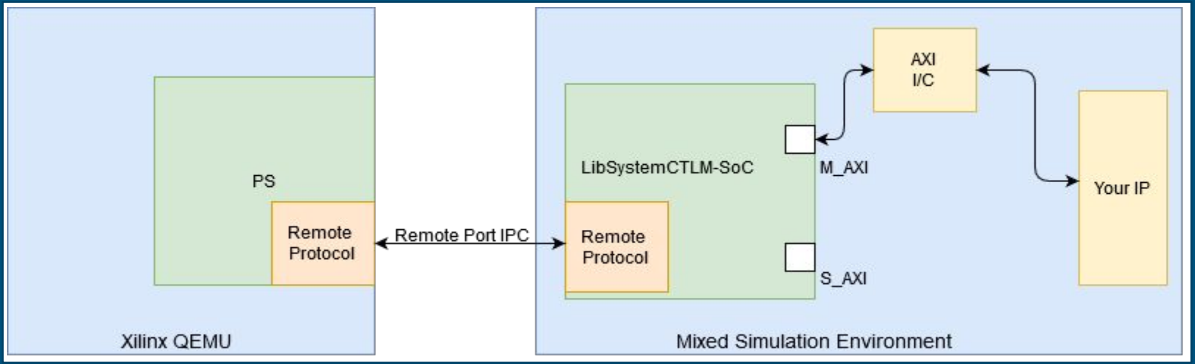
PIRM 2: Co-Simulation of an Avionics Device

SDDEC21-02: Matthew 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/

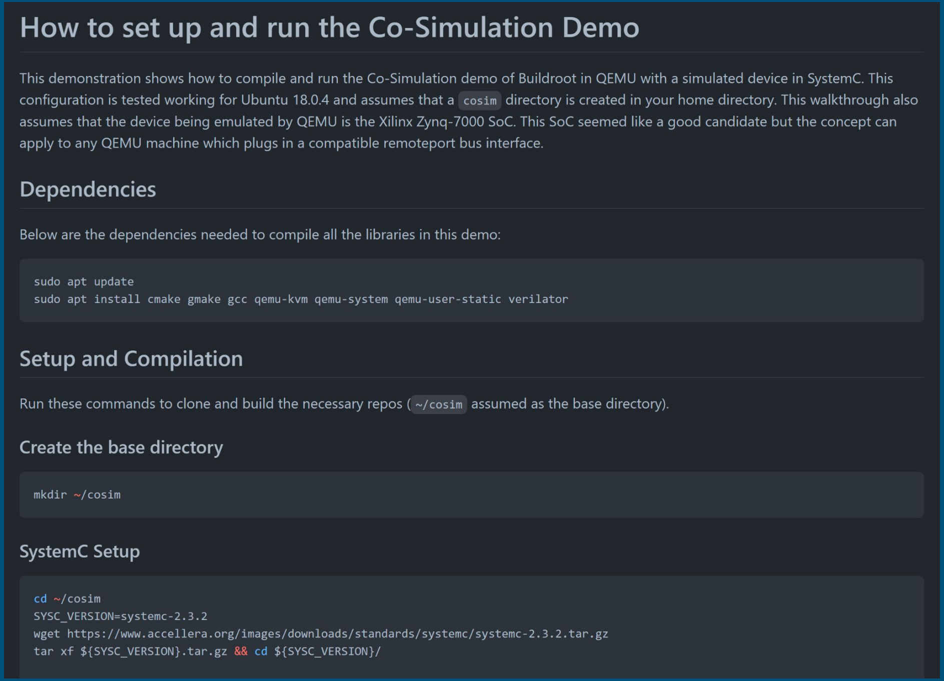
Hardware/Software Co-Simulation

- 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
  - Once the host data source disconnects from the system, the entire system stops



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
```

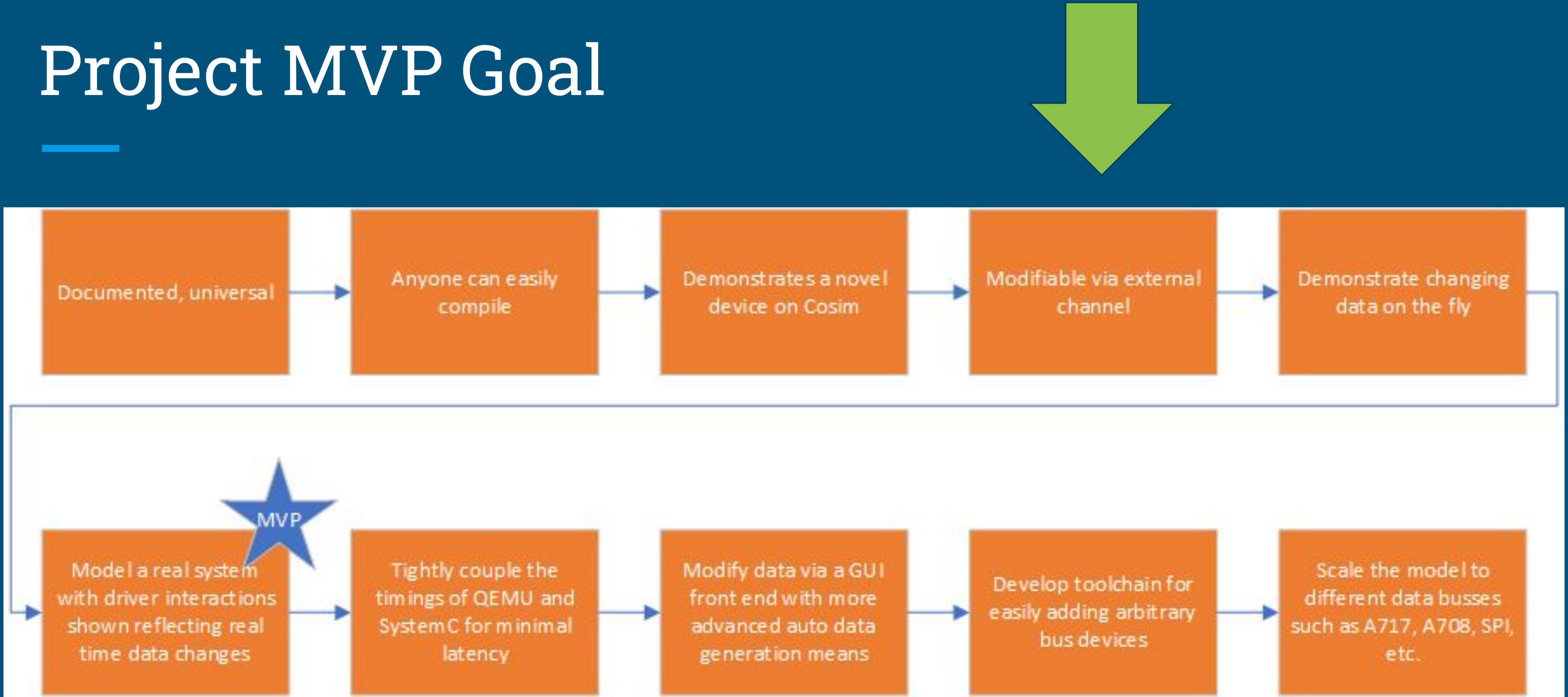
SystemC Setup

```
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}/
```

Functional Requirements/Deliverables

- Documentation
  - Document an initial environment setup walkthrough → PR has been merged into public repository
  - Create an additional demo to for a more complex system → PPM demo working
- External Data Source/Modeling Tool
  - Model an I^{2}C Bus in SystemC and corresponding test application → Planning State Machine, IIO driver interface
  - Drive a simulated IMU device over I^{2}C with static data → In progress, blocked by I2C Master Simulation
  - 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 → Kernel Modules compiled and inserted into Buildroot

Project MVP Goal



- Documented, universal
- Anyone can easily compile
- Demonstrates a novel device on Cosim
- Modifiable via external channel
- Demonstrate changing data on the fly
- MVP Model a real system with driver interactions shown reflecting real time data changes
- Tightly couple the timings of QEMU and SystemC for minimal latency
- Modify data via a GUI front end with more advanced auto data generation means
- Develop toolchain for easily adding arbitrary bus devices
- Scale the model to different data busses such as A717, A708, SPI, etc.

Technical Challenges

- Remote Port
  - Compiling against the SystemC & Xilinx SoC libraries
  - C++ Unix socket management
  - SystemC memory sockets
  - SystemC exceptions
- IMU Driver
  - IMU provides generic driver that does not define hardware interface
  - Hardware interface is currently being set up via an IIO Linux Driver we are compiling into buildroot
  - We have experienced technical challenges getting drivers compiled and added to buildroot (Soft float library linking, buildroot kernel module kconfig, etc.)

IMU Emulation Status

- Formalized how we intend to mock the IMU in our cosimulated system
- System C (In Progress)
  - State machine that will process read and write requests by storing to internal registers and responding to IO accesses
  - Read only data registers populated with sample data (later will be piped into remoteport backend)
- Buildroot
  - Determined how to add IMU test program into buildroot.
  - Program uses IMU driver to communicate with the IMU via IIO interface

Remote Port

- Robust build system with Xilinx libraries
- Customize additional demo to include additional port
- Correctly allocate device spaces and interfaces in SystemC device
- Create Unix socket for SystemC to connect to
- Initiate connection from SystemC side
- Bind the memory port in SystemC
- Wait for data in SystemC

Future Goals

- Construct extendable implementation of SystemC external connection interface (In progress, high risk)
- Provide in depth example of IMU I^2C device being emulated with interface (In progress, progressing quickly)
- Document project thoroughly (Website, in-depth presentations, publish implementation source)