CPROC
A RISC-V processor for front-end ASICs

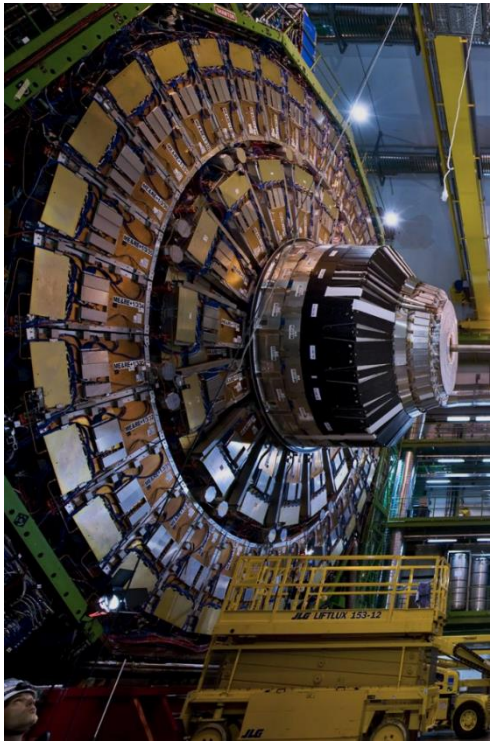
Frederic DULUCQ, Pierrick DINAUCOURT, Rama SYLLA,
Quentin MADARIAGA, Stéphane CALLIER, Duy Manh NGUYEN

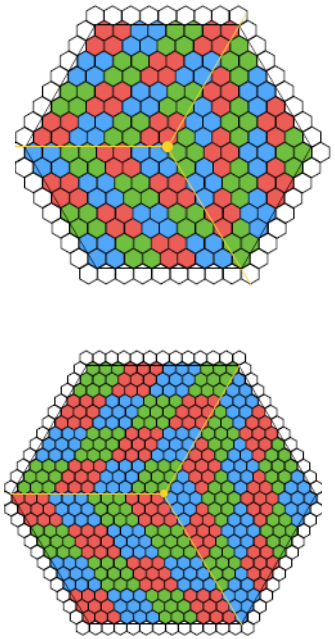
June, 2023

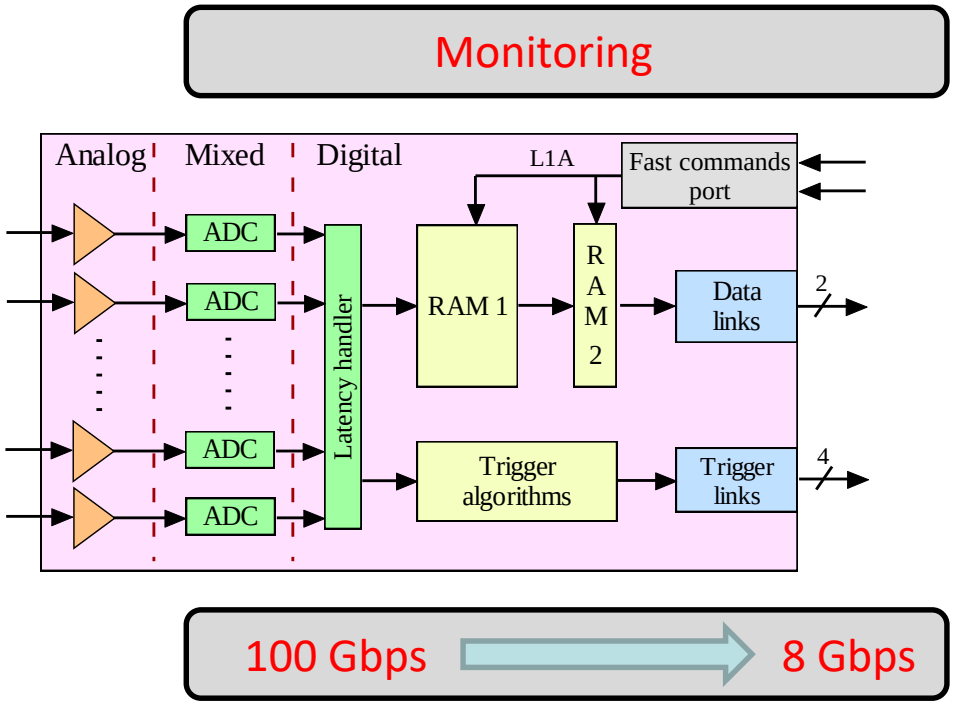
Organization for Micro-Electronics desiGn and Applications

From 2016 with CMS HGCROC…

- ASICs include a lot of logic mainly hardwired
  - Overall behavior can be selected (sensor type, trigger cells 2x2 or 3x3)
  - Some functions can be tuned with parameters (> 10k in HGCROC)
- Many data truncation, compression and algorithm decided at early stages (FE ASICs)
  - Due to links, power, fibers requirements







Embed reprogrammable monitoring/algorithm in future ASICs ?

…to 2023 and CPROC demonstrator

- Three main objectives for the CPROC development:
  - Select/Define the requirements of an embedded processor for front-end ASICs
  - Acquire knowhow on the compiler toolchain (software) versus hardware in the ASIC
  - Feedback (even preliminary) on such component

On the road to FPGASIC ?
Quick answer: for specific parts but not on everything…

- 2015: Creation of the RISC-V foundation

- 2020-2022: two internships on RISC-V processor at OMEGA

- May 2023: submission of **CPROC**, a RISCV demonstrator

What is RISC-V and ISA

RISC-V* is an open standard instruction set architecture (ISA) based on established RISC principles. Unlike most other ISA designs, RISC-V is provided under royalty-free open-source licenses.

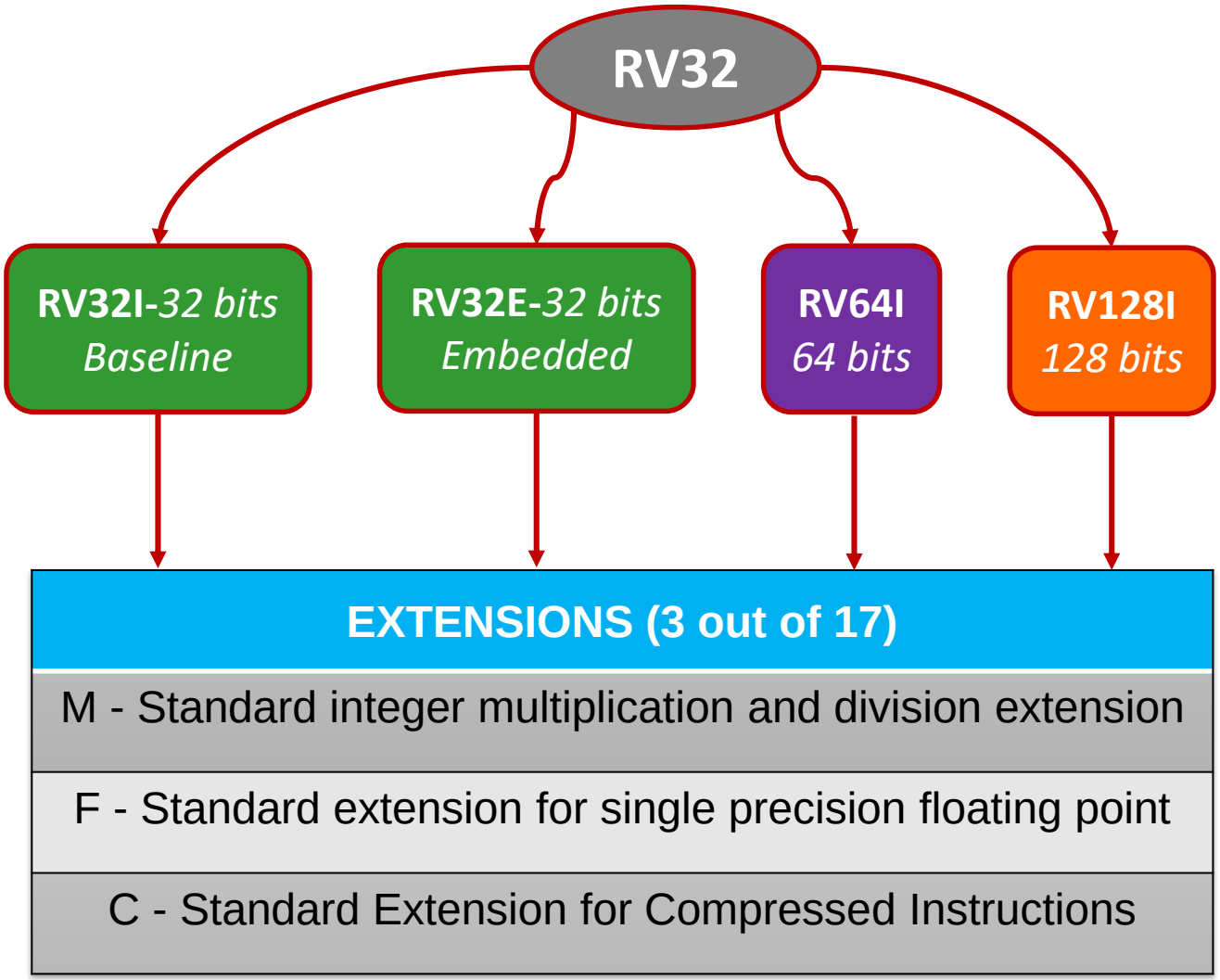
*Foundation formed in 2015

An **ISA** is an abstract model of a computer […]. A realization of an ISA is called an implementation. An ISA permits multiple implementations…

x86 and x64 ISA (mainly in computers)

ARMv8 ISA (smartphone, FPGA…)

RISC-V ISA: Unprivileged Specification 20191213

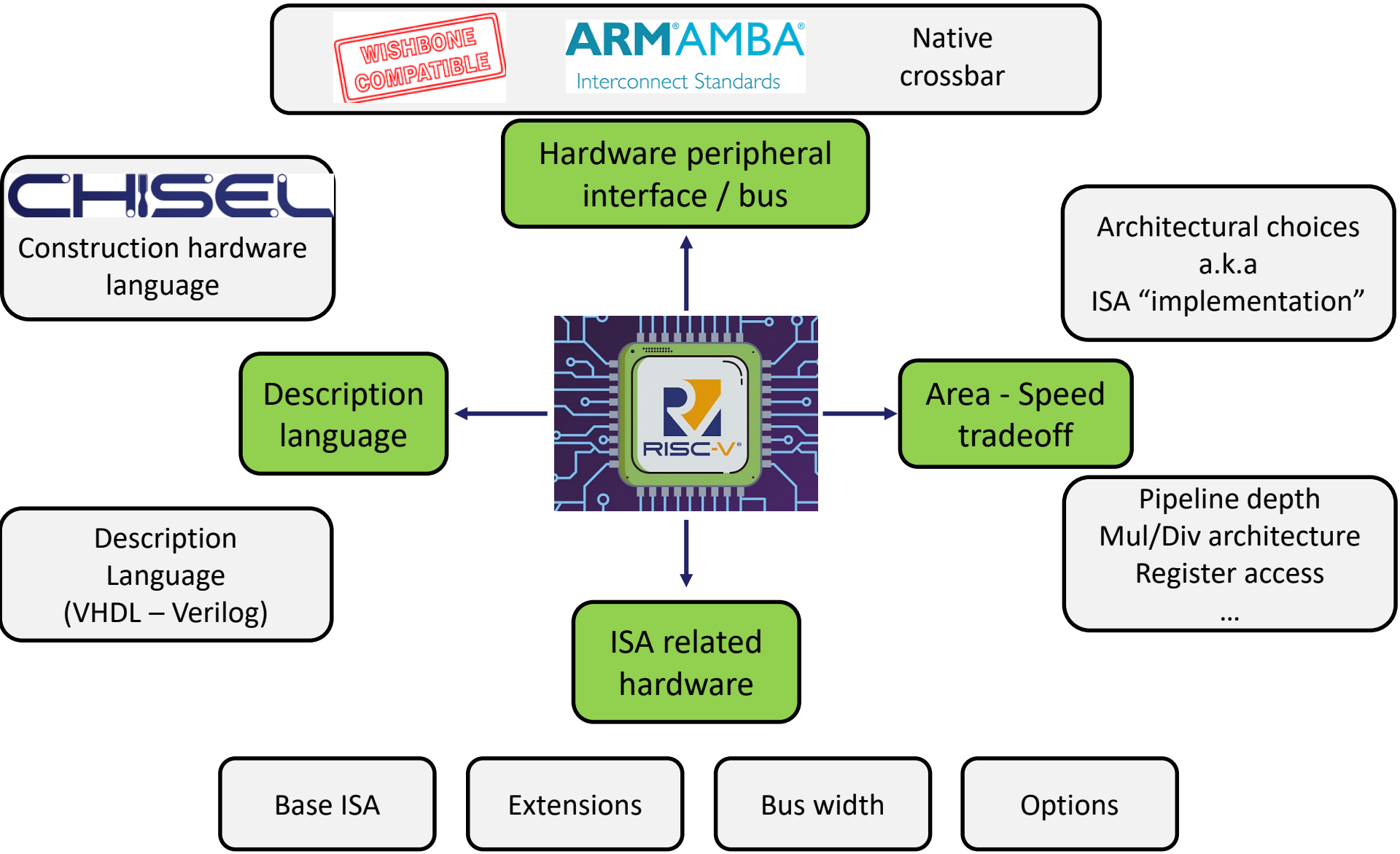


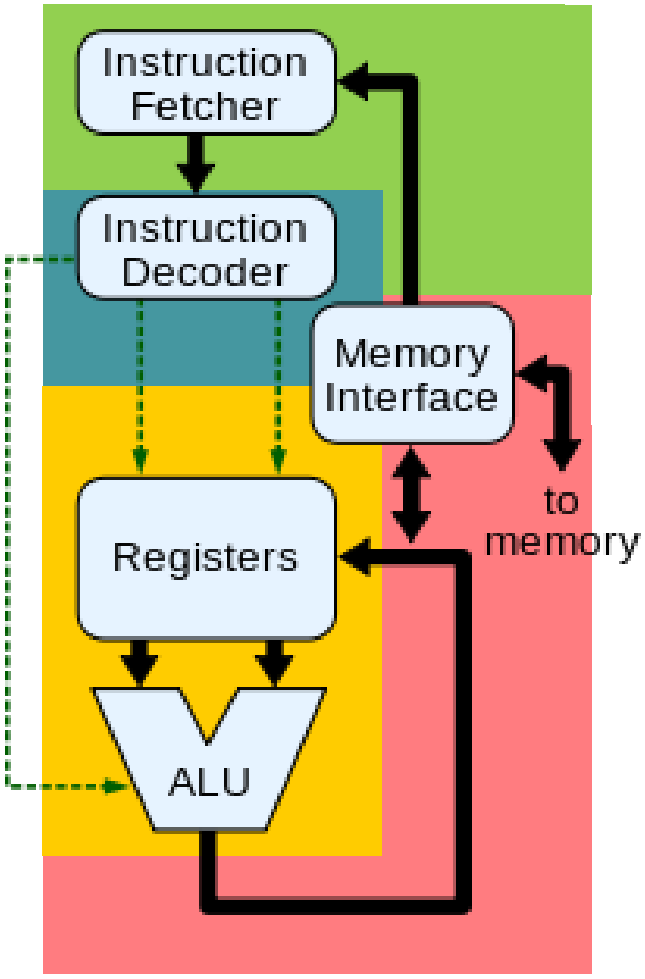
Four base ISAs*

Base	Version	Status
RVWMO	2.0	**Ratified**
RV32I	**2.1**	**Ratified**
RV64I	**2.1**	**Ratified**
RV32E	*1.9*	*Draft*
RV128I	*1.7*	*Draft*
Extension	**Version**	**Status**
M	**2.0**	**Ratified**
A	**2.1**	**Ratified**
F	**2.2**	**Ratified**
D	**2.2**	**Ratified**
Q	**2.2**	**Ratified**
C	**2.0**	**Ratified**
Counters	*2.0*	*Draft*
L	*0.0*	*Draft*
B	*0.0*	*Draft*
J	*0.0*	*Draft*
T	*0.0*	*Draft*
P	*0.2*	*Draft*
V	*0.7*	*Draft*
Zicsr	**2.0**	**Ratified**
Zifencei	**2.0**	**Ratified**
Zam	*0.1*	*Draft*
Ztso	*0.1*	*Frozen*

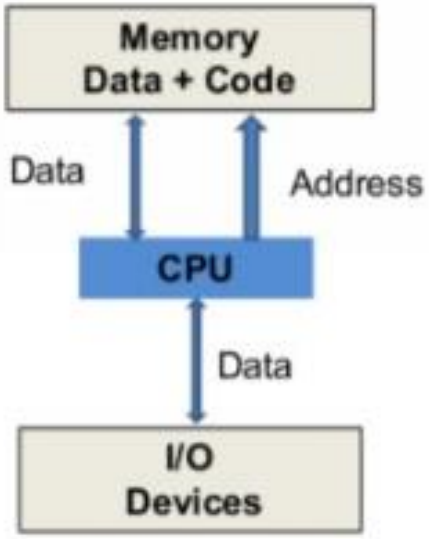
**The four base ISAs in RISC-V are treated as distinct base ISAs.*

How to build a RISC-V processor ?

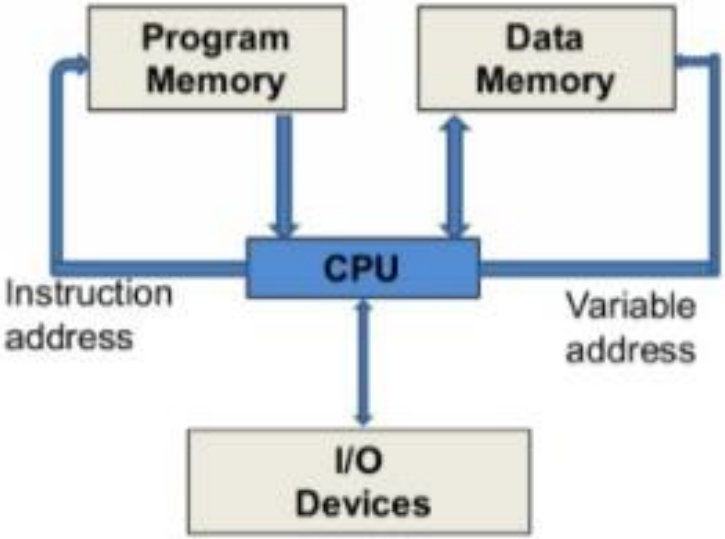




Cycle per Instruction: 4
Pipelined or not



Von Neumann



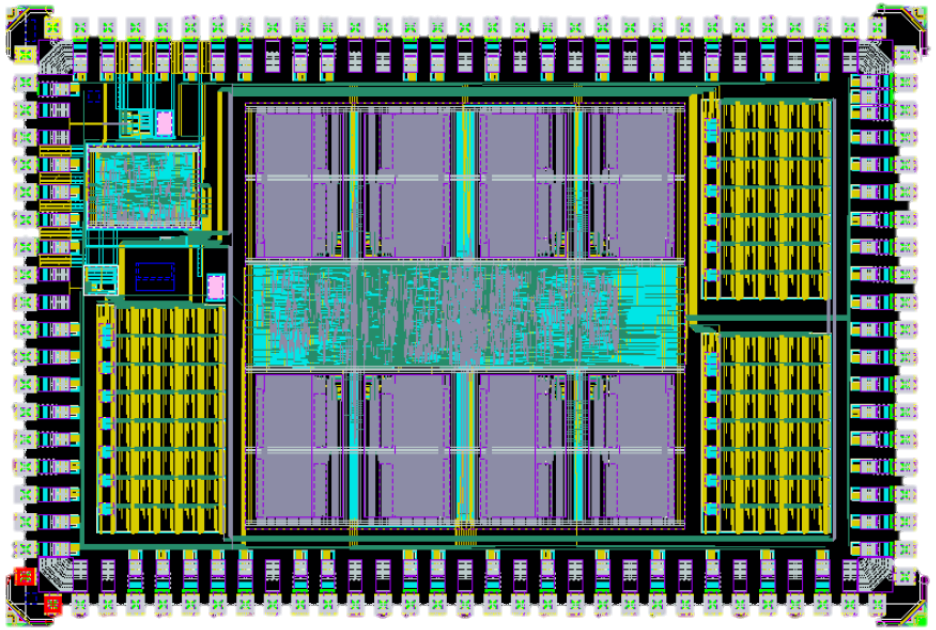
Harvard

CPROC (based on PicoRV32)

- CPROC is a fork of an existing open RISC-V project named PicoRV32

- The open project selection was made around three criteria:
  - A large community / users
  - Written in a directly synthesizable language (Verilog here)
  - With support of an already known interface bus

Floorplan
S. CALLIER

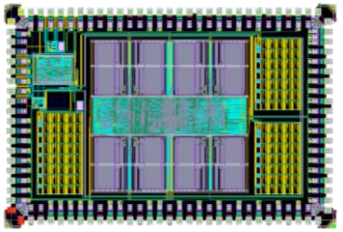


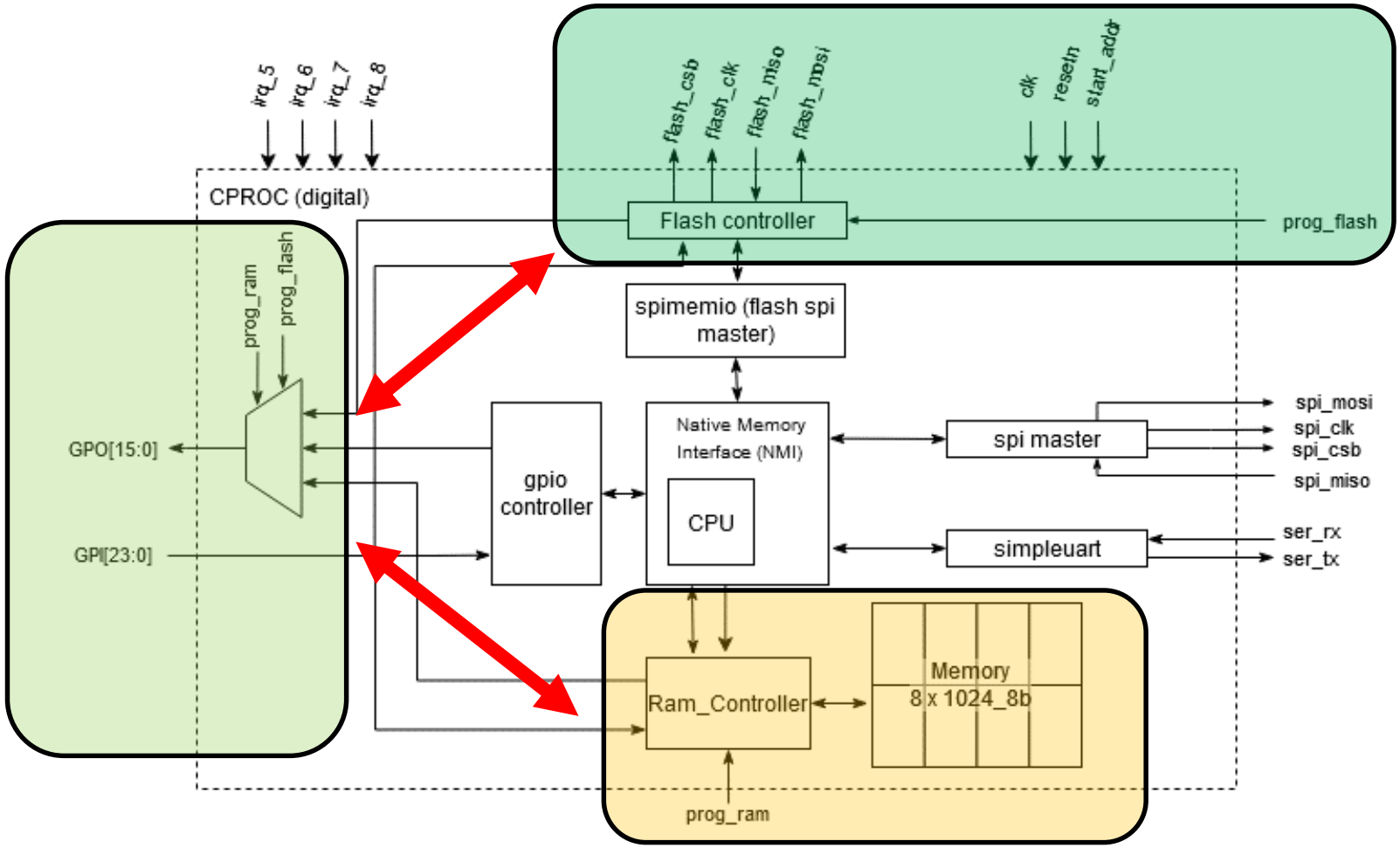
- Main features
  - As small as possible
  - Internal 8 kB SRAM (2048 words x 32 bits)
  - External flash support
  - SPI and UART connections
  - 24 GPIs, 16GPOs
  - 4 programmable IRQs
  - 2 start address choice

- By keeping in mind that **CPROC** is a RISCV demonstrator (also for teaching purpose)

CPROC changes (compared to PicoRV32)

- 3 main changes were applied:
  - Processor bypass through GPIO
  - Start address is a parameter to start from Flash or embedded RAM
  - Programming through GPIO (internal or external memories)



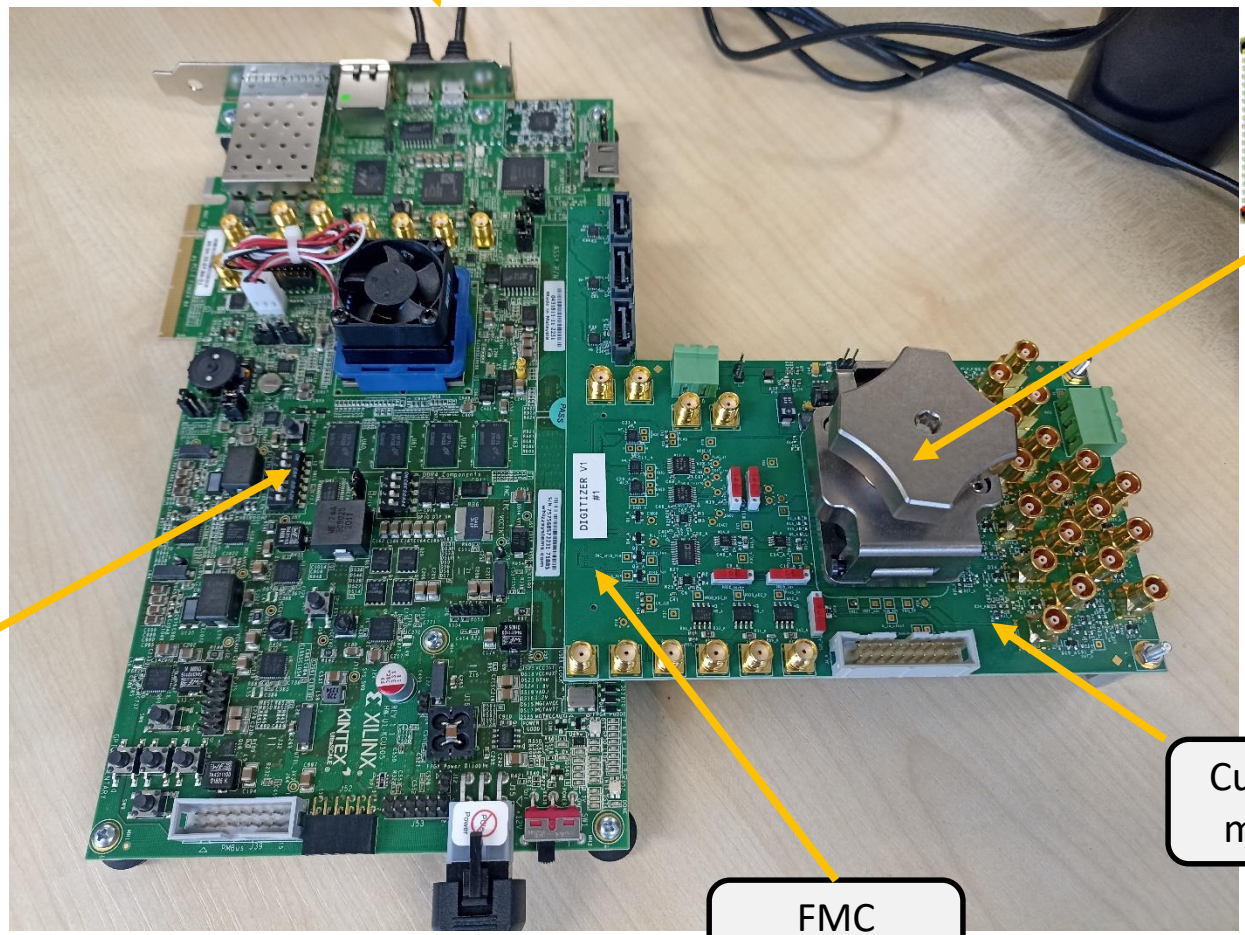


From HKROC to CPROC testboard setup

TeraTerm serial terminal

Monitor
Program
Test

CPROC
BGA socket



Commercial
KCU105 board

Custom CPROC
motherboard

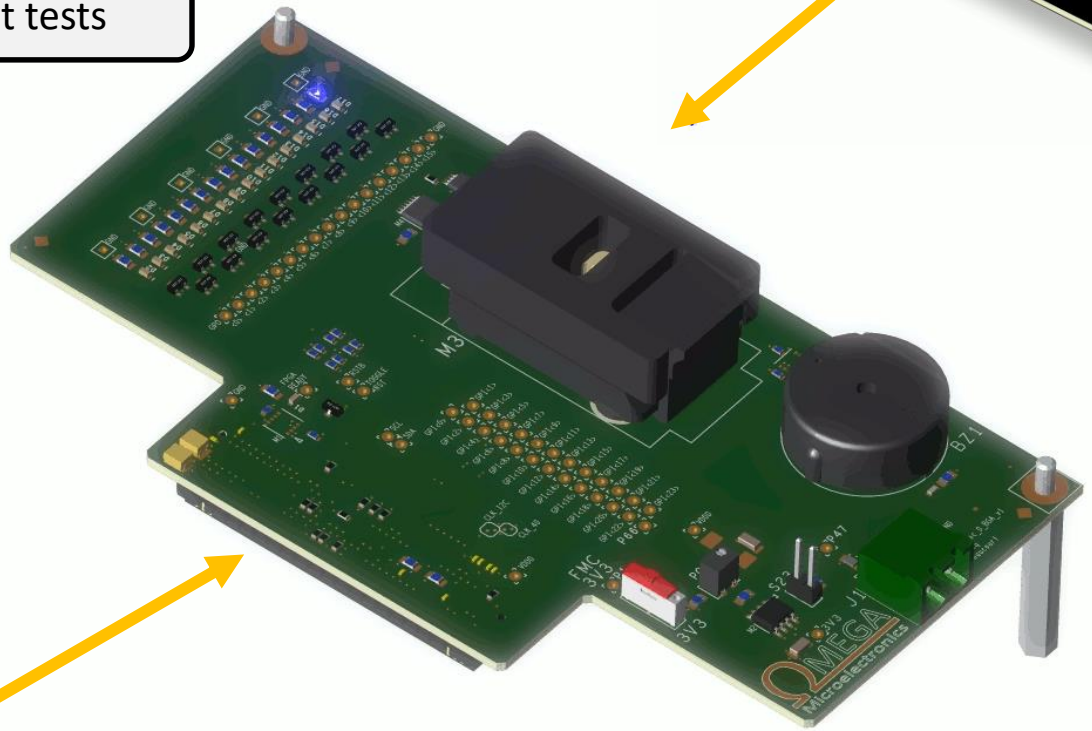
FMC
connector

CPROC testboard and BGA

Dedicated testboard
BGA and substrate design
Pierrick DINAUCOURT

LADOC substrate reuse - WBBGA

K2000 first tests

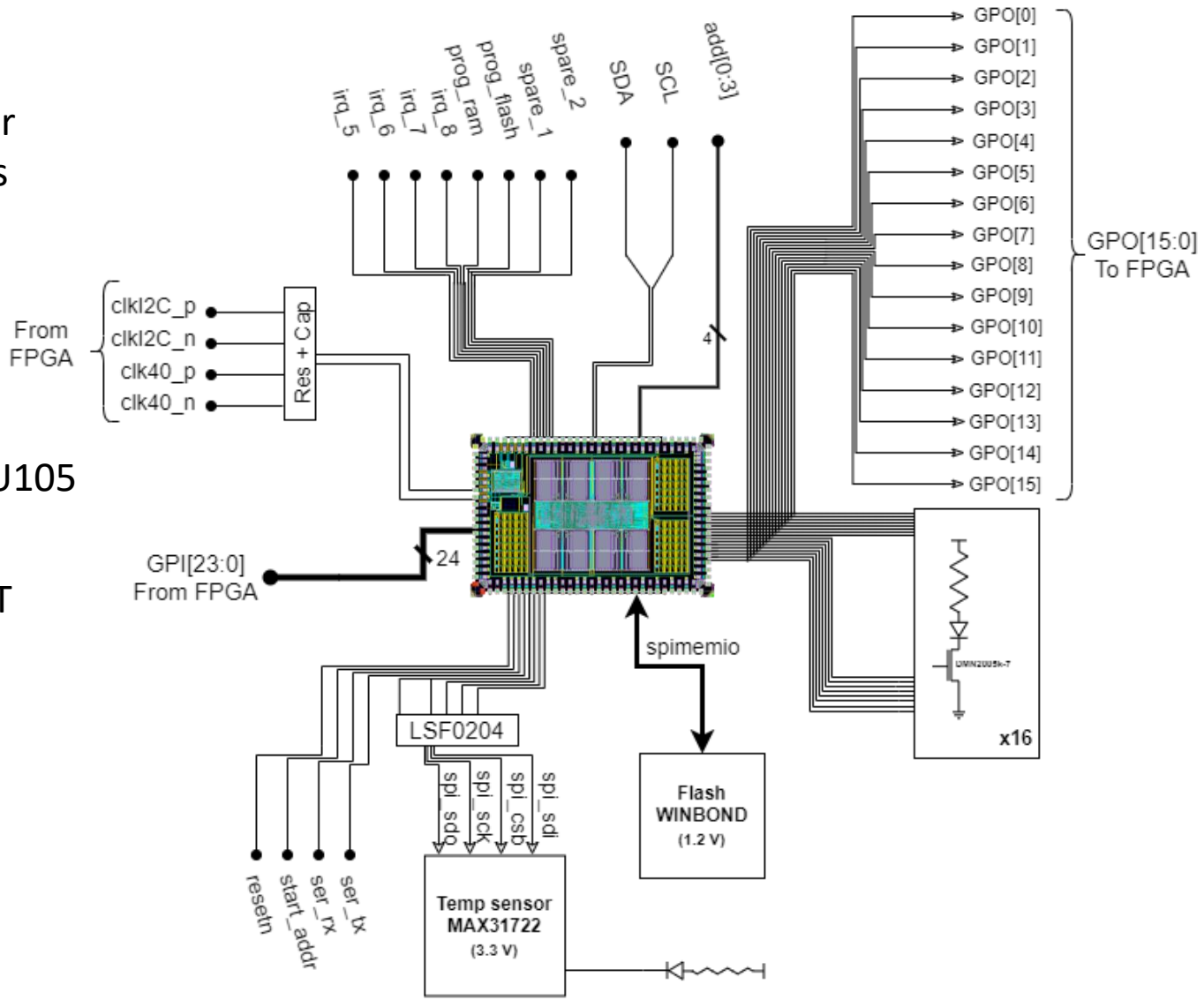


Power through FMC

FMC connector
To KCU 105

CPROC motherboard detail

- FMC connection to the KCU105 board
- Temperature sensor for CPROC SPI master tests
- 64 Mb SPI like Flash (user defined program)
- GPIO connected to KCU105 (LED for all the outputs)
- Same for IRQ and UART
- 3 modes available:
  - RAM programming
  - FLASH programming
  - Start pointer



CPROC tests through terminal

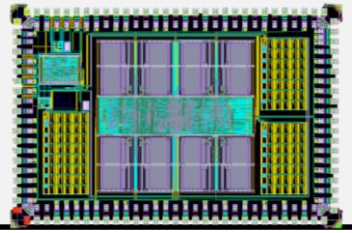
UART

Xilinx SoftCore

MicroBlaze

Custom

CPROC



CPROC test menu

1: I2C tests
2: CPROC RISC-V tests
3: Auto tests

I2C tests

1: I2C Read
2: I2C Write
3: I2C multibyte read
4: I2C multibyte write
5: I2C internal reg read
6: I2C internal reg write
7: Broadcast tests
8: I2C scan
0: main menu

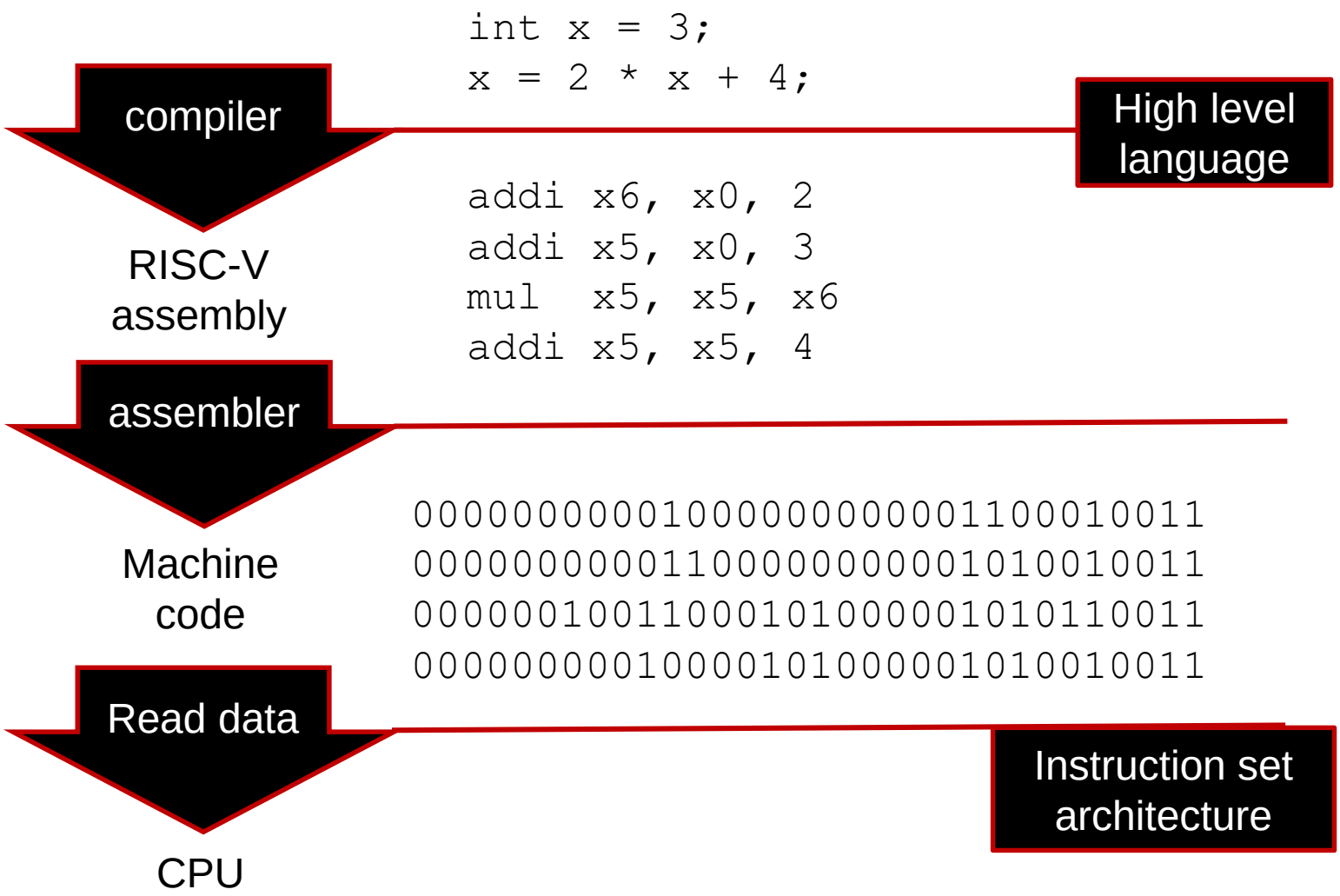
CPROC RISC-V tests

1: Program FLASH
2: Program RAM
3: Verify FLASH
4: Verify RAM
5: Start program
6: Execute default program
7: Enable IRQ
0: main menu

Auto tests

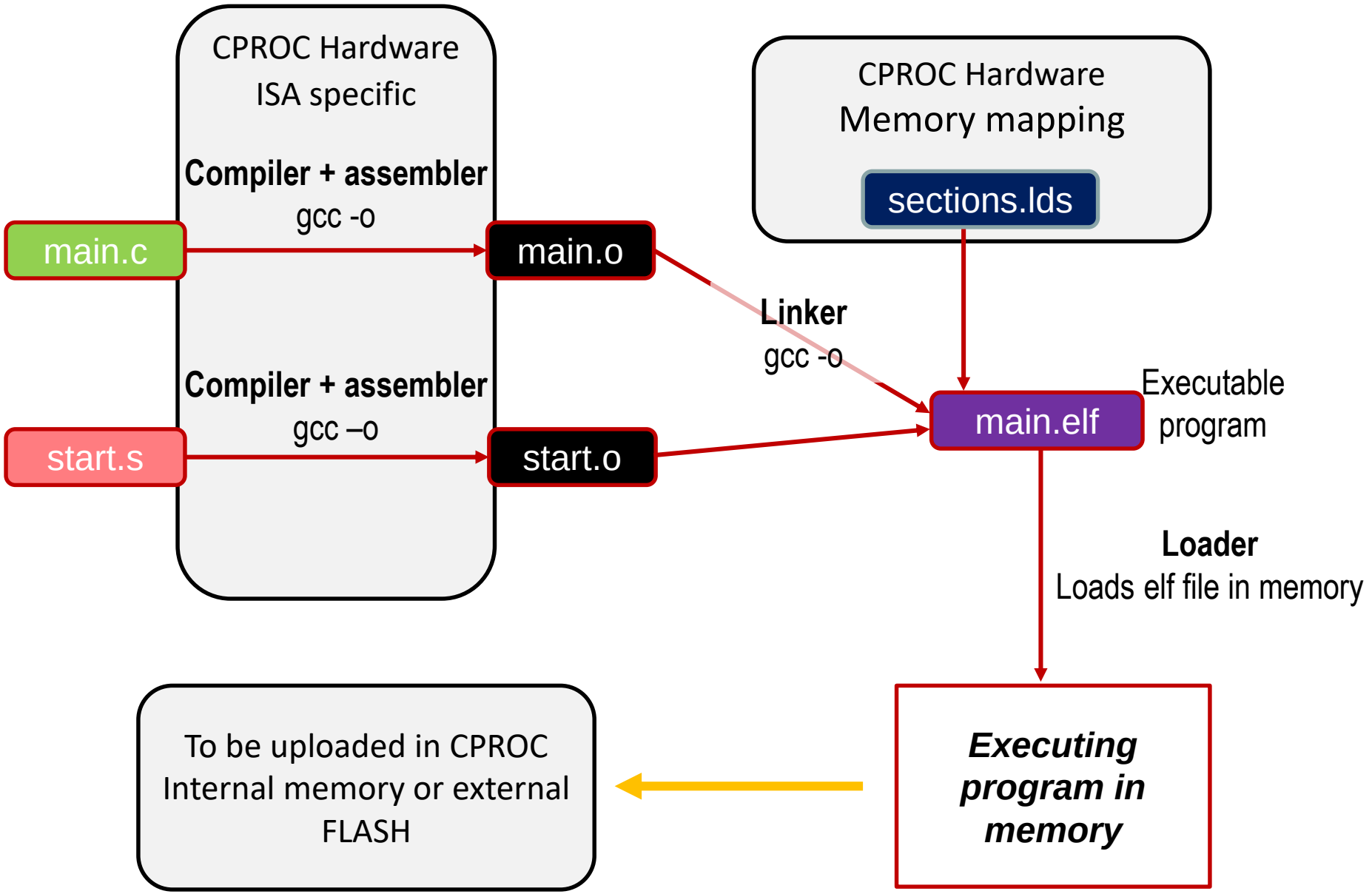
0: I2C auto test
1: CPROC RISCV autotest
2: Board auto test
3: Flash auto test
4: Memory checks
0: main menu

Compiling flow

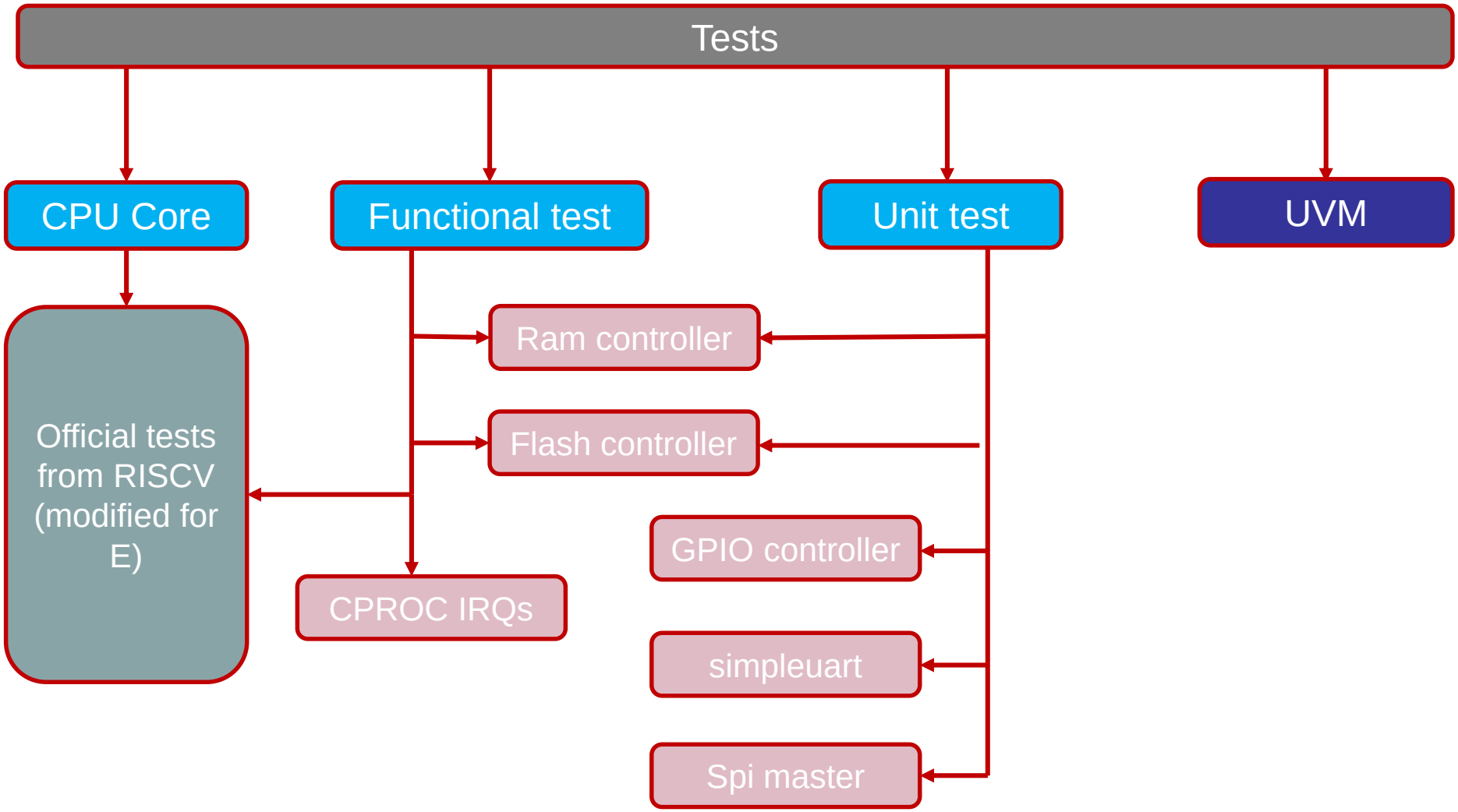


Compilation flow of a small piece of code, RV32IM compiler

CPROC compiling flow



Testing CPROC

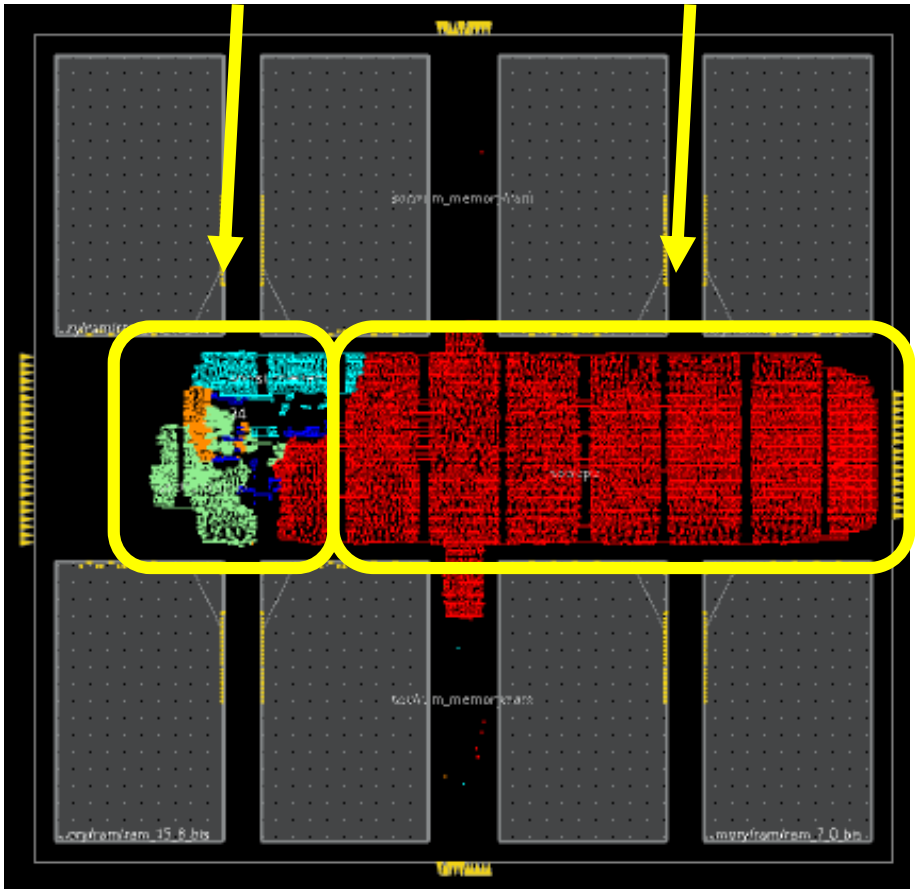


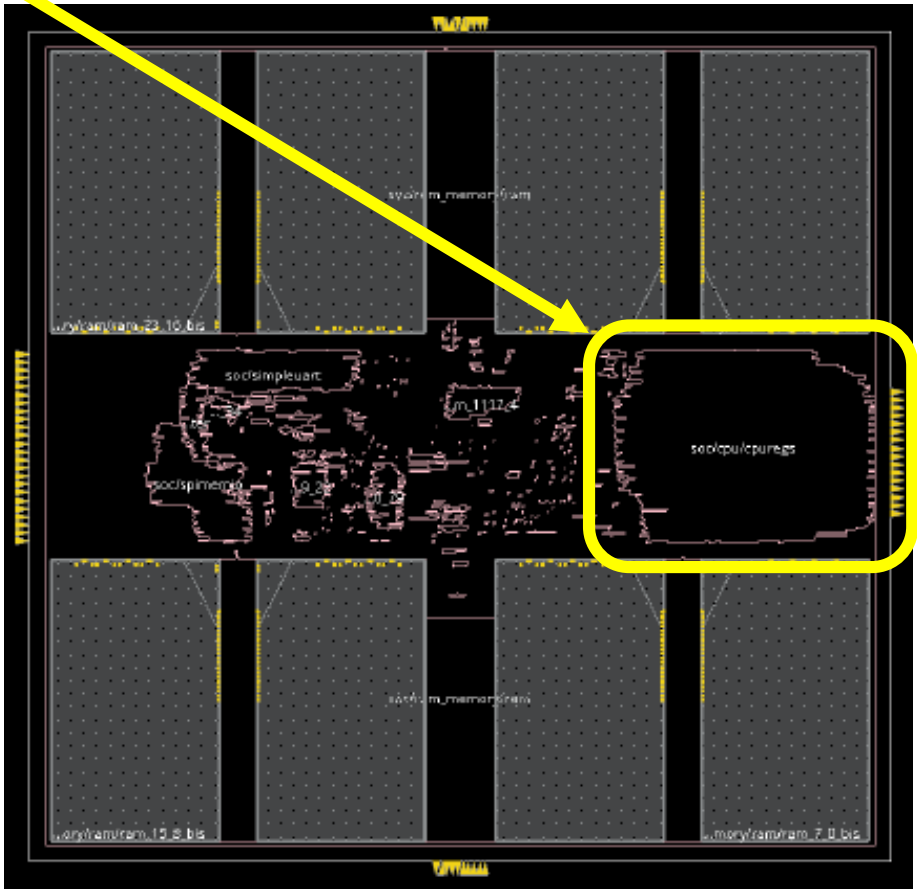
CPROC Amoeba view

Internal peripherals

CPU

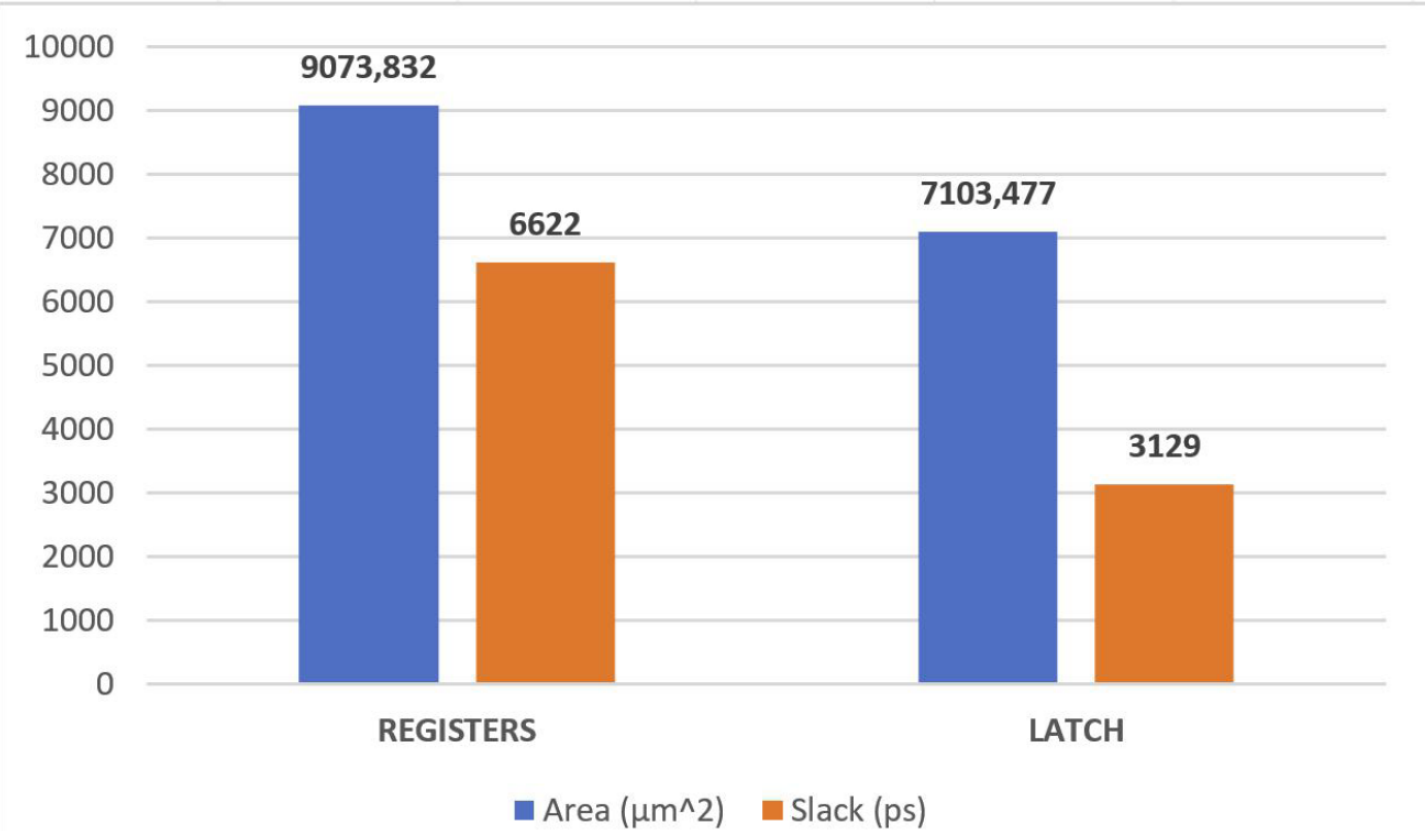
Register file





CPU configuration – Register file using latches

- Only the 8x16 register file has been synthetized, 100 MHz
  - Cadence latch and register memory IPs
- 27% gain in area, two times less slack



8x16 register file using registers and latches

Conclusion

- Application specific RISC-V : need to be optimized for your application / project
  - Architecture: memories shared or not
  - CPI: pipelined or not
  - ISA
- Mix of software / hardware / gcc project…
- Code quality of open source projects
- Full verification versus functional one

CPROC 2023 schedule:
- Testboard: July
- Test program : September
- CPROC : October