Motivation. Certification of safety-critical systems requires the use of timing analyses to estimate Worst-Case Execution Times (WCET). These timing analyses reason about the executions of a program on an underlying computer architecture. WCET analyzers (such as [4], [7], [12]) use architecture models, generally built by hand, and static analysis to estimate these WCETs. Generating hardware models have mainly focused on functional verification of Verilog/VHDL designs ([6], [8], [10], [11]) and not for such WCET analyses, except [14]. However, hardware designers tend to use higher-level and more expressive languages, such as Chisel [3] or SpinalHDL for instance. There is thus a need for a fully automated construction of (abstract) datapath pipeline models from these higher-level hardware construction languages.

Contributions. We currently design a custom FIRRTL [9] pass to automate the construction of datapath pipeline models from Chisel [3] HDL processor designs. This pass focuses on the registers involved in the pipeline datapath, determines its depth and generates a model of it (see Fig. 1 for an example). We explore both the combinatorial and sequential logics of a pipeline in order to extract dependency relations between registers. Next, we assign a pipeline stage to the identified registers using two different rules. Rule 1 relies on register dependencies and takes into account forwarding mechanisms within micro-architectures. Rule 2 relies on a heuristic by taking advantage of a common practice of hardware designers to simultaneously update registers of a same pipeline stage within a same conditional block. This procedure is described in more details in [5]. Specific abstractions can then be applied to match input (timing) models used by WCET analyzers.

Results. The following table reports the effectiveness of our pass on a set of in-order RISC-V processors, ranging from 3 to 5 pipeline stages. The first column describes the code size of each (datapath) pipeline, the next column presents the number of registers (#Regs) and the last two columns summarize the number of registers successfully placed by each rule. For each processor, the depth of each pipeline is correctly computed and Fig 1 illustrates a subset of the datapath pipeline model of RISC-V Sodor. Starting from a given register located in the fetch stage (e.g. if_pc), our pass assigns registers dec_pc, ex_pc and mem_pc to respectively decode, execute and memory pipeline stages thanks to rule 1. Then, it assigns register dec_inst using rule 2, as it is updated in the same context as dec_pc. The procedure is then recursively applied to the next registers (exalu_op1, etc.) and identifies forwarding mechanisms represented by the blue edges on Fig 1.

REFERENCES