VELS: VHDL E-Learning System

**Martin Mosbeck**, Daniel Hauer, Axel Jantsch

Institute of Computer Technology,
TU Wien, Vienna, Austria

NorCAS 2018, Tallinn, 2018-10-31
What is VELS?

- VHDL E-Learning System
- Automatic task generation, assessment, feedback
- Communication via email
- Open source @ GitHub
Why VELS?

- Large classes (250 students/semester)

- Benefits for students:
  - practice with adequate tasks
  - frequent, instant feedback
  - freedom of space, pace, study time scheduling

- Benefits for teachers:
  - framework to develop, deploy tasks
  - automatic assessment
  - progress monitoring
Key features

- Uniform task system
- Parameterized tasks
- New test for each submission
- Simulator interface
- Communication student < -- > VELS via email
Autosub submission system & task system

- Multi-threaded python daemon
- Periodically fetches new emails
Task request

- Request generates random variant of task
- Sent to user as email
Example task - Finite State Machine

Mealy FSM (Finite State Machine)

Your task is to program the behavior of an entity called "fsm". This entity is declared in the attached file "fsm.vhdl" and has the following properties:

- Input: CLK with type std_logic
- Input: RST with type std_logic
- Input: INPUT with type std_logic_vector of length 2
- Output: OUTPUT with type std_logic_vector of length 2
- Output: STATE with type fsm.state

Do not change the file "fsm.vhdl"!

The "fsm" entity shall have the behavior of a deterministic MEALY finite state machine with the following constraints:

- Behavior according to state diagram as shown in Figure 1. The input is transit to the next state and the output in the next state is labelled on the edges.
- Transitions with rising edge of the clock (port CLK, this is a square-wave clock signal).
- Synchronous design: all new outputs (port STATE and port OUTPUT) have to be set with rising edges of the clock.
- The current state has to be output at the output port STATE.
- At synchronous reset (port RST=1, rising edge of clock port CLK) the state machine is set to an initial state: STATE ← START, OUTPUT ← 00

This behavior has to be programmed in the attached file "fsm.beh.vhdl".

The type fsm.state is declared in the attached "fsm.package.vhdl" file. The needed package to use this type is already imported in the attached "fsm.beh.vhdl" and "fsm.vhdl".

Figure 1: State diagram, the edges are labelled input / next output
Task submission

1) Email from user
   *Result Task <N>*

2) Query
   *UserTasks table*
   *task parameters*

3) User task folder
   *behavior.vhdl*
   *entity.vhdl*
   *testbench.vhdl*

4) Simulator

5) Email to user
   *Failure Task <N>*
   Errors: ....

   Email to user
   *Success Task <N>*
### VELS Configuration

#### Task Statistics

<table>
<thead>
<tr>
<th>Task Nr</th>
<th>Task Start</th>
<th>Task Deadline</th>
<th>Task Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2018-09-18 16:00</td>
<td>2018-10-10 16:19</td>
<td>gates</td>
</tr>
<tr>
<td>2</td>
<td>2018-09-18 16:00</td>
<td>2018-10-10 16:19</td>
<td>demux</td>
</tr>
<tr>
<td>3</td>
<td>2018-09-18 16:00</td>
<td>2018-10-15 16:19</td>
<td>counter</td>
</tr>
<tr>
<td>4</td>
<td>2018-09-18 16:00</td>
<td>2018-10-17 16:19</td>
<td>fsm_mealy</td>
</tr>
<tr>
<td>5</td>
<td>2018-09-18 16:00</td>
<td>2018-10-17 16:19</td>
<td>ROM</td>
</tr>
<tr>
<td>6</td>
<td>2018-09-18 16:00</td>
<td>2018-10-17 16:19</td>
<td>RAM</td>
</tr>
<tr>
<td>7</td>
<td>2018-09-18 16:00</td>
<td>2018-10-17 16:19</td>
<td>pwm</td>
</tr>
<tr>
<td>8</td>
<td>2018-09-18 16:00</td>
<td>2018-10-17 16:19</td>
<td>arithmetic</td>
</tr>
</tbody>
</table>

#### Statistics Counters

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>nr-mails_fetched</td>
<td>2694</td>
</tr>
<tr>
<td>nr-mails_sent</td>
<td>2878</td>
</tr>
<tr>
<td>nr_non_registered</td>
<td>74</td>
</tr>
<tr>
<td>nr_questions_received</td>
<td>16</td>
</tr>
<tr>
<td>nr_status_requests</td>
<td>45</td>
</tr>
</tbody>
</table>

- Web interface
- Configure tasks, whitelist
- Monitor progress

**Number of received emails**

![Graph showing the number of received emails over time.]
Task generator tool

Allows to focus on:

- task description
- task variant generation
- testbench generation
Deployment & Results (1/2)

- 3 years, 2 courses/each winter semester
- Overall 1000 students

Mandatory, continuous learning in the graduate course „Digital Integrated Systems“ (50 students)
Deployment & Results (2/2)

<table>
<thead>
<tr>
<th>Task nr.</th>
<th>Sum</th>
<th>Students</th>
<th>Rate</th>
<th>Submissions</th>
<th>Passed</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>242</td>
<td>195</td>
<td>81 %</td>
<td>1215</td>
<td>317</td>
<td>26 %</td>
</tr>
<tr>
<td>2</td>
<td>242</td>
<td>185</td>
<td>76 %</td>
<td>905</td>
<td>326</td>
<td>36 %</td>
</tr>
<tr>
<td>3</td>
<td>242</td>
<td>188</td>
<td>78 %</td>
<td>623</td>
<td>294</td>
<td>47 %</td>
</tr>
<tr>
<td>4</td>
<td>242</td>
<td>176</td>
<td>73 %</td>
<td>1967</td>
<td>313</td>
<td>15 %</td>
</tr>
<tr>
<td>5</td>
<td>242</td>
<td>160</td>
<td>66 %</td>
<td>1318</td>
<td>240</td>
<td>18 %</td>
</tr>
<tr>
<td>6</td>
<td>242</td>
<td>84</td>
<td>35 %</td>
<td>1411</td>
<td>115</td>
<td>8 %</td>
</tr>
<tr>
<td>7</td>
<td>242</td>
<td>147</td>
<td>61 %</td>
<td>795</td>
<td>247</td>
<td>31 %</td>
</tr>
<tr>
<td>8</td>
<td>242</td>
<td>80</td>
<td>33 %</td>
<td>1805</td>
<td>141</td>
<td>7 %</td>
</tr>
</tbody>
</table>

Free practice in undergraduate course “Microcomputer”
Conclusions & take away

- 3 years use, open source @ [GitHub](https://github.com/autosub-team/autosub)
- Allows continuous, incremental learning
- Human teaching still needed
- Clear task specification & comprehensive feedback!
- Synthesis tool interface
- Moodle interface, bundle as Debian package
Questions?