GROW 2025

First Global RISC-V Open Workshop

June 30 - July 2, 2025, University of São Paulo, São Paulo

RS5: Bridging Scalability and Efficiency in Modern Processor Design

Fernando Gehm Moraes

July 1, 11h00, 2025

GAPH-PUCRS

Agenda

1. RS5 RISC-V

2. MEMPHIS Manycore

3. SoC-Wimed

RISC-V RS5

RISC-V developed at PUCRS

- **No reference design (e.g., PULP, IBEX, CV32E40P, …)**
- **No 3PIP**
- **Flexiblity**
- **Support for OSs**
- **Open hardware - https://github.com/gaph-pucrs/RS5**

- 32-bit integer ISA
- Machine and User Privileges
- Standardized Interrupt Controller
- Real-Time Clock (RTC)

RISC-V RS5

RV32IMACZicsr_Zihpm_Xosvm ISA

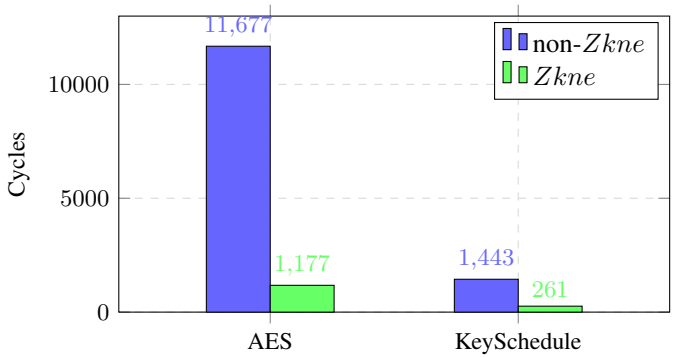
- I - Integer 32-bit
- M - Hardware multiplication and division
- A - Atomic instructions
- C - Compressed Instructions
- CSR with machine-mode and user-mode
- Hardware counters
- Custom offset-based virtual memory

Nunes, W. A., Dal Zotto, A. E., Borges, C. S., Moraes, F. G (2024). RS5: An Integrated Hardware and Software Ecosystem for RISC-V Embedded Systems. In Proceedings of the IEEE Latin America Symposium on Circuits and Systems (LASCAS), pages 1-5.

RISC-V RS5

Other optional extensions

- Zicond
- Zcb (code size reduction)
- **Zkne (AES)**
  - *aes32esmi* and *aes32esi*
  - modified TinyCrypt library
- **Vector subset** (VLEN from 64 bits up to 1024 bits)



<2% area overhead

RISC-V RS5

Xosvm extension (dedicated extension)

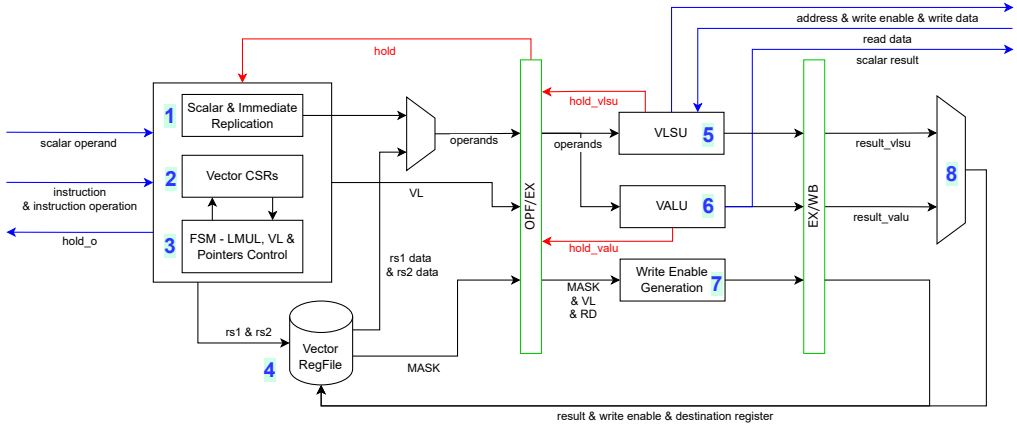
- "**osvm**": offset and size virtual memory
- **Paged** memory organization
  - divides the memory into contiguous, statically-sized pages for each process
- **Alternative** for:
  - **Sv32** extension requires the Supervisor mode (S-Mode)
    - Embedded Systems usually do not have S-Mode
  - **Physical Memory Protection (PMP)** extension
    - In Ibex PMP nearly doubles area footprint
- no need to add additional instructions - only R/W operations over CSRs

RISC-V RS5

Vector extension – subset, targeting ML applications

TABLE II
INSTRUCTIONS OF THE RS5 VECTOR UNIT AND THE CYCLES PER REGISTER FOR DIFFERENT SEW CONFIGURATIONS.

Instruction Class	Instruction Names	SEW=8 Cycles/Reg	SEW=16 Cycles/Reg	SEW=32 Cycles/Reg	Optional
Arithmetic	VADD, VSUB, VRSUB	1	1	1	N
Logic	VAND, VOR, VXOR	1	1	1	Y
Shifts	VSLL, VSRL, VSRA	1	1	1	N
Mask Compares	VMSEQ, VMSNE, VMSLTU, VMSLT, VMSLEU, VMSLE, VMSGTU, VMSGT	1	1	1	N
Min/Max	VMIN, VMINU, VMAX, VMAXU	1	1	1	Y
Multiplication	VMUL, VMULH, VMULHU, VMULHSU	1	1	3-4	N
Widening Multiplication	VWMUL, VWMULU, VWMULSU	2	2	4-5	N
Multiply and Accumulate	VMACC, VNMSAC, VMADD, VNMSUB	2	2	4-5	N
Division	VDIV, VDIVU, VREM, VREMU	1-8	1-16	1-32	Y
Sum Reduction	VREDSUM	1	1	1	Y
Min/Max Reduction	VREDMIN, VREDMINU, VREDMAX, VREDMAXU	1	1	1	Y
Logic Reduction	VREDAND, VREDOR, VREDXOR	1	1	1	Y
Register Moves	VMV, VMVR, VMVSX, VMVXS	1	1	1	N
Unit-Strided Load/Store	VLE, VSE	~VLEN/32	~VLEN/32	~VLEN/32	N
Strided Load/Store	VSLE,VSSE	VLEN/8	VLEN/16	VLEN/32	N
Index load/Store	VLUXEI, VLOXEI, VSUXEI, VSOXEI	VLEN/8	VLEN/16	VLEN/32	N



Nunes, W. A., Santos, A.., Moraes, F. G (**2025**). Accelerating Machine Learning with RISC-V Vector Extension and Auto-Vectorization Techniques In: ISCAS, pages 1-5.

RISC-V RS5

Configurable in the top SystemVerilog file

- *march* in the gcc flags

Parameter	Description	Options
Environment	Environment type	ASIC, FPGA
MULEXT	Include Hardware Multiplication/Division extension	MUL_OFF, MUL_ZMMUL, MUL_M
AMOEXT	Include Atomic operation extension	AMO_OFF, AMO_ZALRSC, AMO_ZAAMO, AMO_A
COMPRESSED	Include Compressed extension	TRUE, FALSE
XOSVMEnable	Include XOSVM extension (MMU)	TRUE, FALSE
ZIHPMEnable	Include ZIHPM extension (Performance Monitors)	TRUE, FALSE
ZKNEEnable	Include ZKNE extension (AES Hardware acceleration)	TRUE, FALSE
BRANCHPRED	Include Branch prediction	TRUE, FALSE
VEnable	Include Vector extension	TRUE, FALSE
VLEN	Vector length in bits	64, 128, 256, ...

```
module RS5
    import RS5_pkg::*;
#(
`ifndef SYNTH
    parameter bit           DEBUG          = 1'b0,
    parameter string        DBG_REG_FILE   = "./debug/regBank.txt",
    parameter bit           PROFILING      = 1'b0,
    parameter string        PROFILING_FILE = "./debug/Report.txt",
`endif
    parameter environment_e Environment      = ASIC,
    parameter mul_e         MULEXT           = MUL_M,
    parameter atomic_e      AMOEXT           = AMO_A,
    parameter logic [31:0]  START_ADDR       = '0,
    parameter bit           COMPRESSED       = 1'b0,
    parameter bit           VEnable          = 1'b0,
    parameter int           VLEN             = 256,
    parameter bit           XOSVMEnable      = 1'b0,
    parameter bit           ZKNEEnable       = 1'b0,
    parameter bit           ZICONDEnable     = 1'b0,
    parameter bit           ZCBEnable        = 1'b0,
    parameter bit           HPMCOUNTEREnable = 1'b0,
    parameter bit           BRANCHPRED       = 1'b1,
    parameter bit           FORWARDING       = 1'b1
)
(
```

RISC-V RS5

RISCOF compliant

- The RISC-V Architectural Tests ensure that software written for a given RISC-V Profile will run on all implementations that comply with that profile
- These tests ensure that the implementation follows correctly the ISA

Report generated on 2025-06-23 20:20 GMT by riscof v

Environment

Riscof Version	1.25.3
Riscv-arch-test Version/Commit Id	3.9.1
DUT	RS5
Reference	sail c simulator
ISA	RV32IMACUZicond_Zicsr_Zihpm_Zca_Zcb_Zkne
User Spec Version	2.2
Privilege Spec Version	1.1

sail c simulator

RV32IMACUZicond_Zicsr_Zihpm_Zca_Zcb_Zkne

Yaml

Show all details / Hide all details

Name
/home/moraes/RS5/riscof/extensions_work/extensions_checked.yaml *(show details)*
/home/moraes/RS5/riscof/extensions_work/platform_checked.yaml *(show details)*

Please visit YAML specifications for more information.

Summary

☑ 112Passed, ☑ 0Failed

Results

Show all details / Hide all details

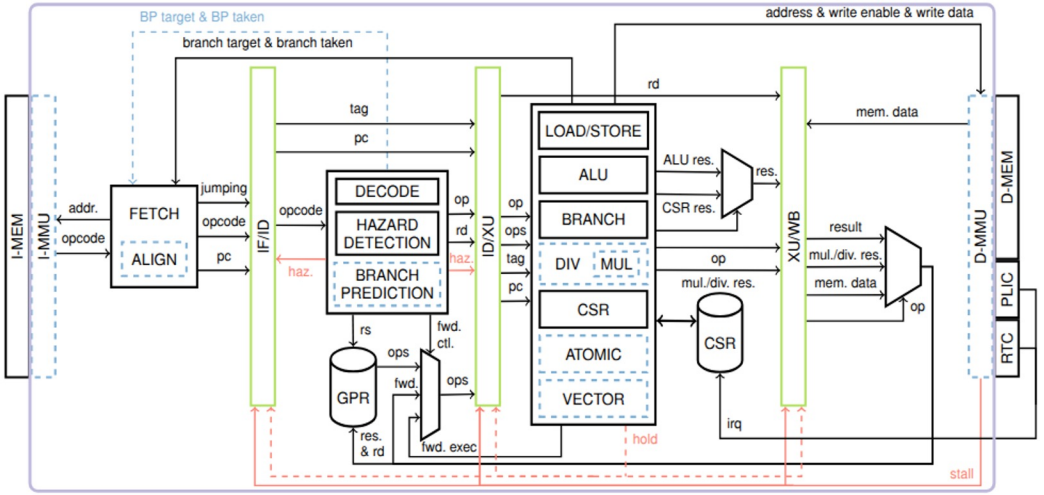
Test	Result	Pat
/home/moraes/RS5/riscof/riscv-arch-test/riscv-test-suite/rv32i_m/A/src/amoadd.w-01.S	Passed *(show details)*	/ho
/home/moraes/RS5/riscof/riscv-arch-test/riscv-test-suite/rv32i_m/A/src/amoand.w-01.S	Passed *(show details)*	/ho
	ed *(show details)*	/ho
	ed *(show details)*	/ho

https://riscof.readthedocs.io/en/stable/intro.html#intent-of-the-architectural-test-suite

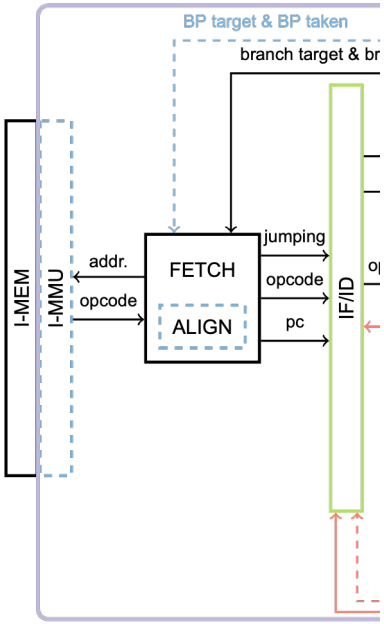
RS5 - implementation

4 stage pipeline

- Fetch (+ aligner and decompresser)
- Decode (+ branch prediction)
- Execute (+ div, mul, and atomic)
- Writeback



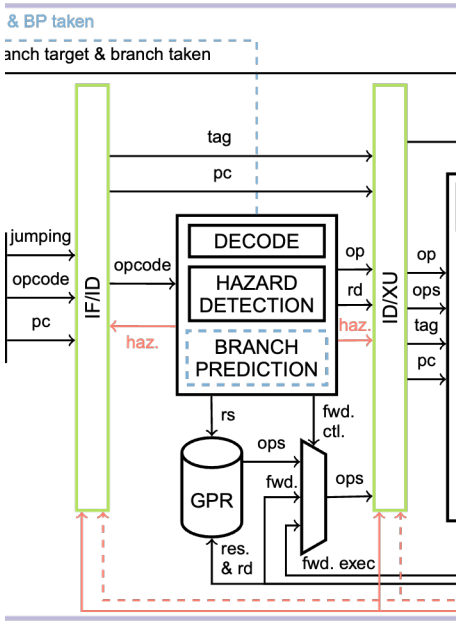
RS5 - Fetch (IF)



Fetch

- Controls the reception of instructions from the instruction memory (I-MEM)
- Selects the **Program Counter** (PC) based on the following events:
  - Reset
  - Trap occurrence
  - Trap return
  - Jumps
  - Sequential operation (default)
- Can operate with **virtual memory** addresses when connected to an I-MMU for translating virtual to physical addresses (**XOSVM** extension)
- With Compressed Extension: includes an **address aligner**
  - Contains an instruction decompressor that converts 16-bit instructions into 32-bit instructions for decoding

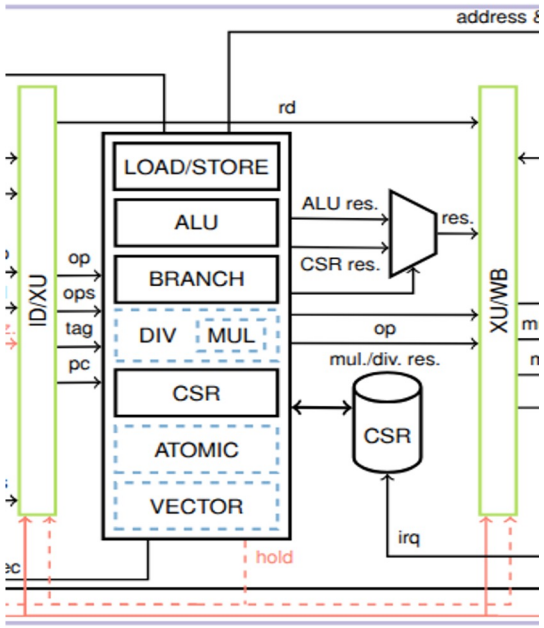
RS5 - Decode (ID)



Decode

- Extracts the operation (**op**) based on the fetched instruction
- Identifies the destination register (rd) and source registers (*rs*)
- Fetches the operands (**ops**) from the general-purpose register file (**GPR**)
- **Forwarding**
- **Hazard Detection**
- **Branch Predictor** (opcional)

RS5 – Execute (XU) (1/2)

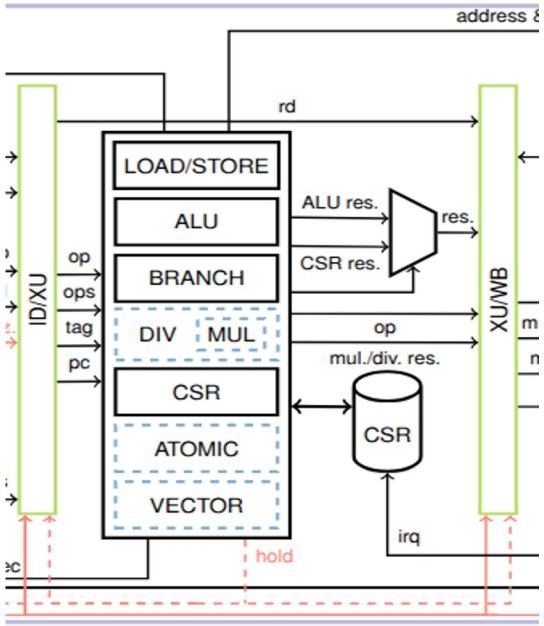


Execute

UNITS:

- **LOAD/STORE**
- **ALU**
- **BRANCH**
- **CSR**
  - Executes atomic operations on CSR
  - Manages privileges and traps
- **Multiplication** (MUL) and **Division** (DIV) – optional (Zmmul)
- **Atomic** (optional): controls read-modify-write operations in memory
- **Vector** (optional)

RS5 – Execute (XU) (2/2)



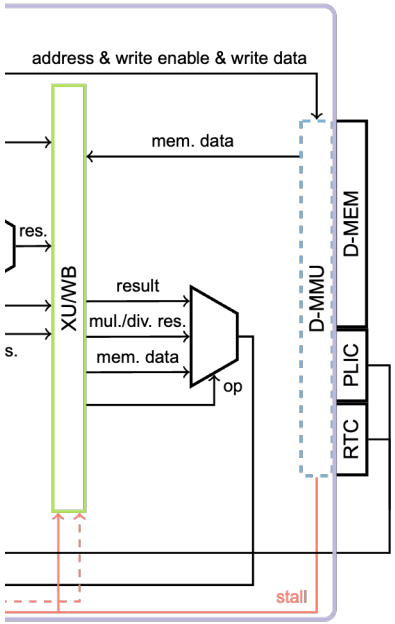
Execute

Memory Write: performed in the XU stage

Multi-Cycle Operation Management: the execute stage controls the hold signal for multi-cycle operations:

- Multiplication: typically requires 4 to 5 cycles
- Division: may take up to 32 cycles
- Atomic operations: also managed via the hold signal
- Vector operations: 1-32 cycles

RS5 – Writeback (WB)



Writeback

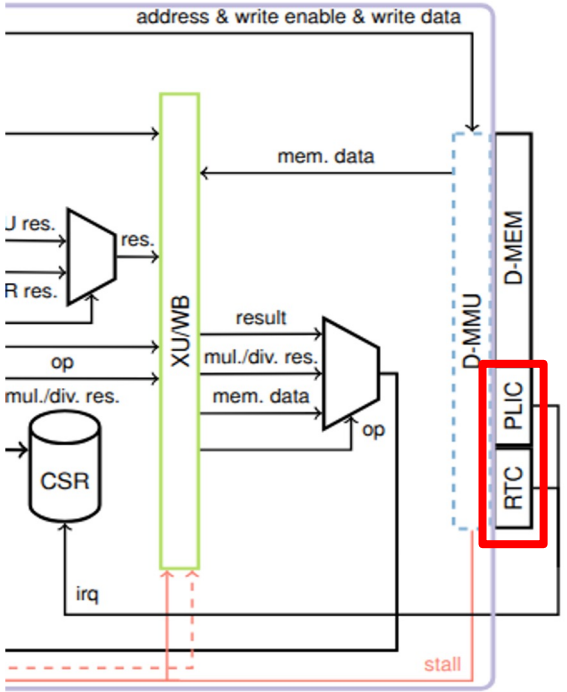
- Performs the write-back of operation results
- Writes data obtained from the data memory (D-MEM) after a read request issued by the LOAD/STORE unit
- **Optional**: a D-MMU can virtualize D-MEM addresses

RS5 – PLIC and RTC

PLIC (Platform Level Interrupt Controller)

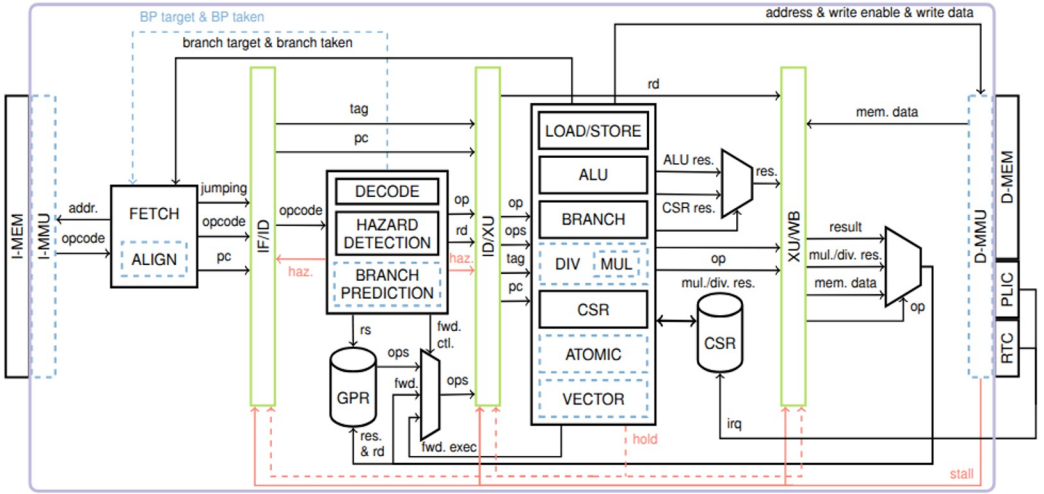
RTC (Real Time Clock)

Memory mapped peripherals



RS5 - performance

- 225.3 CoreMark @100MHz
- +26% LUTs/FFs and >2x performance w.r.t. **Ibex** (LowRISC, 2021)
- Tested on Nexys 7 FPGA



RISC-V RS5 (LASCAS 2024)

RS5 EVALUATION AND COMPARISON WITH SIMILAR CORES (Z*:ZICNTR, D*: ZIHPM, X*: XOSVM).

Core	LUTs	FFs	DSPs	Extensions	Modes	Freq. MHz	CoreMark
RS5 Baseline	**2141**	**957**	-	Z*	M, U	100	86.3
RS5	2380	1466	-	Z*, D*	M, U	100	86.3
	2553	1032	-	Z*, X*	M, U	100	86.3
	2222	842	12	Z*, Zmmul	M, U	100	212.3
	2814	1113	12	Z*, M	M, U	100	212.3
	3574	**1757**	12	Z*, M, X*, D*	M, U	100	212.3
Ibex	2184	1247	-	Z*, C	M, U	50	46.8
	2688	1329	1	Z*, M, C	M, U	50	111.6
Steel	2140	1434	-	Z*	M	50	68.0
RS5 - no LUTRAM	2721	1949	-	Z*	M, U	100	86.3
	3395	2105	12	Z*, M	M, U	100	212.3
SCR1	2938	1617	-	-	M	66	70.4
	3518	1747	4	M	M	50	114.3
CV32E40P	5111	2015	5	Z*, M, C	M, U	70	186.8

1. RS5 RISC-V

2. MEMPHIS Manycore

3. SoC-Wimed

Ruaro, M., Caimi, L. L., Fochi, V., and Moraes, F. G. (2019). **Memphis**: a framework for heterogeneous many-core SoCs generation and validation. Design Automation for Embedded Systems, 23(3-4):103-122.

Manycores

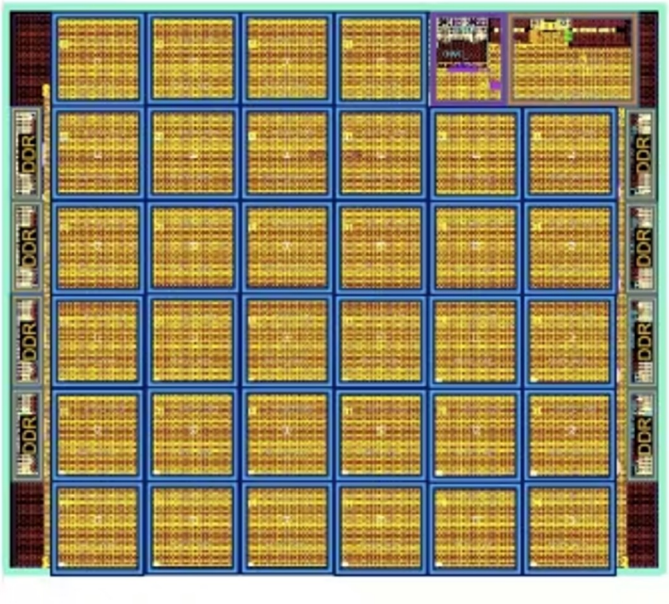
Manycore

- A SoC with hundreds to thousands of processing elements (PEs)

Network-on-Chip

- Scalability
- Parallel communications (flows)

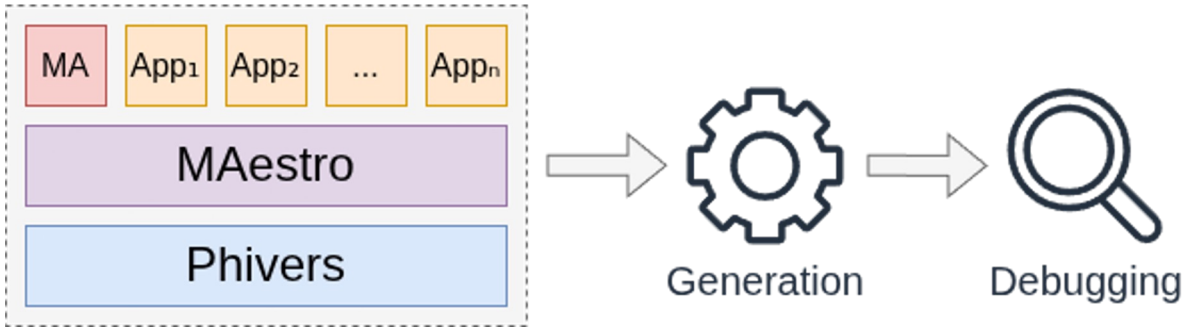
ET-SoC-1: 1’000+ RISC-V Cores



Memphis
Manycore Modeling Platform for Heterogeneous SoCs

Components

- Software stack (applications, management, support libraries)
- Operating System (MAestro)
- Hardware layer → Phivers (Processor Hive for RS5)
- Generation tools
- Debugging tools



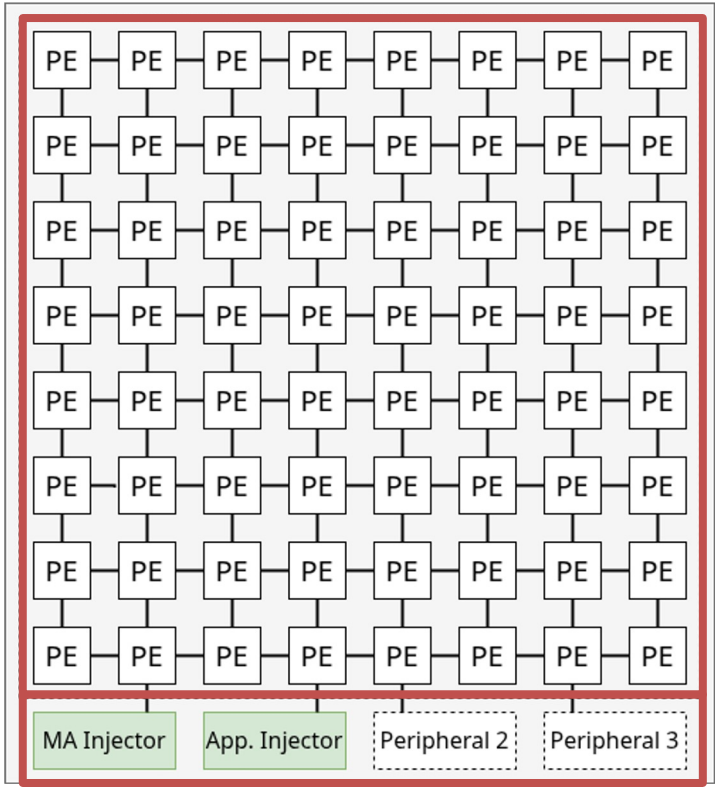
Memphis
Manycore Modeling Platform for Heterogeneous SoCs

Homogeneous region: **GPPC**

- **G**eneral **P**urpose **P**rocessing **C**ores

Heterogeneous region: **Peripherals**

- Connected to **PE** borders
- Provides:
  - I/O Interface
    - Default: **Application Injector**
    - Default: **MA Injector**
  - Hardware acceleration

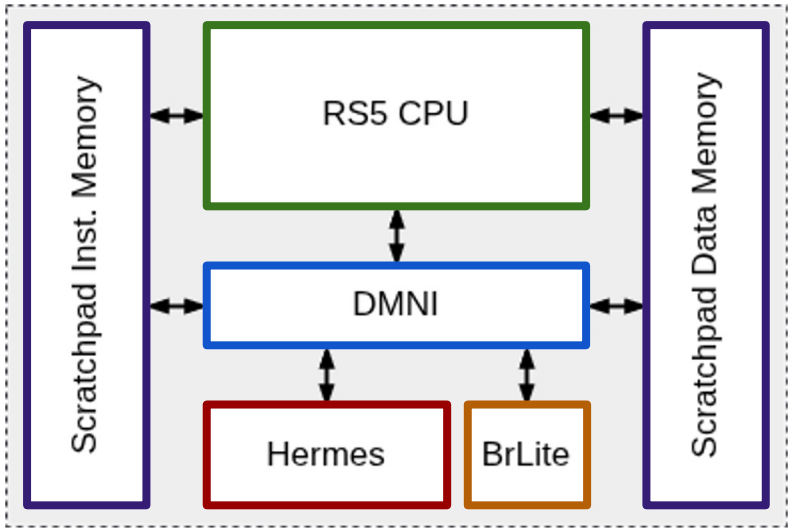


Memphis
Manycore Modeling Platform for Heterogeneous SoCs

PE – **Homogeneous** region

Processing **E**lement:

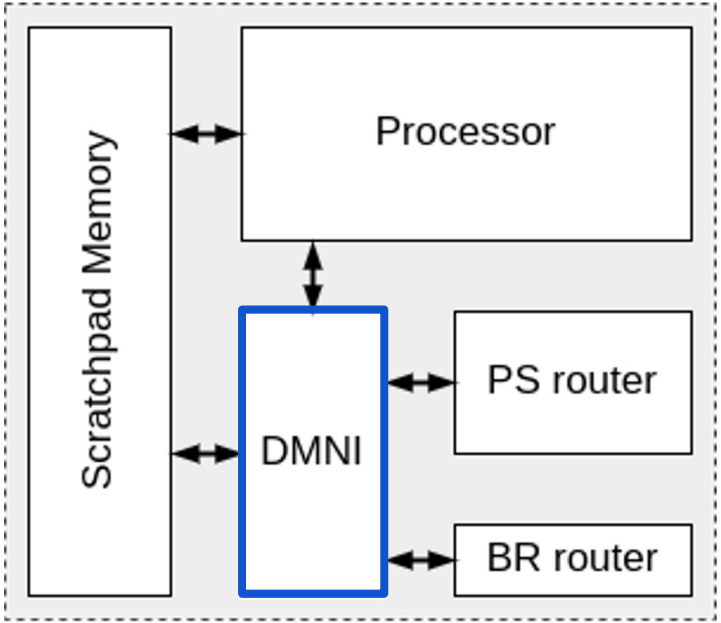
- RS5 **CPU**
- **DMNI** (DMA + NI)
- **Scratchpad memories**
- **Hermes** router
- **BrLite** router



DMNI

Direct Memory Network Interface

- Interface between local memory and the 2 NoCs
- Specialized for NoC-based manycores



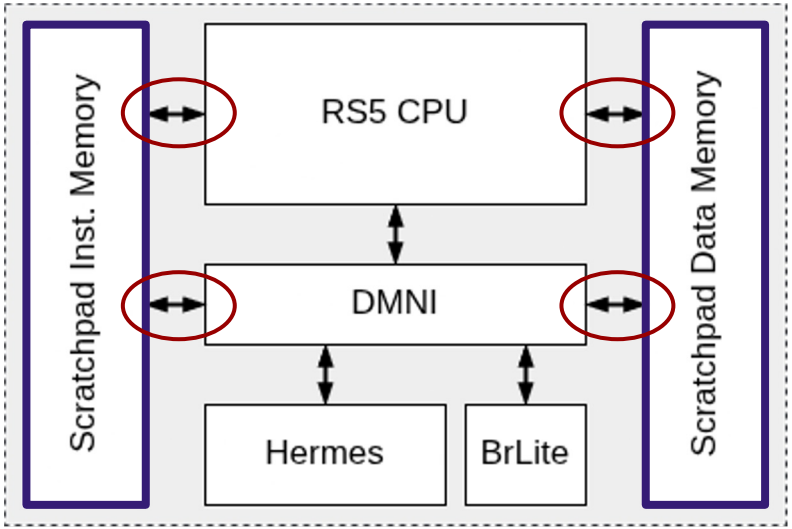
Ruaro, M., Lazzarotto, F. B., Marcon, C. A., and Moraes, F. G. (2016). DMNI: A specialized network interface for NoC-based MPSoCs. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), pages 1202-1205.

Local Memory

Scratchpad memories

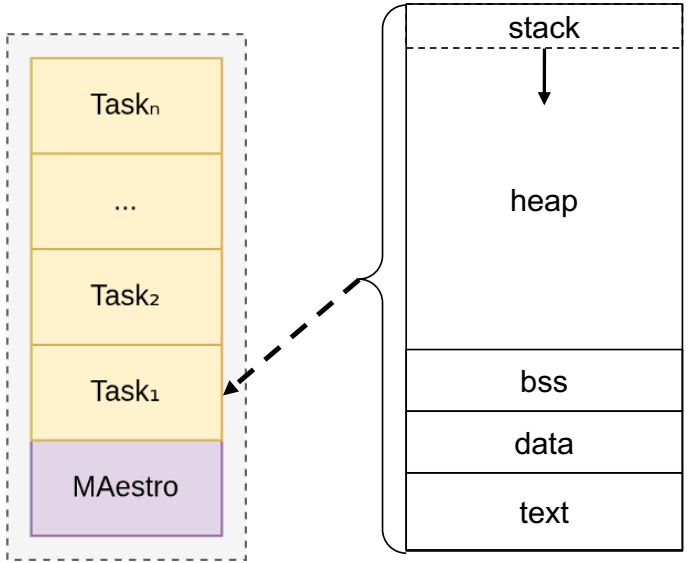
Dual-port

- CPU access
- DMNI access



Parameterizable n# of pages

- **Xosvm** extension



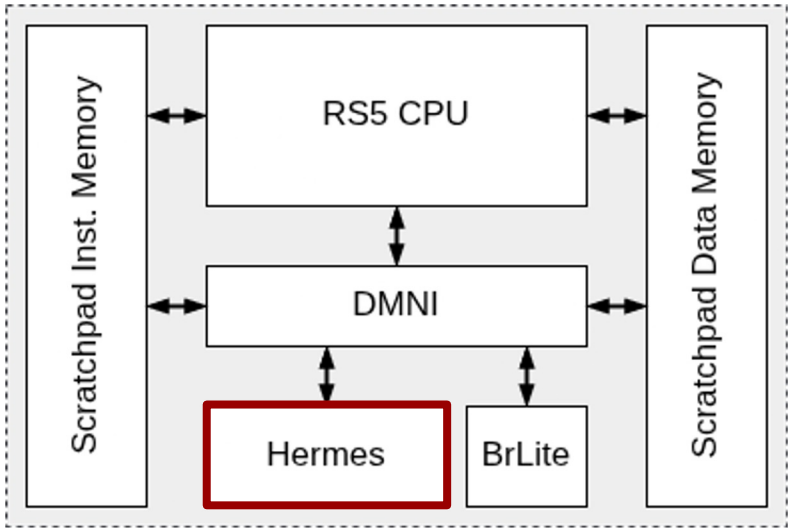
Packet-Switching Router

Hermes router

- XY routing
- Packet switching
- Wormhole credit-based control flow
- End-of-packet (EOP)

Hermes has other versions:

- Asynchronous
- Virtual-channel
- Circuit-switching

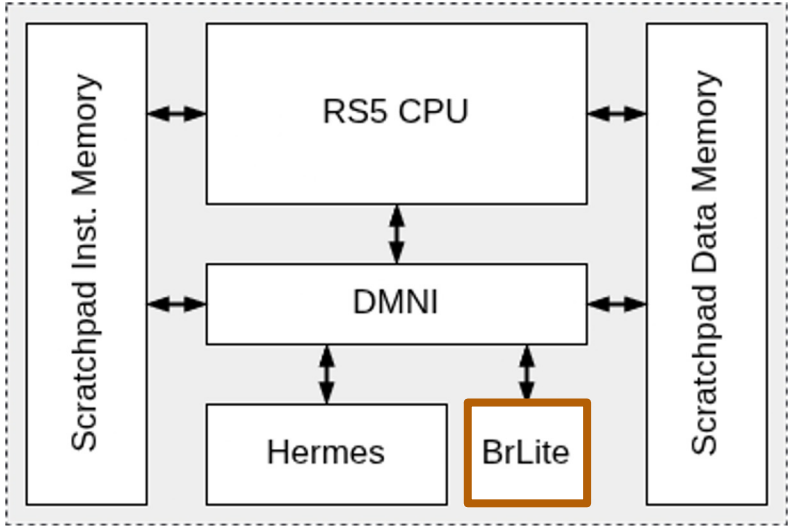


Moraes, F., Calazans, N., Mello, A., Möller, L., and Ost, L. (**2004**). HERMES: an infrastructure for low area overhead packet-switching networks on chip. Integration, 38(1):69-93.

Broadcast Router

BrLite router

- Based on BrNoC
- **Broadcast** transmission only
- Single-flit messages
- Fast transmission of control messages
- “Fault-tolerant” NoC

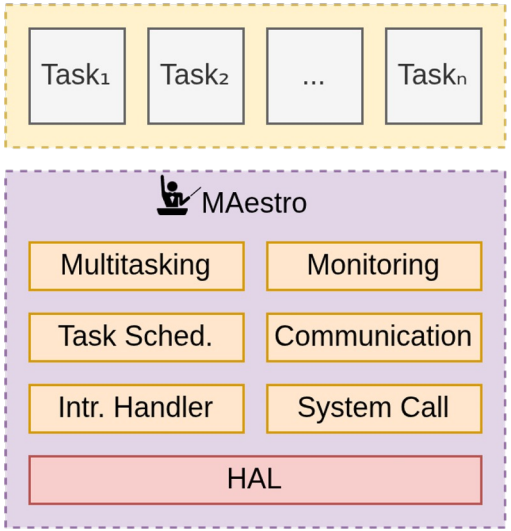


Wachter, E., Caimi, L., Fochi, V., Munhoz, D., Moraes, F. (2017). BrNoC: A broadcast NoC for control messages in many-core systems. Microelectronics Journal, 68:69-77.

Operating System

MAestro Kernel (C Language): ~20 KB

- User tasks
  - Virtual Memory
  - Scheduler (best-effort or real-time)
- System Calls
- Message Passing Interface (MPI API)



Ruaro, M., and Moraes, F. G. (2016). Dynamic **real-time scheduler** for large-scale MPSoCs. In Proceedings of the Great Lakes Symposium on VLSI, pages 341-346.

Applications - toolchain

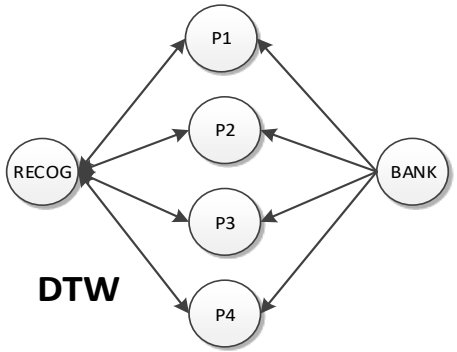
Support for recent GCC toolchain (14.2.0) – auto-vectorization

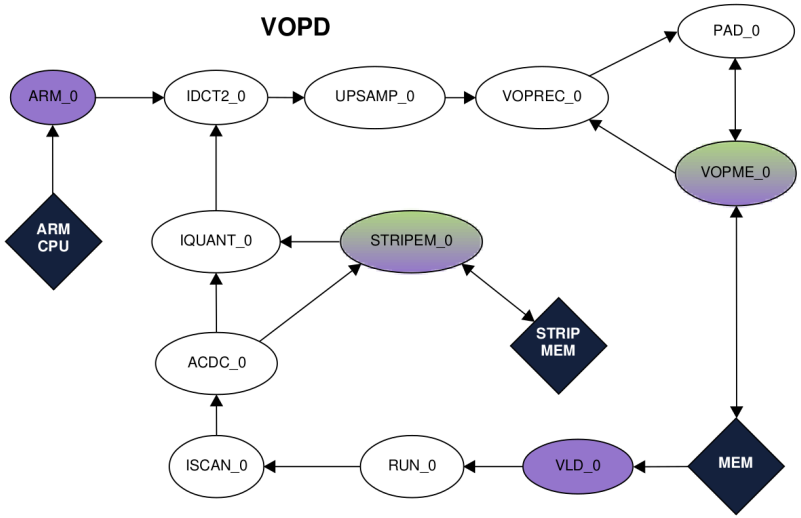
- **Newlib support (4.4.0)**
- **Default: newlib-nano specs**
  - Complete libc support
  - Software floating-point
  - 6 POSIX calls
  - 8 custom calls (NoC)

Application

Application

- Set of **tasks**
- Modeled as a CTG (**C**ommunicating **T**ask **G**raph)

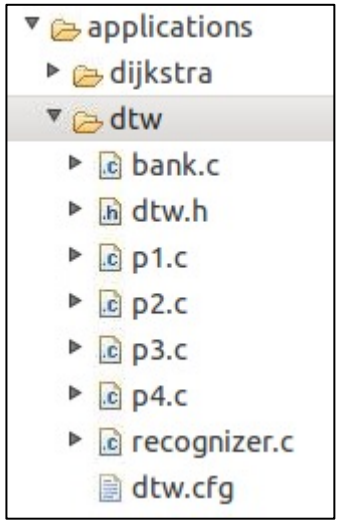




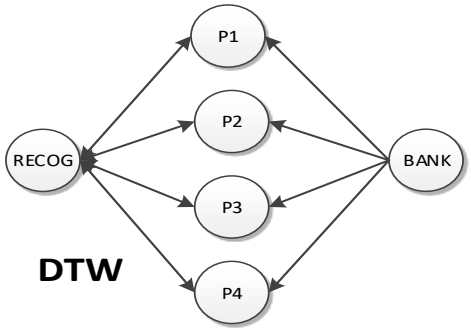
Task

Task is a .c file which perform some computation and communication with other(s) task(s)

Example of an application task files



Example of a task code




```c
int main(){

    int test[SIZE][SIZE];
    int pattern[SIZE][SIZE];
    int result, j;

    Receive(&msg, recognizer);

    Echo("Task P1 INIT\n");

    memcpy(test, msg.msg, sizeof(test));

    for(j=0; j<PATTERN_PER_TASK; j++){

        Echo("Task P1 FOR\n");

        memset(msg.msg,0, sizeof(int)*MSG_SIZE);

        Receive(&msg, bank);

        //Echo("Task P1 received pattern from bank\n");

        memcpy(pattern, msg.msg, sizeof(pattern));

        result = dynamicTimeWarping(test, pattern);

        msg.length = 1;

        msg.msg[0] = result;

        Send(&msg, recognizer);
    }

    Echo("Task P1 FINISHEDD IN\n");
    Echo(itoa(GetTick()));

    exit();
}
```

Main View

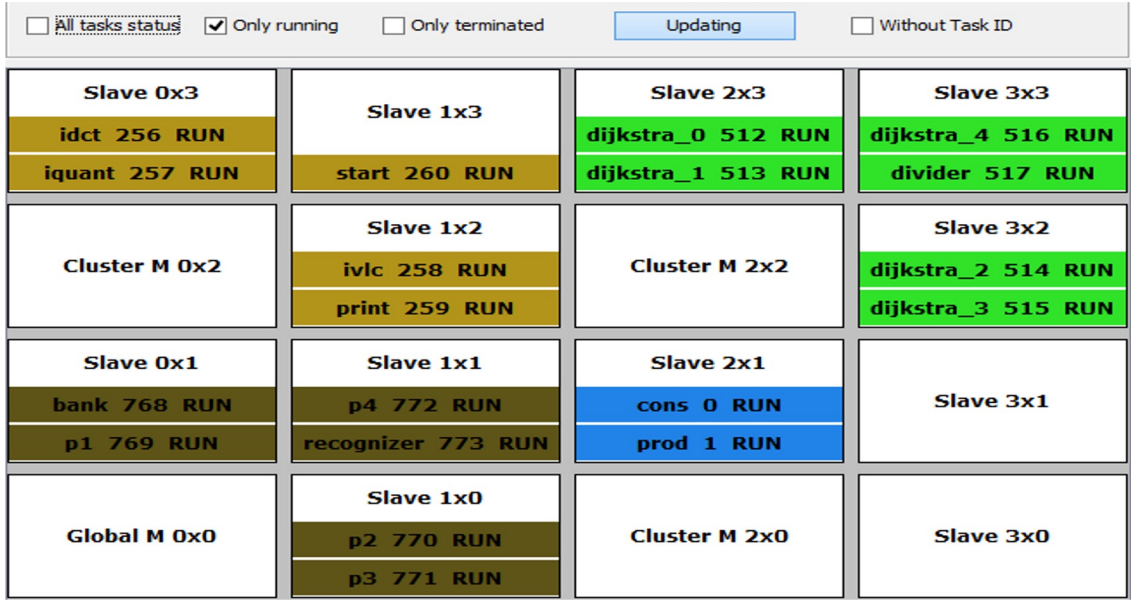
Debug possibilities

- Communication flows
- Routing
- Link utilization
- Management Protocols

Mapping View

Debug possibilities

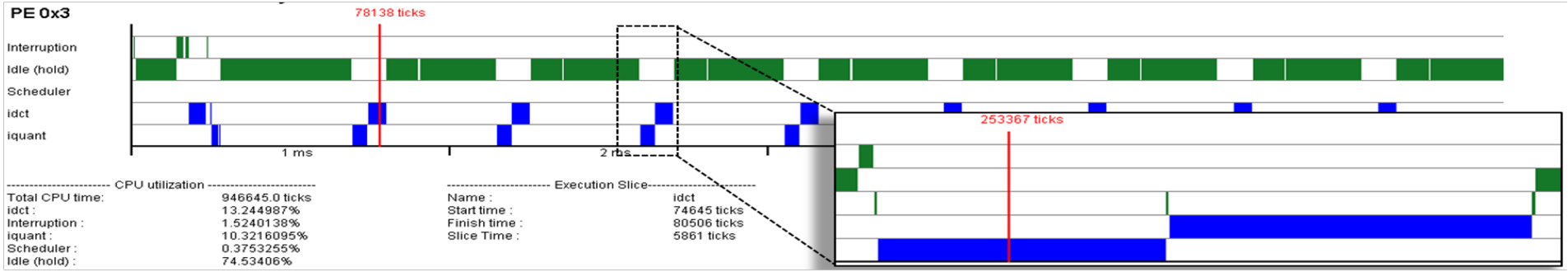
- Task mapping algorithm
- PEs occupation
- Task execution status



CPU Utilization View

Debug possibilities:

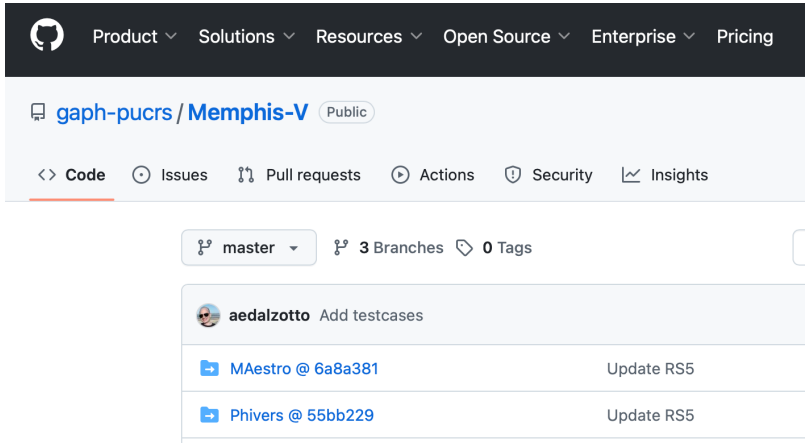
- Scheduling algorithms
- OS and task bugs
- Other software malfunctions



Resources

Code, installation and usage instructions:

https://github.com/gaph-pucrs/memphis-v

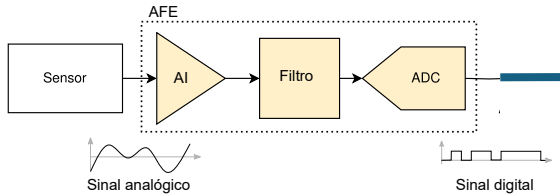


1. **RS5 RISC-V**
2. **MEMPHIS Manycore**
3. **SoC-Wimed**

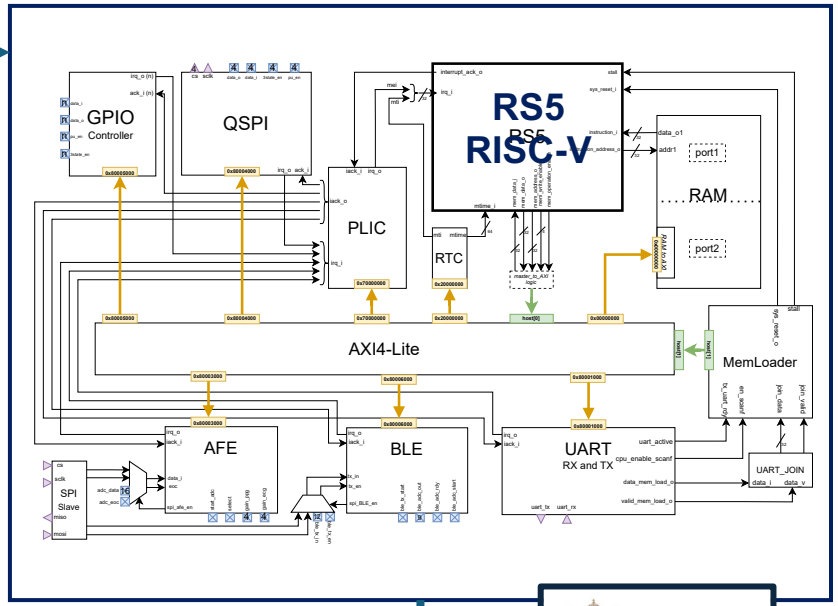
SoC-WiMed: Wireless SoC for Medical Monitoring of Vital Signs with a Focus on Security and Low Power Consumption

PUCRS UNISINOS unipampa EnSilica

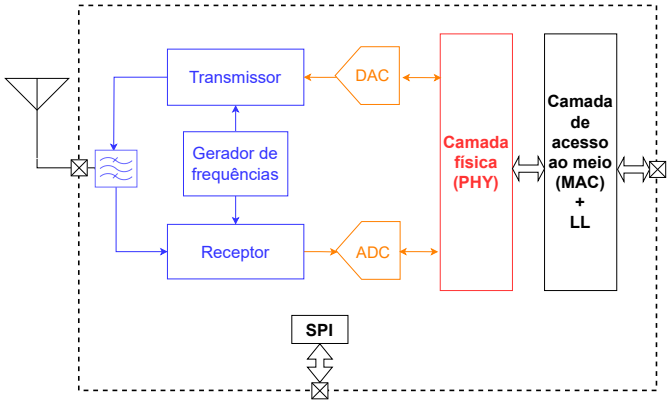
Overview of the SoC-WiMed



unipampa



PUCRS

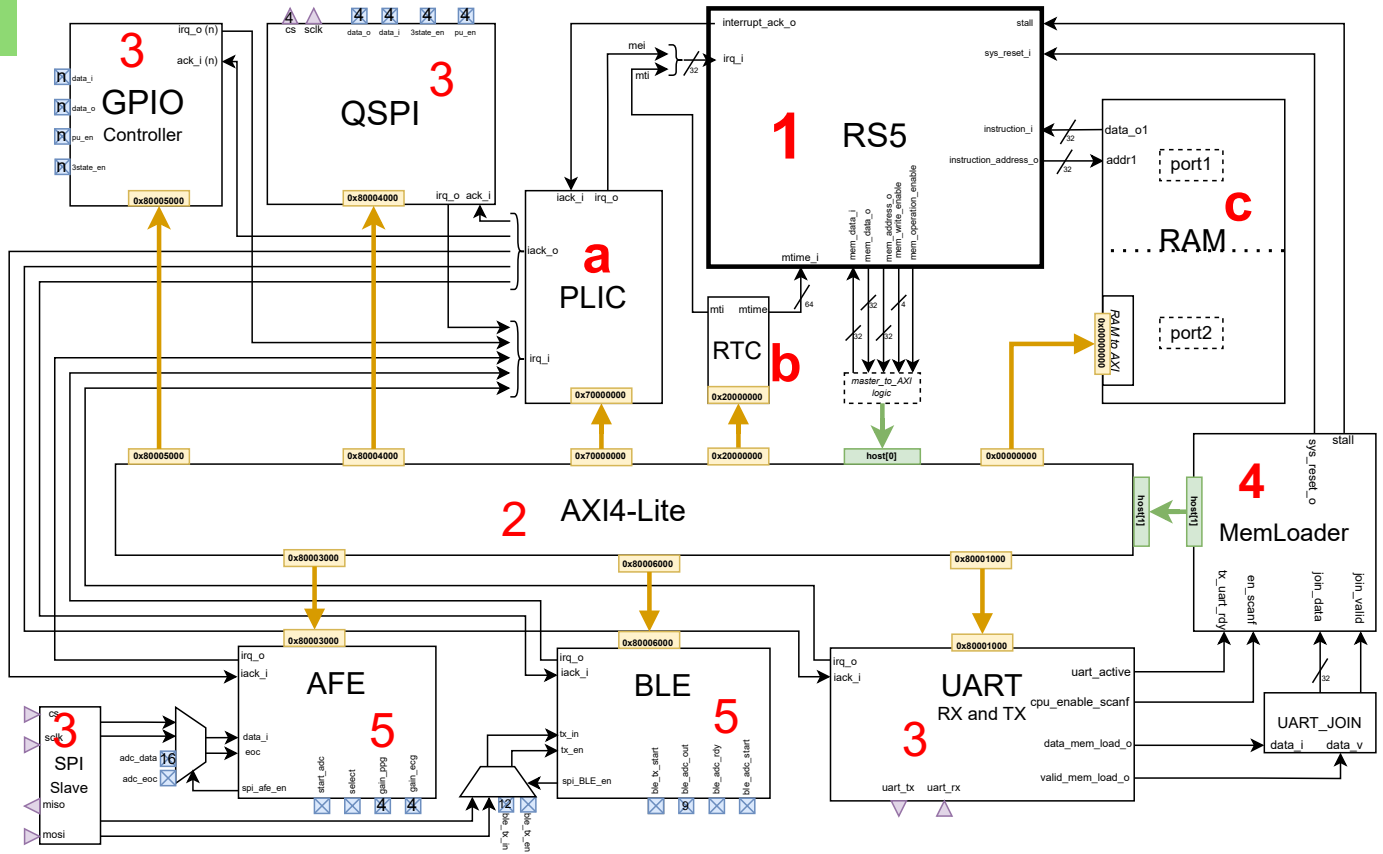


UNISINOS

Digital Block (PUCRS)

Hardware:

1. **RS5 - RV32IMAC**
   a. PLIC
   b. RTC
   c. Memory
2. **AXI-Lite bus**
3. **UART /SPI / GPIO / QSPI**
4. **MemLoader**
5. **Interface with other subsystems – AFE and BLE**



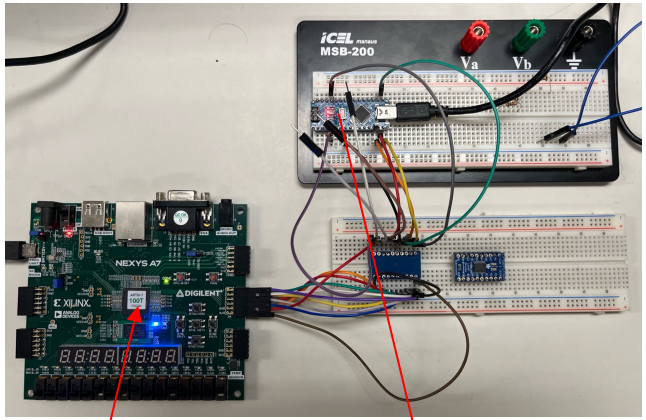
https://github.com/gaph-pucrs/RS5

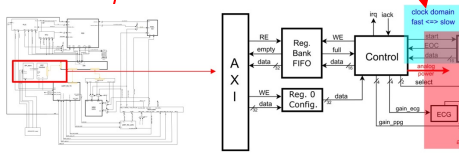
Software - Zephyr RTOS

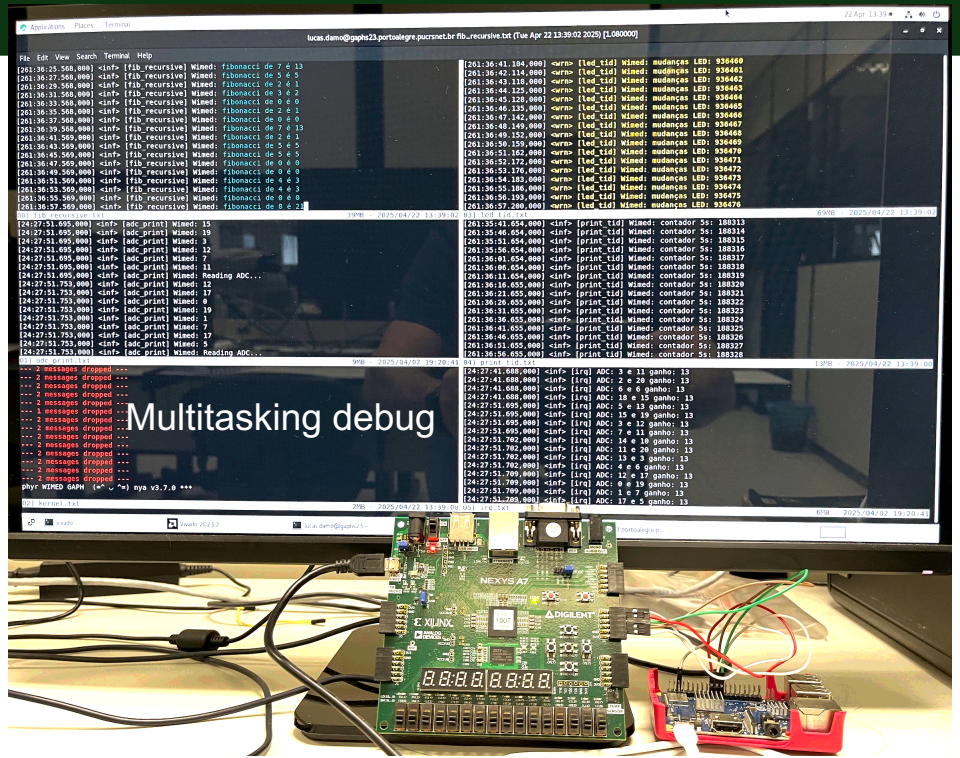
- Deployment of the open-source Zephyr OS
- Low memory requirements (< 20 KB)
- Configuration and development of drivers
- Bluetooth support
- Multitasking capability

FPGA Prototyping

- FPGA Artix7 (@100MHz)
- Arduino – simulates low-frequency ADC data





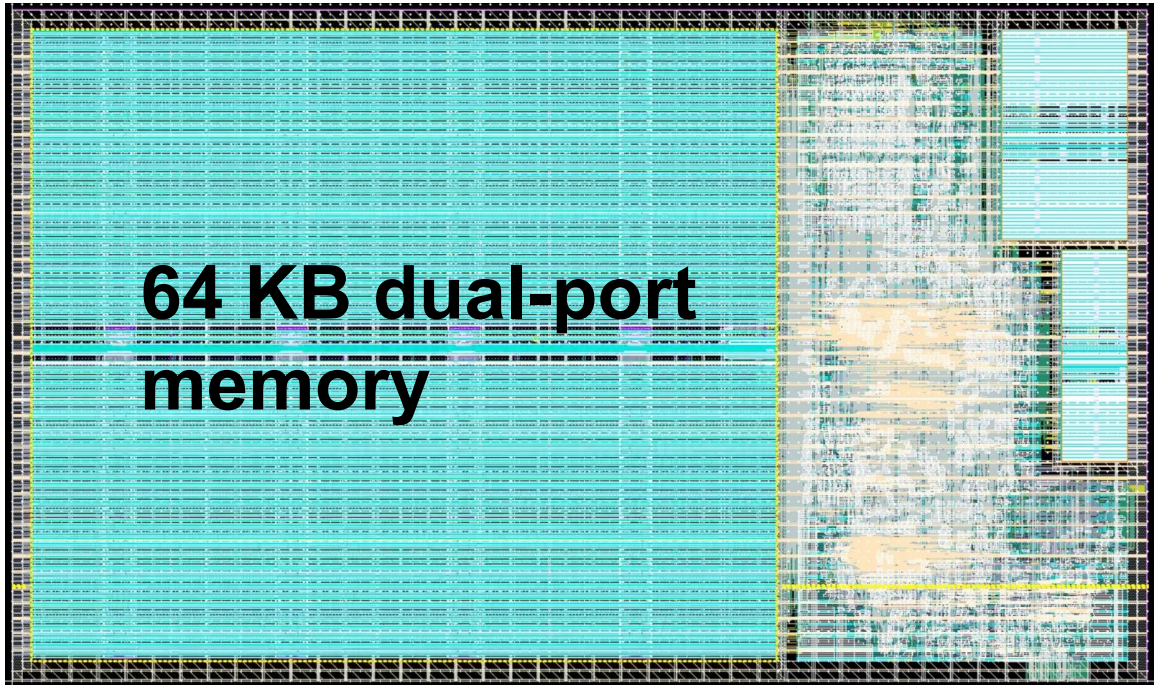


Arduino

USB-Serial connection
AES 128-bit encryption (Zkne)

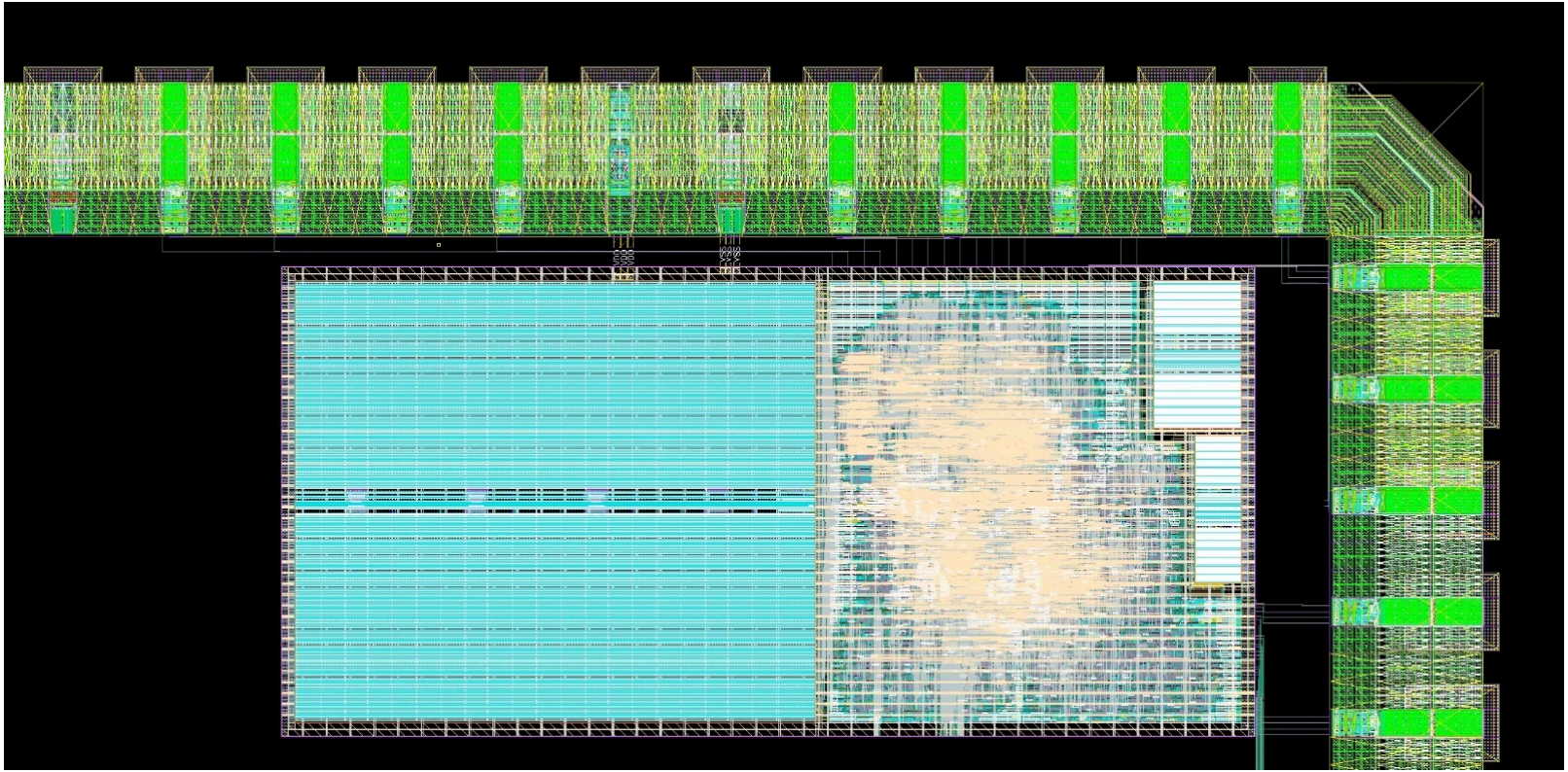
ASIC Implementation – TSMC 28 nm

- Post-synthesis simulation with physical memories @256MHz in the 3 corners - signoff ok (setup/hold)
- GDS: 620 µm x 380 µm



ASIC Implementation

- Analog on top – integration with the pad ring



Thanks for the attention

Q&A