4th national RISC-V student contest 2023-2024

Sponsored by Thales, the GDR SOC² and the CNFM

Guidelines to report results

This document is a complement to the **`Annonce RISC-V contest 2023-2024 v2.pdf`** file to prepare the submission of results.

Prerequisites

The following steps need to be successful before analyzing the performance and starting your optimizations:

- The kit available at https://github.com/thalesgroup/cva6-softcore-contest is running in your Linux environment.
- You can launch all targets of the Makefile as described in the **`README.md`** file.
- The simulation of the MNIST application runs to its end and returns a successful result: the '4' digit is recognized and the credence (probability) is 82.
- The MNIST application executes on the ZYBO Z7-20 board and reports the same result.

In addition, before working on your optimizations, you need to ensure that:

- You are using the same tool versions as in the **`README.md`** file.
- You get the same metrics with the reference design as the organization team (see below).

And finally, before reporting results for the contest, you need to ensure that:

- You fulfill all constraints described in the **`Annonce RISC-V contest 2023-2024 v2.pdf`** file.

Analyzing the results

In this year's competition, the goal is to improve the performance of a CV32A6-based processor running the MNIST handwriting recognition application by modifying CV32A6 architecture.

The number of cycles is reported at the end of the simulation or execution on FPGA:

On Zybo board :

```
Expected  = 4
Predicted = 4
Result : 1/1
credence: 82
image env0003: 1731593 instructions
image env0003: 2354247 cycles
```

On simulation:

```
Expected  = 4
Predicted = 4
Result : 1/1
credence: 82
image env0003: 1731593 instructions
image env0003: 2316644 cycles
```

You can determine the maximal clock frequency of your design with the static timing analysis report after the place & route:

```
Timing Report
Slack (MET) : 0.600ns  (required time - arrival time)
Source:  i_ariane/i_cva6/issue_stage_i/i_issue_read_operands/alu_valid_q_reg_rep/C
(rising edge-triggered cell FDCE clocked by clk_out1_xlnx_clk_gen  {rise@0.000ns fall@10.000ns period=20.000ns})
Destination:            i_ariane/i_cva6/gen_cache_wt.i_cache_subsystem/i_cva6_icache/gen_sram[0].tag_sram/gen_cut[0].i_tc_sram
_wrapper/i_ram/Mem_DP_reg/ADDRARDADDR[11]
(rising edge-triggered cell RAMB36E1 clocked by clk_out1_xlnx_clk_gen  {rise@0.000ns fall@10.000ns period=20.000ns})
```

The maximal clock period is 1/(clock_period-slack), i.e. $1/(20.000*10^{-9}-0.600*10^{-9}) = 51.54$ MHz in this illustration.

The simulation testbench and the FPGA top design are a bit different, mostly with respect to main memory access time. Therefore you can experience slightly different number of cycles between both environments. The jury will consider the **number of cycles on the Zybo** board to sort the results.

Reference

The reference project is the one you can find in https://github.com/thalesgroup/cva6-softcore-contest.

Before getting further in the contest, you have to check that it provides the same results on your Linux machine than on the organizers' machine:

- Number of cycles as reported at the end of the simulation: 2316644 cycles
- Actual clock period of the design: 20 ns
- Minimum clock period of the design as reported after place & route: 19.4ns
- FPGA resources as reported in corev_apu/fpga/report_cva6_fpga_impl/cva6_fpga.utilization.rpt:
    - 8582 LUTs
    - 4611 FFs
    - 16 RAM36

If the results with the reference project are not the same as above, you first need to check that you are using the same versions of the tools[1]. If you still have a difference of more than 1% after this, get in touch with the organizers.

You shall also get the '4' digit recognized with a credence of 82.

The organizers may update the repository to recognize more digits on the FPGA board and consolidate your results.

[1] The organizers know this rule cannot be strictly enforced because of IT constraints.

Reporting the results

In both the 6-page report and in your recorded video, you'll clearly report these results:

- Recognized digit(s) and their credence(s), before and after your optimizations
- Number of cycles in simulation before and after your optimizations (and the corresponding reduction in %)
- Number of cycles on the FPGA board before and after your optimizations (and the corresponding reduction in %)
- Maximal frequency of your design before and after your optimizations (and the corresponding change in %)
- Number of LUTs and flip-flops of CVA6 in your design, before and after your optimizations

The report shall be written as a scientific paper, presenting your approach, your solution and commenting your results. It should help the jury to assess your solution.

The jury will double check in their environment the results of the teams who claim the best results in their reports. This also ensures that results are compared based on the same tool versions. So you'll publish a modified GitHub repository, as a fork of https://github.com/thalesgroup/cva6-softcore-contest. You'll create a **`report`** folder where you will upload:

- Your 6-page report written as a scientific paper
- Simulation log (uart)
- P&R report with the maximum frequency (corev_apu/fpga/report_cva6_fpga_impl/cva6_fpga.timing.rpt)
- Report of resources (corev_apu/fpga/report_cva6_fpga_impl/cva6_fpga.utilization.rpt)
- Log of the execution on the FPGA board (copy or screenshot of the hyperterminal output)

If you modify the source code to exploit your architectural changes (while fulfilling the constraints of **`Annonce RISC-V contest 2023-2024 v2.pdf`**), make sure you clearly comment your modification so that the jury can understand them.

Remember that your design shall run on the FPGA board and fulfill all constraints listed in **`Annonce RISC-V contest 2023-2024 v2.pdf`**. The returned result shall be the '4' digit with a credence of 82, both in simulation and when executed on the Zybo board.

During your recorded video, consider showing the execution of your solution on the Zybo board or executions on your machine. Because of the file size, do not upload the video in GitHub; you'll later get instructions for the file transfer.

A few remarks

To speed up the simulation, several optimizations are used: the application runs as baremetal (no OS used); the image to recognize is already stored in the memory at start up.

We suggest that you keep your GitHub repository private until the end of the contest (May 13th, 2024) and make it public just after. We'll look at the file timestamps to ensure you completed the contest on time.

Your presentation video shall last at most 10 minutes. All team members are encouraged to participate. The deadline to submit your video, about one week after, will be announced by the organizers. There is no special prize for the video, but it should be good enough to be presented to a large audience if your time is a winner.

A final word

If you find the contest difficult, remember that the other teams face the same level of difficulty. Even if you think your results are limited, submit them!

Version

These guidelines can undergo some evolutions based upon the teams' feedback. Please check their final version before submitting your results.

Version	Date	Comment
V0	2024-01-09	Initial version