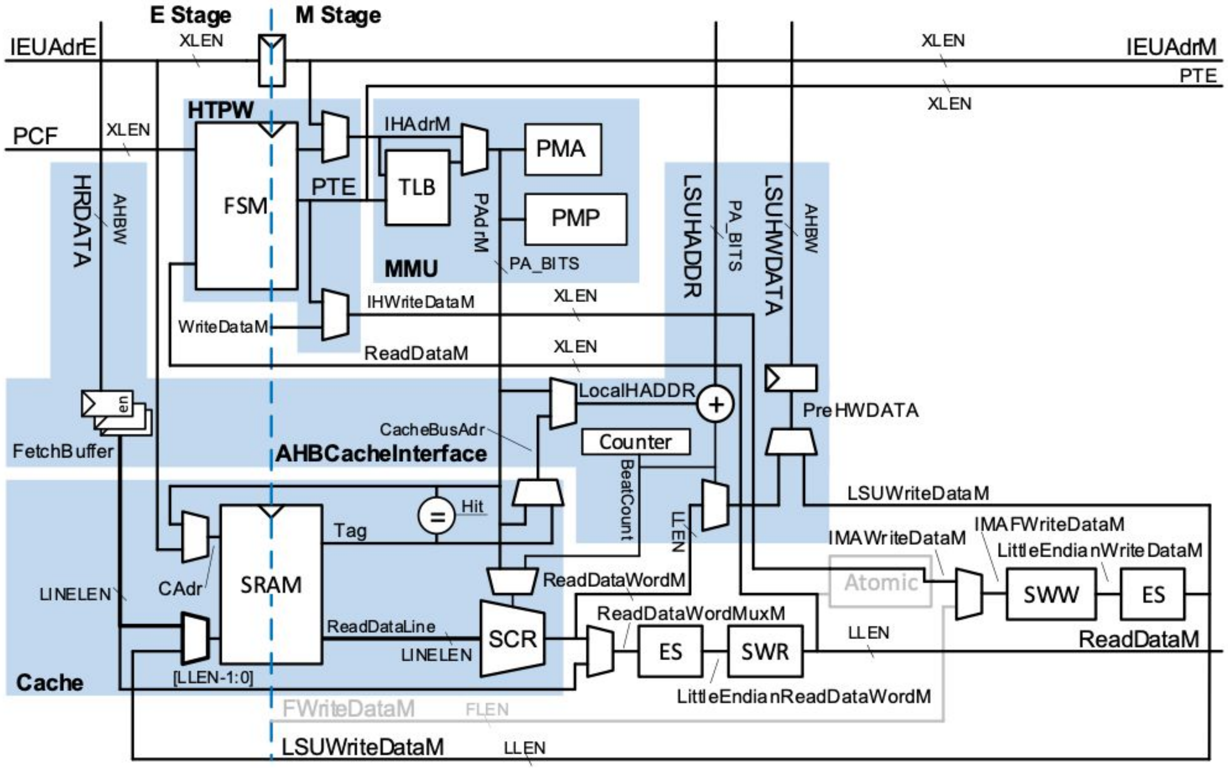
RV32I/RV64I Pipeline

Forwarding

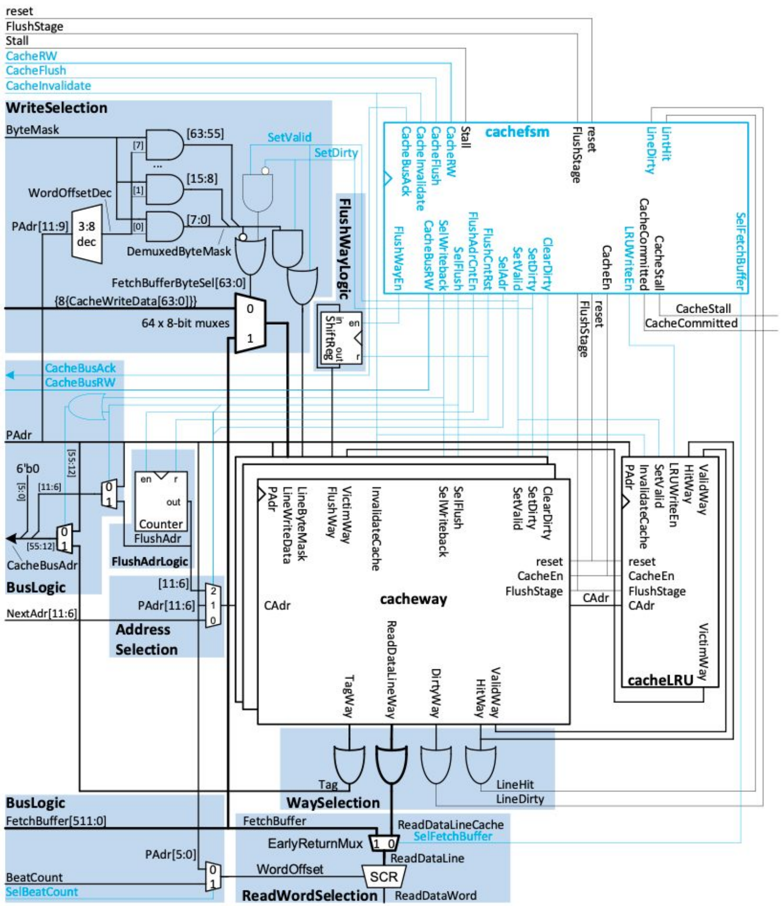
Two-cycle load-use latency



Load/Store Unit

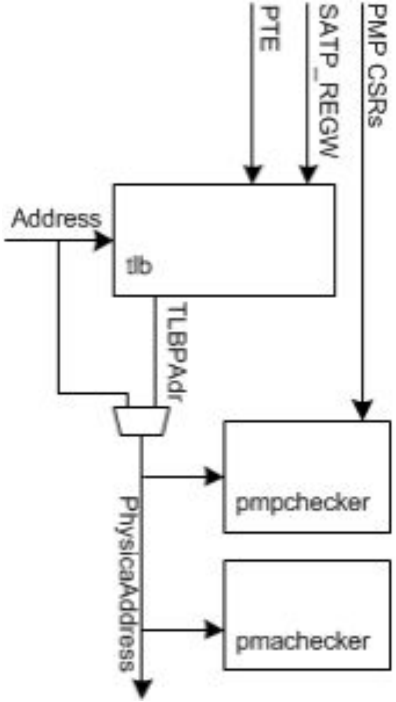


Cache

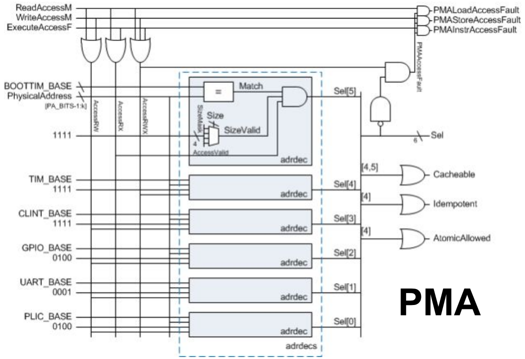


reset
FlushStage
Stall
CacheRW
CacheFlush
CacheInvalidate
WriteSelection
ByteMask
WordOffsetDec
PAdr[11:9]
3:8 dec
DemuxedByteMask
FetchBufferByteSel[63:0]
{8{CacheWriteData[63:0]}}
64 x 8-bit muxes
SetValid
SetDirty
FlushWayLogic
ShiftReg
cachefsm
CacheStall
CacheCommitted
CacheBusAck
CacheBusRW
PAdr
6'b0
Counter
FlushAdr
FlushAdrLogic
CacheBusAdr
BusLogic
NextAdr[11:6]
PAdr[11:6]
Address Selection
cacheway
CAdr
cacheLRU
Tag
LineHit
LineDirty
WaySelection
BusLogic
FetchBuffer[511:0]
FetchBuffer
ReadDataLineCache
SelFetchBuffer
EarlyReturnMux
PAdr[5:0]
WordOffset
ReadDataLine
SCR
BeatCount
SelBeatCount
ReadWordSelection
ReadDataWord

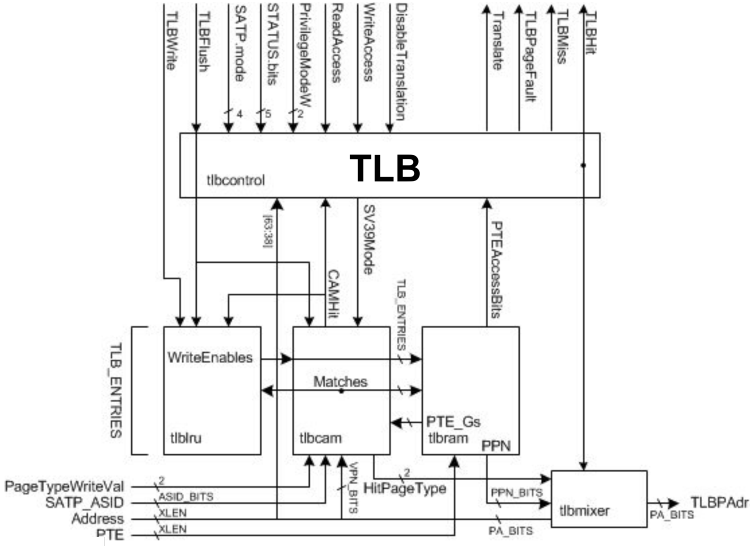
Memory Management Unit



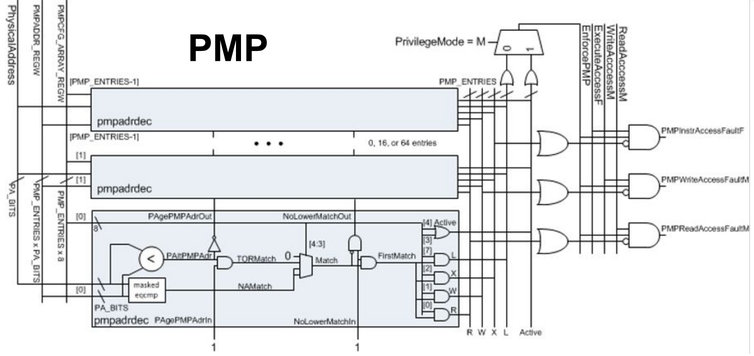
MMU



PMA



TLB



PMP

FPU

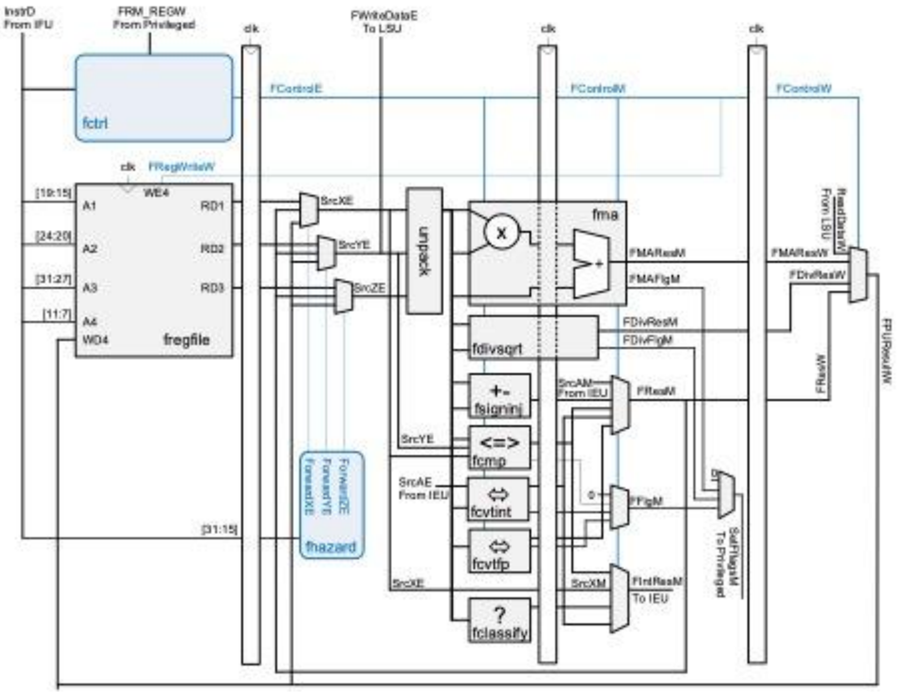
Configurable half/single/double/quad

FMA for add, sub, mul, fma variants

Recurrence division and square root

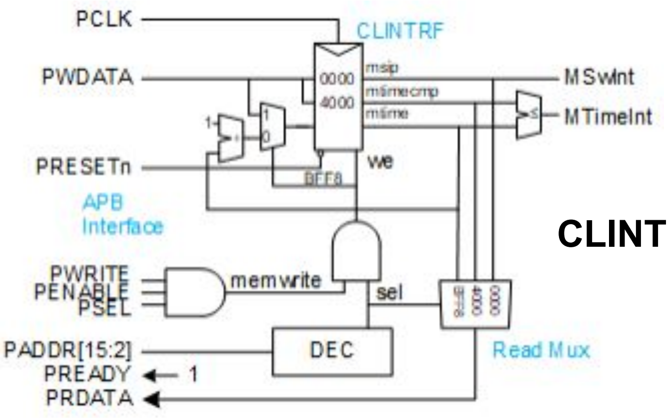
- Configurable Radix 2/Radix 4
- Early termination
- Handles denorms & special cases

Shared postprocessing: norm/rnd/flags

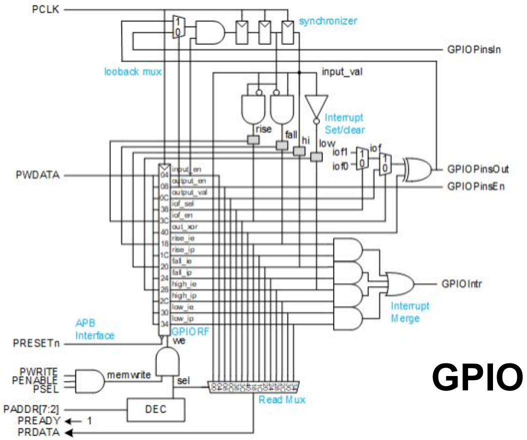


Peripherals (AHB with APB bridge)

CLINT:	Core-Local Interruptor	Timer & Software Interrupts
PLIC:	Platform-Level Interrupt Controller	External Interrupt Routing
GPIO:	General-Purpose I/O	SiFive Compatible
UART:	Serial Port	PC16550D-Compatible



CLINT



GPIO

Verification

riscv-arch-test	Architecture Test Suite
UCB TestFloat	FP-specific tests
wally-riscv-arch-test	Custom privileged and peripheral tests

Benchmarking

CoreMark: 2.5 CoreMarks/MHz (1.16 CPI)

- RV64GC with caches, branch predictor
- Note that this compilation uses 315k instructions/iteration with all flags
- Western Digital claims 270k instructions/iteration - still investigating

Embench 1.0

- Speed-optimized: 1.05
  - 5% faster per MHz than a Cortex M4
- Size-Optimized: 1.03
  - 3% larger program than ARM V7M
- Some size and speed anomalies relative to ARM

The Team

Prof. David Harris, Harvey Mudd College

- Processor design experience at Intel, HP, Sun, Broadcom
- Author of *CMOS VLSI Design*, *Digital Design & Computer Architecture*, *Logical Effort*, *Skew-Tolerant Circuit Design*

Prof. James Stine, Oklahoma State University

- Designed several IC designs every year for last 20+ years + tools/libraries for Google Skywater Technology/EDA Companies/SRC/GF/MOSIS

Ross Thompson, Oklahoma State University

- Working on CPU IC designs at Air Force Research Laboratory (AFRL) and AMD
- Lead architect of several architectures including secure hardware architectures

Prof. Sarah Harris, UNLV

- Author of *Digital Design & Computer Architecture* (including RISC-V Edition)

OSU Students: Jacob Pease (FPGA), Juliette Reeder (K extension), Sivan Auerbach (Branch prediction strategies)

HMC Students: Noah Boorstin (Core), Kaveh Pezeshki (Linux), Ben Bracker (Linux and Peripherals), Skylar Litz (Linux), Alessandro Maiuolo (FPU), Cedar Turek (FPU), Daniel Torres (Benchmarking), Kip Macsai-Goren (Privileged Tests), Madeleine Masser-Frye (Synthesis)

CMU Students: Katherine Parry (FPU)

riscv-arch-test and Embench Working Groups

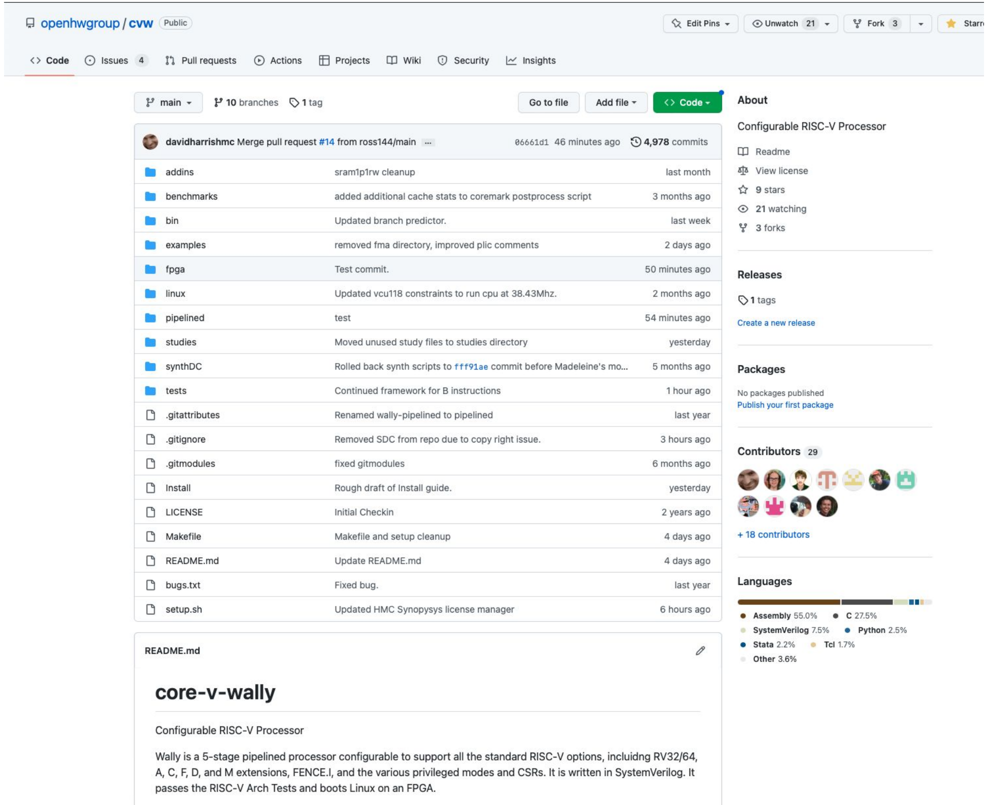
Two years of class trials at HMC and OSU

OpenHW Repository

Moved to OpenHW GitHub site

https://github.com/openhwgroup/cvw

http://bit.ly/3QXNmRa



Technical Readiness Level

Presently at TRL3

- Passes riscv-arch-test, SoftFloat, and custom tests in Questa
- Boots Linux on FPGA board on Xilinx VCU108 board

Desire to advance to TRL5

- Continue to refine and prettify RTL while completing textbook
- Robust methods to install required tools (e.g. riscof, SAIL, Embench)
- Broader user & class testing
- Collaborate with the OpenHW Group on RTL test suites
- Silicon prototype
- Partner with potential commercial users

Textbook with Elsevier : RISC-V System on Chip Design

Ties together principles, RISC-V implementation, and verification in each chapter

1) Introduction	7) Caches	13) Floating Point
2) Tool Flow	8) MMU	14) Atomic
3) HDL Design Practices	9) Load/Store Unit	15) Peripherals
4) Pipelined Core	10) Instruction Fetch Unit	16) Benchmarking
5) Privileged Ops	11) Compressed	17) Linux
6) Bus Interface	12) MulDiv	18) Implementation

Conclusion

Grateful for the OpenHW Group and the opportunity to present a new architecture for education and research.

> Big thank you to Rick O’Connor, Duncan Bees, Mike Thompson and other OpenHW staff for all their help!
>
> Thank you to Imperas for help with verification suites.

Configurable for different RV extensions and a has full repository to help students and those interested in microarchitecture learn challenging topics.

Supplemental material to be included with textbook including possible laboratories.

Both commercial and academic collaborations are appreciated.