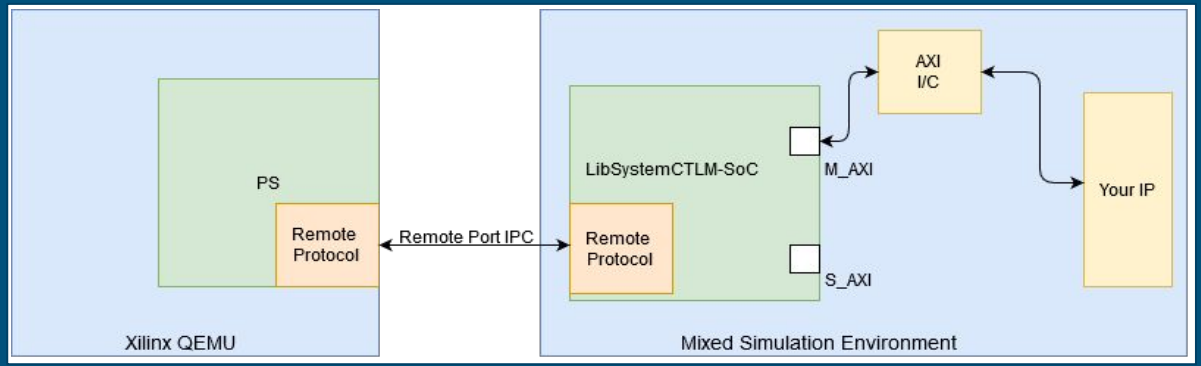
PIRM 1: Co-Simulation of an Avionics Device

SDDEC21-02: Matt Dwyer, Braedon Giblin, Cody Tomkins, Spencer Davis, & Prince Tshombe

Faculty Advisor: Dr. Phillip Jones
Client: Matthew Weber (Collin's Aerospace)
Website: https://sddec21-02.sd.ece.iastate.edu/

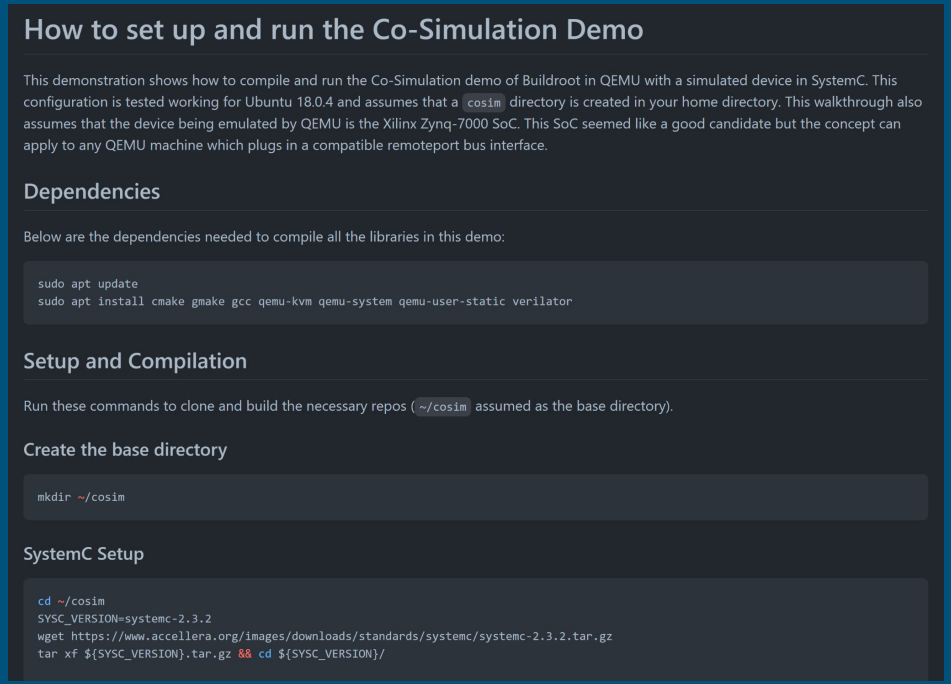
Hardware/Software Co-Simulation

- Simulate the processor your code is running on (Embedded ARM Cortex-A9)
  - Processor emulator (QEMU)
  - Buildroot Linux
- Simulate the hardware interactions and mock all calls made
  - Hardware implementation (SystemC)
  - Hardware transaction modeling (TLM)
- Connect the two simulated environments (FPGA PS-PL connection)
  - Xilinx Remote Port



Problem Statement

- Steep learning curve for beginners
  - Few documented example projects
  - Lacking basic documentation
- Desire for additional flexibility
  - Once the simulation has been setup, difficult to manipulate data “Generated” by simulated hardware
  - Desire to “feed” data into the system from an external source
  - Processing System (QEMU) being none the wiser, assumes it is a real device



How to set up and run the Co-Simulation Demo

This demonstration shows how to compile and run the Co-Simulation demo of Buildroot in QEMU with a simulated device in SystemC. This configuration is tested working for Ubuntu 18.0.4 and assumes that a `cosim` directory is created in your home directory. This walkthrough also assumes that the device being emulated by QEMU is the Xilinx Zynq-7000 SoC. This SoC seemed like a good candidate but the concept can apply to any QEMU machine which plugs in a compatible remoteport bus interface.

Dependencies

Below are the dependencies needed to compile all the libraries in this demo:

```
sudo apt update
sudo apt install cmake gmake gcc qemu-kvm qemu-system qemu-user-static verilator
```

Setup and Compilation

Run these commands to clone and build the necessary repos (`~/cosim` assumed as the base directory).

Create the base directory

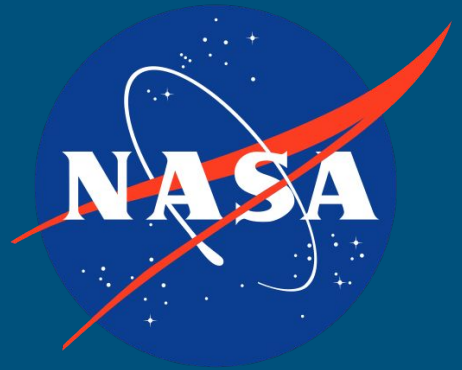
```
mkdir ~/cosim
```

SystemC Setup

```
cd ~/cosim
SYSC_VERSION=systemc-2.3.2
wget https://www.accellera.org/images/downloads/standards/systemc/systemc-2.3.2.tar.gz
tar xf ${SYSC_VERSION}.tar.gz && cd ${SYSC_VERSION}/
```

Intended Users

- Corporations who simultaneously develop hardware/software solutions
  - Aerospace, Defense, Industrial Automation, Automotive
- Users looking to extensively test hardware and software independently of one another
- People interested in applying Co-Simulation to their own project who are stymied by the barrier of entry



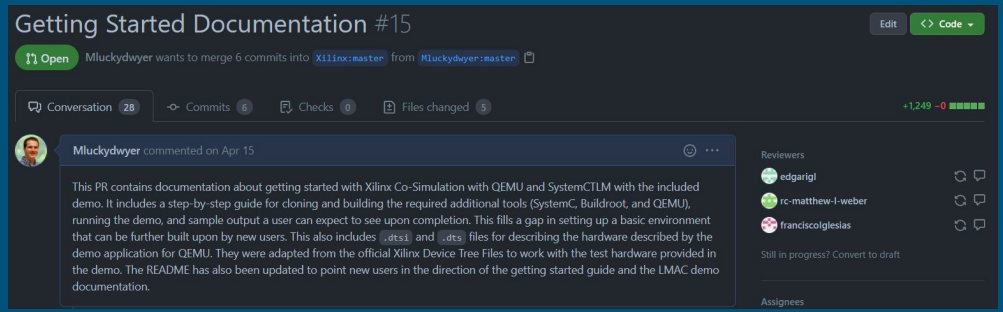
JOHN DEERE

Functional Requirements/Deliverables

- Documentation
  - Document an initial environment setup walkthrough → PR has been submitted, being revised
  - Create an additional demo to for a more complex system → PPM demo working, publishing soon
- External Data Source/Modeling Tool
  - Model an I^2C Bus in SystemC and corresponding test application → In progress -- HIGH RISK
  - Drive a simulated IMU device over I^2C with static data → In progress
  - Develop Remote port custom communication tunnel for external data source tool → Completed functional demo last semester
  - Demonstrate an off-the-shelf Linux IMU driver running on QEMU, working with modeled hardware → IMU and driver selected. Required for our final MVP demonstration

Open Sourced Goal

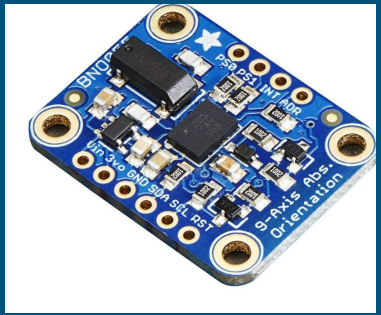
- Involve the open source community as much as possible
  - Work with existing repository maintainers to push work that adheres to their standards and their vision for the future of the repository
  - Make contributions that will be meaningful to future users
  - Ensure our demos and use cases are general enough so that any user of the repository in the future will find value in our work
- All of our changes should be merged into the Xilinx Cosim repository by the end of the semester



Highest Risk Todos

I^2C IMU Implementation

- Need to mock an I^2C Master device in SystemC that will be recognized by Linux
- Device timings need to match Linux driver expects
- Synchronization between controller and QEMU may be difficult



Libremoteport Data Input

- Libremoteport is a complex, poorly documented library
- It is not clear if we can have multiple ports driving the SystemC model
- Will likely need synchronization with the existing simulation

Future Goals

- Construct extendable implementation of SystemC external connection interface
- Provide in depth example of IMU I^2C device being emulated with interface
- Create visualized/graphical demonstration of interface and IMU
- Author supporting documentation (Dockerfile, remote-port, examples)
- UART Implementation for Linux Serial System
- Document project thoroughly (Website, in-depth presentations, publish implementation source)