Week 3

Codefest: Decide the SW/HW boundary, pick tools, code toy example

ECE 410/510
Spring 2025

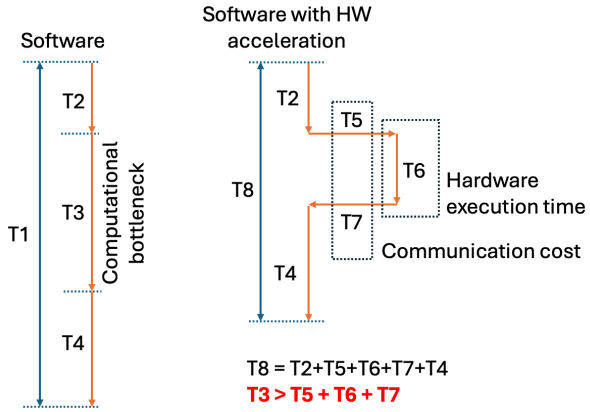
Challenge #12

Learning goals:

- Complete an estimate of the execution and communication times.
- Decide what parts stays in SW and what part becomes HW.
- Evaluate and pick a high-level tool for rapid prototyping and co-design.
- Implement a toy example to test the entire design workflow. E.g., FrozenLake RL code in which you implement the Q value updating formula in hardware.

Suggested tasks:

1. Before you complete the tasks below, you should benchmark and profile the algorithm that you chose. See codefest #2, challenge#9 for suggested tasks, tools, etc.
2. Based on your initial benchmarking and profiling of your code, make a first informed decision what part(s) should be accelerated in HW.
3. You may want to do a back-of-the-envelope calculation to see if the HW acceleration is worth the effort. To do so, determine the following times:



4. The HW acceleration is only worth it if T5 + T6 + T7 < T3.
5. Next, based on your background and interests, pick one of the high-level tools listed below that will allow you to do rapid HW prototyping in Python.
   - **PyMTL (Mamba):**
     - https://pymtl.github.io
     - User group: https://groups.google.com/g/pymtl-users
     - An open-source hardware modeling, generation, simulation, and verification framework.
     - MyHDL allows a subset of Python code to be translated to Verilog or VHDL. It offers co-simulation options where native Python code runs alongside a compiled simulation model of your hardware.
     - Hardware-software co-simulation using PyMTL3:
       - Create your hardware model in PyMTL3
       - Develop software that will interact with the hardware
       - Set up the co-simulation environment
       - Run and analyze results

- **PyRTL**:
  - https://ucsbarchlab.github.io/PyRTL
  - PyRTL is an open-source Python-based hardware development toolkit.
  - PyRTL provides a minimal set of hardware primitives, expressed as a Python class, which can then be extended with other classes and libraries as appropriate.
  - Allows for fast design iteration in a variety of domains, including cryptography and machine learning.
  - Paper:
    - Agile Hardware Development and Instrumentation with PyRTL
    - https://doi.org/10.1109/MM.2020.2997704

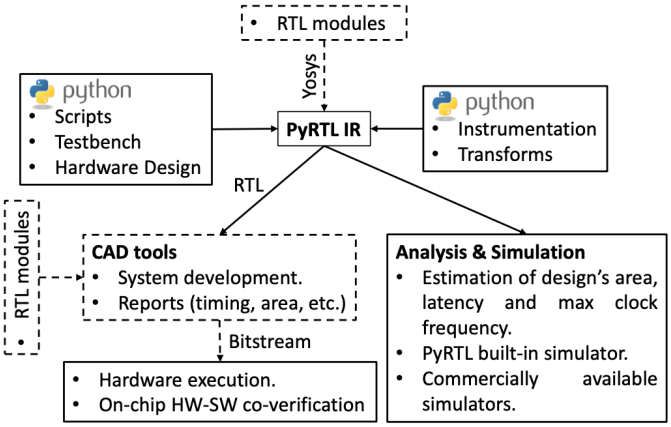


Fig. 1. Overview of PyRTL's flow.

- **Cocotb:**
  - https://www.cocotb.org
  - cocotb is an open source coroutine-based co-simulation testbench environment for verifying VHDL and SystemVerilog RTL using Python.
- **MyHDL:**
  - https://www.myhdl.org
  - MyHDL turns Python into a hardware description and verification language, providing hardware engineers with the power of the Python ecosystem.
  - MyHDL designs can be converted to Verilog or VHDL.
- **pyUVM:**
  - https://github.com/pyuvm/pyuvm
  - pyuvm is the Universal Verification Methodology (UVM) implemented in Python instead of SystemVerilog. pyuvm uses cocotb to interact with simulators and schedule simulation events.
- **yosys:**
  - https://github.com/YosysHQ/yosys
  - Yosys is a framework for RTL synthesis and more. It currently has extensive Verilog-2005 support and provides a basic set of synthesis algorithms for various application domains.

6. Implement a toy example to test the entire design workflow. E.g., FrozenLake RL code in which you implement the Q value updating formula in hardware (think adders + multipliers).
7. Once you know the tools will be right for you, start implementing your own HW design for your chosen algorithm.