Moodle & Verilator make open-source, web-based automated marking of SystemVerilog labs a reality

Automatically Assessing SystemVerilog Using Moodle

Background: In 2023 the School of Electronics and Computer Science at the University of Southampton began experimenting with Moodle for a new VLE. It was used in the 2024/25 academic year for a new Computer Engineering undergraduate programme, which includes a Digital Computer Systems module with significant SystemVerilog content. Students received one introductory SystemVerilog lecture before undertaking three assessed laboratories and a CPU design coursework. We used Moodle with the CodeRunner plugin to simulate and mark student-submitted SystemVerilog during the assessed laboratories. This poster presents some of our experiences.

Implementation: SystemVerilog support is implemented as a custom language within Jobe, a job engine for sandboxed remote execution of student-submitted code. A Jobe job specifies the language, execution environment, source code, and optional support files. For SystemVerilog, Verilator is used for execution.
At least two source files are required: the module under test and a testbench. One is provided by the instructor and the other by the student. Additional files, such as memory initialisation and sub-modules, can be submitted by either the instructor or student as appropriate Submitted files are then compiled and run by Verilator with the output filtered to remove the “finish" log. Filtered output is then returned to CodeRunner for marking and display to the student.

Assessment: Students are presented with a text box in their browser to enter code as shown in Figure 1. More complex assignments can allow the submission of additional source files. The output of the instructor-supplied testbench, or module, is presented below the submission once a student clicks “check”.

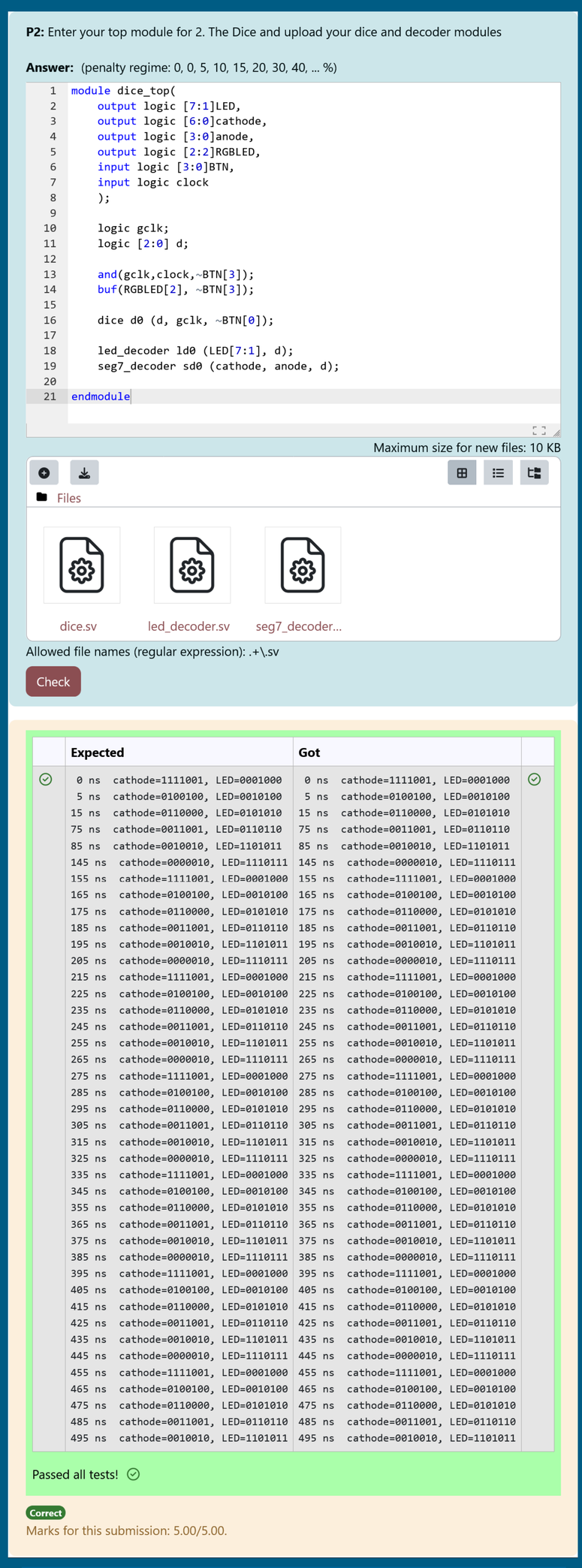

Figure 1: Student SystemVeriolog submission with supporting files showing graded testbench output.

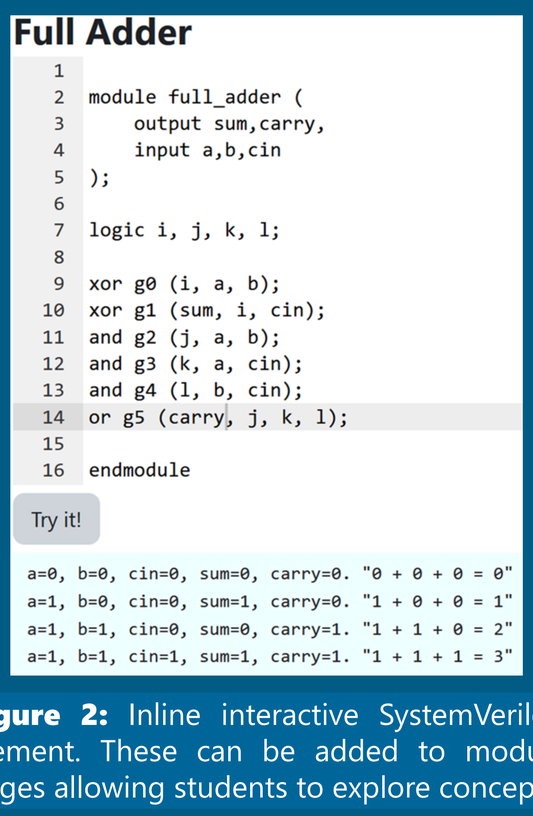

Figure 2: Inline interactive SystemVerilog element. These can be added to module pages allowing students to explore concepts.

Teaching: CodeRunner can also be used for interactive teaching. Inline interactive code elements (Figure 2) can be added to module webpages, allowing students to experiment with SystemVerilog without individual setup overhead.

Limitations & Future Work: Current marking is all-or-nothing: an incorrect answer results in zero marks. Improved marking capabilities are planned. Simulation output is currently limited to the output of `$display` and `$write` tasks. Other interactive approaches, such as the Cambridge SystemVerilog Tutor, present simulation waveforms. We would like to implement similar functionality.

Graeme M. Bragg, Jules Field
{gmb | jules}@ecs.soton.ac.uk
University of Southampton