# Pipeline Datapath Models from RISC-V based cores

Samira Ait Bensaid, Mihail Asavoae, Farhat Thabet and Mathieu Jan
Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
*samira.aitbensaid@cea.fr, mihail.asavoae@cea.fr, farhat.thabet@cea.fr, mathieu.jan@cea.fr*

## Motivation

- Safety-critical systems rely on the use of timing analyses under architecture considerations to estimate Worst-Case Execution Times (WCET).
- Such architecture models generally built by hand in WCET analyzers, while using the open hardware frameworks [2] its automation could be possible.
- Generating hardware models for timing analysis have Verilog/VHDL designs [3]. However, hardware designers tend to use higher-level and more expressive languages, such as Chisel.

## Contributions

- Automatic construction of datapath pipeline model from high-level hardware designs [1].
- Evaluation of the approach on in-order RISC-V processors.

## Experimental results on RISC-V processor designs

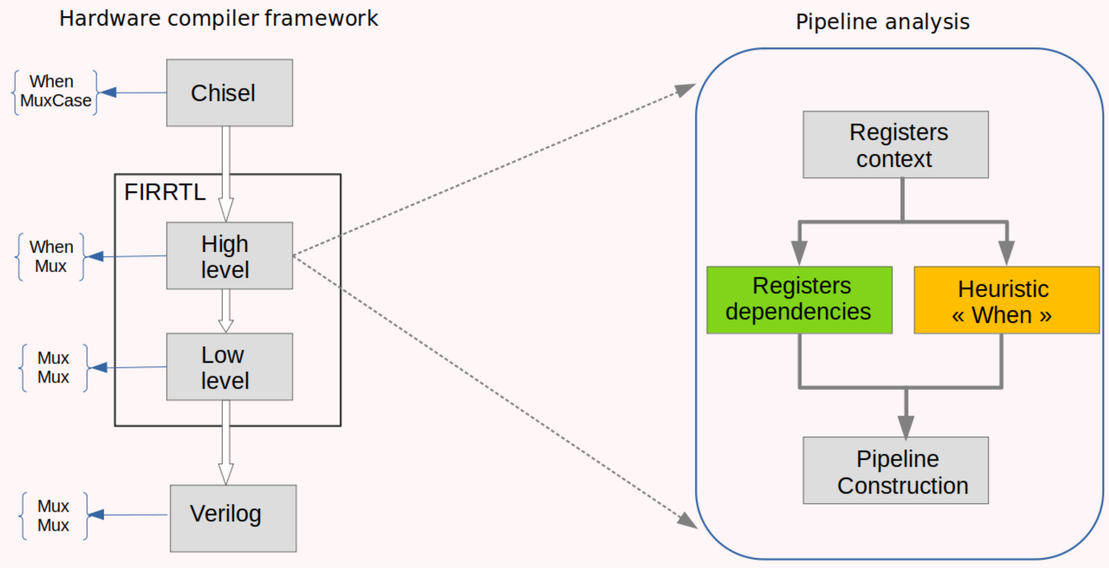
- An application of the analysis on a set of in-order RISC-V processors: 3 to 5 stages.

| | LOC | $\#Regs$ | Rule 1 | Rule 2 | $\#Stage$ |
|---|---|---|---|---|---|
| RISC-V Mini | 241 | 15 | 5 | 10 | 3 |
| Sodor | 646 | 48 | 34 | 14 | 5 |
| KyogenRV | 4567 | 93 | 47 | 36 | 5 |

## Perspectives

- Develop an extended analysis for multi-modular datapath pipelines and out-of-order processors.
- Generate the abstract formal models to verify timing properties.

## Pipeline datapath analysis



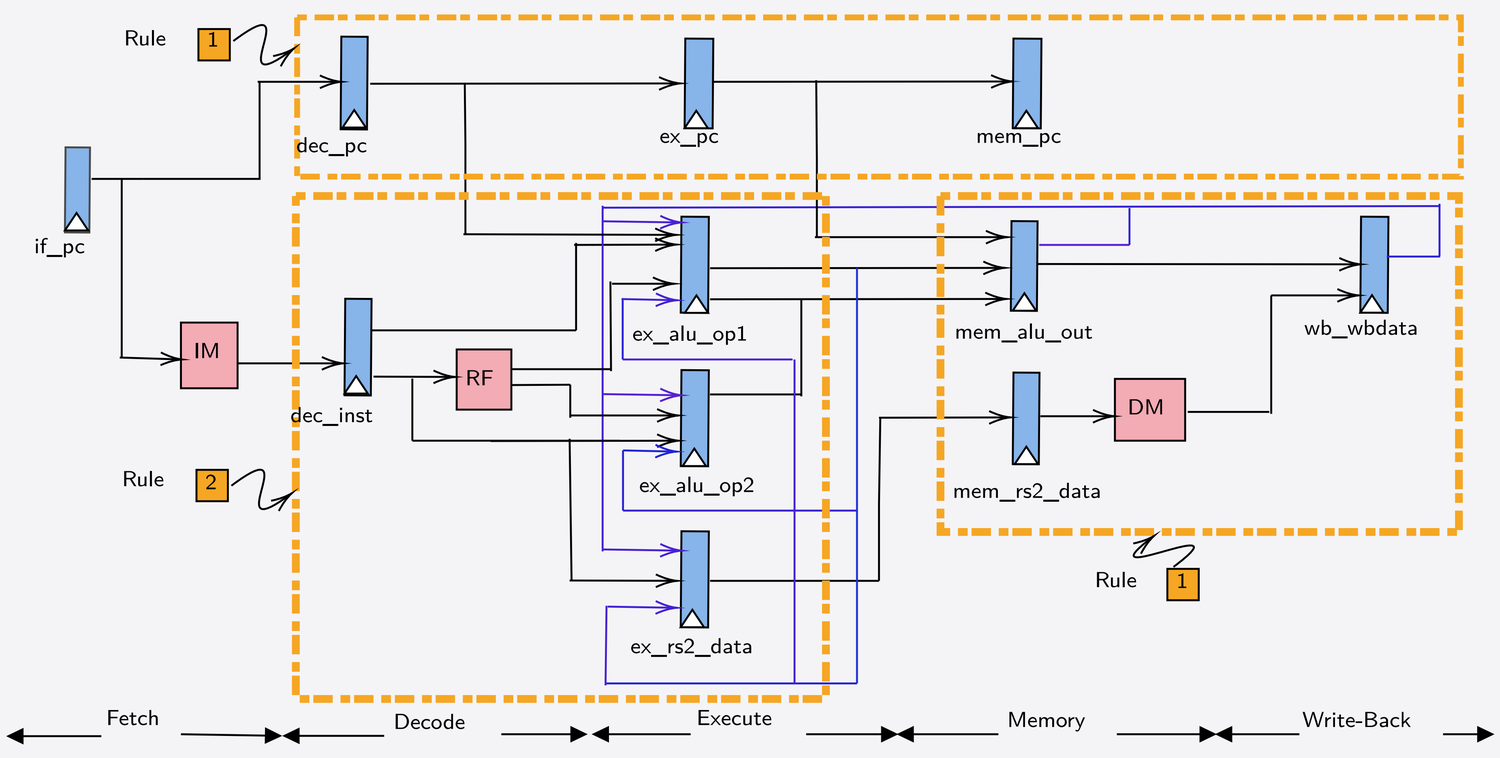

- Pipeline analysis algorithm:
  - Identified the processor registers.
  - Explore the combinatorial and sequential logics to build the registers context.
  - Build the dependency relations between registers.
  - Assign to each register its pipeline stage based on two rules:
    - Rule 1: register dependencies.
    - Rule 2: based on a heuristic "when" conditional block.
  - Deduce the pipeline depth and construct the pipeline datapath model.

Listing 1: Chisel registers updates in when conditional block

```
1  Class DatPath :
2
3  val dec_pc = RegInit (size)
4  val exe_pc = RegInit (size)
5  val exe_rs2_data = Reg (size)
6  val dec_rs2_data = Wire (size)
7
8  when (C4) {
9    exe_pc := dec_pc
10   exe_rs2_data := dec_rs2_data
11 }
```

## Illustration on RISC-V SODOR 5-stages processor

- RISC-V Sodor [4] processor is Chisel [5] based processor and we consider its 5-stage version.




| Reg | Rule | $\#Stage$ |
|---|---|---|
| if_pc | - | 1 |
| dec_pc | 1 | 2 |
| ex_pc | 1 | 3 |
| mem_pc | 1 | 4 |
| dec_inst | 2 | 2 |
| ex_alu_op1 | 2 | 3 |
| ex_alu_op2 | 2 | 3 |
| ex_rs2_data | 2 | 3 |
| mem_alu_out | 1 | 4 |
| mem_rs2_data | 1 | 4 |
| wb_wbdata | 1 | 5 |

## Bibliographie

[1] Bensaid, S.A., Asavoae, M., Thabet, F., Jan, M.: WiP: Automatic Construction of Pipeline Datapaths from High-Level HDL Code. In: RTASS (2022).
[2] Izraelevitz, A.M., Koenig, J., Li, P., Lin, R., Wang, A., Magyar, A., Kim, D., Schmidt, C., Markley, C., Lawson, J., Bachrach, J.: Reusability is FIRRTL ground: Hardware construction languages, compiler frame- works, and transformations. In: ICCAD. pp. 209–216 (2017).
[3] Charv at, L., Smrcka, A., Vojnar, T.: HADES: microprocessor hazard analysis via formal verification of parameterized systems. In: MEMICS. EPTCS, vol. 233, pp. 87–93 (2016).
[4] Risc-v sodor. https://github.com/ucb-bar/riscv-sodor.
[5] Bachrach, J., Vo, H., Richards, B., Lee, Y., Waterman, A., Avi ˇzienis, R., Wawrzynek, J., Asanovi c, K.: Chisel: Constructing hardware in a scala embedded language. In: DAC. p. 1216–1225 (2012).