Co-Simulation of an Avionics Interface Device

DESIGN DOCUMENT

Team SDDEC21-02
Mathew Weber – Collins Aerospace
Dr. Phillip Jones
Spencer Davis
Matt Dwyer
Braedon Giblin
Cody Tomkins
Prince Tshombe
sddec21-02@iastate.edu
https://sddec21-02.sd.ece.iastate.edu

Created: 03-07-2021/V1
Revised: 04-05-2021/V2

Executive Summary

Development Standards & Practices Used

In this project, we utilize many practices related to open-source software. Our project hinges on us utilizing open-source repositories, as well as expanding and contributing to these such projects as well. Our team also utilized AGILE-like development, where we utilized a KANBAN style task board to keep track of our ongoing and defined tasks.

Summary of Requirements

- Setup and execute a Cosim model using SystemC TLM backend and Xilinx QEMU processor simulator simultaneously
- Expand the Cosim capabilities by implementing bi-directional memory communication
- Model and test an off the shelf Linux driver for a memory-mapped peripheral

Applicable Courses from Iowa State University Curriculum

- CPR E 381
- CPR E 308

New Skills/Knowledge acquired that was not taught in courses

Our team acquired new knowledge of hardware simulation platforms including SystemC TLM and Xilinx QEMU. Our team also explored Linux driver testing and gained insight in how the driver may interface with a memory mapped peripheral.

Table of Contents

1 Introduction

1.1 Acknowledgement

We would like to give a big thanks to our client Matthew Weber for providing us with the technologies we need and for his technical help and patience throughout this project. We would also like to thank Dr. Phillip Jones for giving us technical advice and helping us solve problems we have had throughout this project. This project wouldn't be as well done without them.

1.2 Problem and Project Statement

Problem: The co-simulation environment that exists using Xilinx QEMU (Quick Emulator) in conjunction with Xilinx SystemC TLM libraries lacks good technical demonstrations and documentation that will make using this software chain easier.

Solution: This project's goal is to create demos and simulations that can be documented and used as examples for future users of this software. The primary demo our team will be building will feature an arbitrary Linux driver running in QEMU simulation, with a SystemC backend capable of communicating bidirectionally with the host PC. An application hosted on the host PC can then interact with the SystemC backend, driving the backing registers of the linux device. We will complete this demo with a comparison between the co-simulation environment and a traditional QEMU simulation, explaining where benefits can be seen with cosim as opposed to current simulation strategies.

1.3 Operational Environment

The project will be created in a Linux environment. We are using an Ubuntu 18.3.4 server. This version of Linux was selected based on prior cosim demos, ensuring compatibility with all required tools. Our project will be simulating hardware, so we do not need any special hardware for this design.

This is an open source project, so we will be working with the project community, where we can ask questions and get feedback from the project creators. This means we must abide by their standards for coding design and documentation.

1.4 Requirements

1. Identify an off-the-shelf Linux driver for an I ²C IMU device
2. Bi-directional communication between the SystemC model and the host PC.
    a. Communication must allow multiple devices to be modified by the front end
    b. Must support read/writes to registers synchronized with QEMU accesses
    c. Interface should be configurable and scalable
3. Front end application to manage the host PCs connection with SystemC backend
4. Documentation and demonstration of design, as well as a robust comparison between Cosim and previous simulation interfaces

1.5 Intended Users and Uses

Our intended users are corporations looking to utilize these toolflows for testing their products. For instance, the avionics community would be interested in these modeling chains as they can test software drivers prior to having novel avionics hardware designed and synthesized.

1.6 Assumptions and Limitations

We are assuming that:

- All source code will be published to an open source repository
- We can freely use all SystemC libraries to model our communication of interfaces

Some our limitations are:

- Some group members have little experience using a Linux based operating system and need to learn a lot of new material to be able to contribute
- The amount of current documentation of the system process is fairly limited.
- Our contributions and documentation will be constrained by what repository maintainers are interested in having in their projects.
- As of right now, the co-simulation programs do not utilize a convenient user-interface, which may make testing of our new code difficult.

1.7 Expected End Product and Deliverables

By the end of first semester:

1. The group should have the foundation of a bi-directional SystemC interface implemented with capabilities of sending and receiving data from the SystemC model
2. A Linux driver for an I²C IMU identified with a basic understanding of the expected backend behavior documented
3. Have a line of communication with open-source repository maintainers for demos we are working with
4. Documentation on initial Cosim demos generated and submitted to repo as a pull request.

2 Project Plan

2.1 Task Decomposition

This project consists of multiple tasks. Below is a list of those overarching tasks and some of the intricacies involved in each:

1. Initial Cosim demo and environment setup
    a. Setup a shared computing environment for all members to use collaboratively.

    b. Work through the initial demo provided by the client to learn the ropes of the tools as hand
    c. Explore the technologies (SystemC, QEMU, TLM) and how they interact with one-another in the simulated environment
2. Modifying the Demo
    a. Understanding how to modify the demo to add additional functionality or alter previous functionality of the timer register counter
    b. Implement the Threading demo provided by our client to further augment the initial Co-Simulation demo
    c. Better understand how all the software interacts and plan on how to add new demo features
    d. Document the process for running the demo for addition to the repository through pull-request submission
3. Reach out to public project maintainers about project direction
    a. What additions would be welcomed by the development teams utilizing the same tools?
    b. What other resources are available to aid in the contribution process?
    c. What areas are most in need of support and extension?
4. Implementing bi-directional communication of the host and the SystemC model
    a. Develop protocol for modifying any SystemC device data via communications from host
    b. Understand interactions between simulated hardware in SystemC, the simulated Linux software driver and OS in QEMU, and the input data from the host OS
    c. Implement a front end interface to be run on the host
    d. Demonstrate controllability of the simulation via the host communication interface
5. Identify and obtain a driver for an I ²C IMU and simulate the driver using Co-simulation augmented by host controlled communications.
    a. Identify an open-source Linux driver for IMUs use in demo
    b. Identify a common IMU for simulation
6. Document the additional demonstration in detail
    a. Record all steps to reproduce results from a beginner to intermediate experience level
    b. Receive public feedback from the development community surrounding similar demos and the utilized tools.
    c. Publish a final draft that is accepted for publication
    d. Provide easy handles for other developers to extend functionality of the demo and understand how to adapt it to their needs.
7. Repeat Process for similar ARINC 717 or UART in Linux Serial System
    a. Follow a similar design pattern utilizing the tools from previous parts to implement more complex standards and devices
    b. Explore additional sensors and testing use cases not thoroughly documented to build out additional testing capabilities
    c. Work off of additional client-side teams working in parallel on similar projects

2.2 Risks And Risk Management/Mitigation

The overall risk for this project is quite low. This is primarily due to the fact that it is entirely in software development and utilizing demos already freely provided online. The biggest risk factor that is foreseen is the poor reception and feedback of our contributions to the public projects. This could occur for a number of reasons such as poor maintainer support, unaligned goals for the future of the project, or already generated documentation and additions.

To mitigate this risk, we have worked with our client to develop another publication strategy if the primary public repositories do not favor our contributions. This would involve publishing our additional documentation and improvements on our own. Since each of the projects is open-source, meaning free to distribute and alter, there would be no licensing issues in doing this and while it would not be a part of the official documentation for the interacting projects, it would likely still contribute to the Co-Simulation development community as a whole.

In addition, when developing the low level test drivers for the UDP communication protocol and other devices, our client has been generous in providing support from professionals in that area. As such, we will likely struggle at first to generate those low level drivers, but with the support of the consultant provided, the majority of those risks should be mitigated.

Finally, documentation for the tools we are utilizing and developing are significantly scarcer than other public tools are due to their limited use. This is likely due to their poor documentation. As such, part of our project goals is to better the documentation provided and available for developers wishing to utilize these tools (QEMU + SystemCTLM Co-Simulation). This has the potential to be another significant risk due to the undocumented nature of the products planned, and the uncertinaty in our udnerstanding and planned use cases.

To mitigate this risk of unknown and undocmented tools, our team will be vigilant in documenting all tools and knowledge we gain along the way for our team and others. We will employ a fail-fast methodology of building prototypes and diving deep to try and understand what will be feasible and search for alternatives when outlooks and early attempts return poor results. We also plan to be aggressive in outreach and support. Seeking guidance from our client and his team of experts along with the development community of the tools at our disposal. With these resources we foresee a great ability to pivot to new innovative solutions should our plans need to be refactored.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

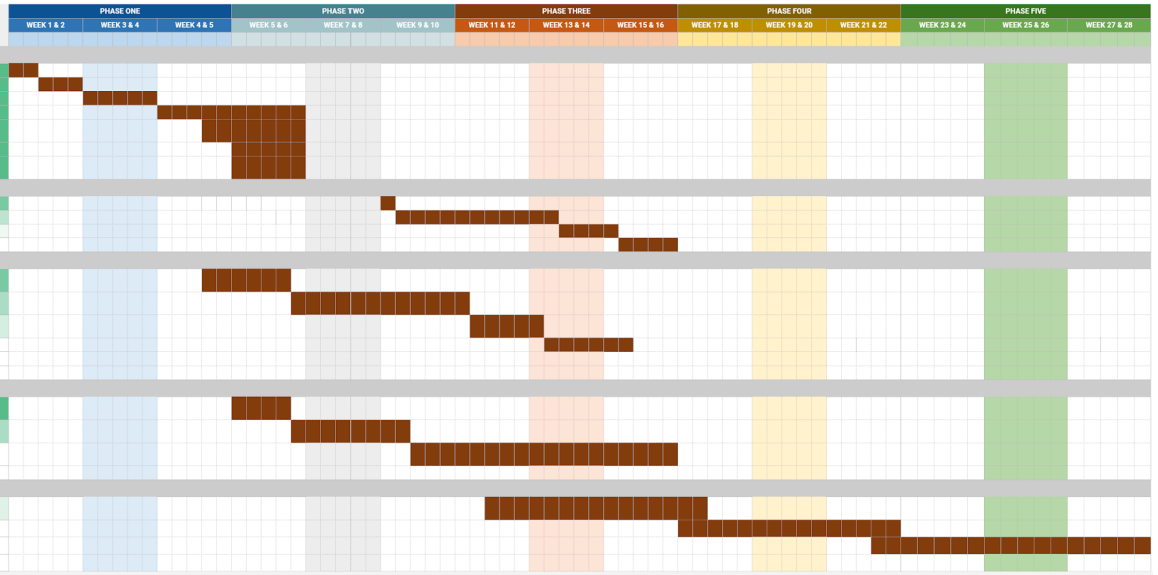
1. Initial Cosim demo and environment setup
    a. Have everyone on the team complete the demo application
2. Modifying the Demo
    a. Make a team pull request to the *Cosim-Demo* repo and get feedback
    b. Have every member of the team understand and change the memory-mapped register data to the Real Time Clock (RTC)
    c. Make contact with the *Cosim-demo* repository managers to gain feedback on the additional features and documentation
    d. Publish a centralized startup documentation for the *Cosim-demo* repository

    e. Create a list of milestones for implementing the thread framework described by the client.
3. Implement working remote-port communication capabilities for bi-directional communication between TLM and Linux environment
    a. Determine a candidate device for bi-directional memory mapped simulation form a hardware and firmware side
    b. Create a comprehensive document for adding new simulated devices from a hardware (SystemC side) and software (Linux Buildroot in QEMU) side.
    c. Develop a test application to aid in understanding both sides of the Linux socket protocol and custom data packets in use
4. Public Contributions
    a. Make 3+ documentation contributions to public Xilinx Co-Simulation repositories
    b. Augment the initial demo application repo (*Cosim-demo*) to include additional bi-directional remote-port capabilities
5. Collaborate with and build off the work of parallel teams work during the summer at Collins
    a. Consult with additional team progress achieved over the summer in the Co-simulation documentation and domain
    b. Understand real internal use cases and need to develop additional demos for advance communication protocols (ARINC 717, UART, Linux Serial System)

Project Timeline

PROJECT TITLE	Xilinx Co-Simulation	COMPANY NAME	Senior Design Team Dec21-02
PROJECT Client	Matthew Weber	DATE	3/0/2021
		REVISED	4/5/2021

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION (Weeks)	PCT OF TASK COMPLETE
1	**Startup and Learning**					
1.1	Get a shared sever				1	100%
1.1.1	Upgrade server				0.5	100%
1.2	Have everyone run the demo				2	100%
1.3	Modify the demo application				4	100%
1.4	Read SystemC doucmentation articals				3	100%
1.5	Read indepth QEMU documenation				2	100%
1.6	Reach out to development community				2	100%
2	**Remote Port Protocal**					
2.1	Define project sructure and changes				2	80%
2.2	Provide working example				4	40%
2.3	Documentation				1	10%
2.4	Publication/PR				1	0%
3	**Threading**					
3.1	Provide plan for implementing theading architecture				1.5	80%
3.2	Get working implementation/demo for remote-port				4	50%
3.2.1	Document Setup and running example				2	25%
3.2.2	Submit PR/Publish it				1	0%
3.3					0	0%
3.3.1					0	0%
4	**Contributions**					
4.1	Get Inital feedback from public developers				1	100%
4.2	Make a PR for Cosim Documentation				2	50%
4.3	Publish remote-port example with detailed documentation				6	0%
4.4					0	0%
5	**Additional Co-Simulation Demos**					
5.1	IMU driver/hardware Co-Simulation inital demonstration				6	20%
5.2	ARINC 717 Protocal implementaiton				5	0%
5.3	UART for Linux Serial System				5	0%
5.4					0	0%



2.5 Project Tracking Procedures

A2.6 Personnel Effort Requirements

Task	Effort Estimate (Team Combined Hours)	Description	Reason
Environment setup	20	This involves getting a shared server setup, meetings times established, website updated, communication mediums, and other team dynamics. Along with familiarization with Linux and the tools being used.	Getting a server acquired and team dynamics should be trivial. Learning the tools and getting the basic demo up and running for each team member will take time as these tools are new to all members of the team.
Modifying The Demo	45	Making a change to the demo to demonstrate understanding and as a starting point to extending the demo with new functionality.	Modifying the demo should be a slightly difficult undertaking. Team members will still be learning the tools and exploring bugs and other errors that are encountered. In addition, once the demo is modified, it needs to be then replicated by all members of the team to ensure that all members understand the process and reasoning behind changes to contribute going forward.
Remote-Port Protocol	60	Implement a bi-directional remote-port protocol so that time series data devices can be played back and communication/data	This protocol will allow for bi-direcitonal communication between the simulated hardware and the software on

		can be sent from both the hardware and software virtual interfaces using the TLM port of the QEMU simulation.	the Xilinx ARM processor in QEMU. As such , ensuring the standard practices of communicating in TLM will be new for this task and ensuring the communication is working bi-directionally will require significant testing and documentation.
Thread Protocol	20	Better comprehend threading structure of SystemC for future use in the Co-Simulation server planned.	This technology will likely be of use when developing the device communication server implemented in SystemC at the end of the first semester. As such, it is important to understand this feature and framework initially with the threading demo for custom implementation later on.
Additional Sensor Example	90	Implement an additional sensor as an example to further bolster the offering of the Cosim demo repository. This is to allow an easier and wider range of opportunities for others to learn the development tools..	This involves digging into Linux driver implementation and datasheets for memory-mapped sensors. IT then also involves implementing said sensor in SystemC and in the software side of the driver in QEMU and buildroot for simulation. As no members of the team have experience in these skills, it will be a significant learning curve.
Public Cosim Demo Documentation	30	Make multiple public improvements to the	This involves communicating with

		little documentation that is provided for the Cosim demo repository and other tooling utilized during the project.	members of the community which can be slow at times. It is also important to compose detailed documentation, as it is what can help others to solve issues and learn the tools. Documentation should not be rushed, although should be trivial if the tools are known.
Colins Team Collaboration	300+ hours	Meeting with additional development teams at Colins to understand additional use cases that would benefit from the Co-Simulation framework being developed by our team.	Over the summer, it is likely that additional development teams at Collins will continue to develop additional features on our public contributions. As such, during the Fall semester, we plan to meet with those teams to understand their advances and guide our future work around developing additional use cases and protocols that aid theri internal needs. As such, it is unclear what exact interfaces we will be asked to implement although ARINC 717 and UART for the Linux Serial System have been mentioned as likely goals.

2.7 Other Resource Requirements

For this project, a shared computing environment is needed for our team to effectively develop the additions to the software described. As such, since we are simulating complex processors in parallel

and hardware devices attached to them, this requires a significant amount of computing resources. A powerful Linux server is needed to support these computing needs for our project. This is currently being provided by the Department of Electrical and Computer Engineering and guaranteed until the completion of our project. A team communication platform for weekly meetings is also required to communicate with one another. A Discord server was chosen due to its no cost and open environment. This allows us to work remotely, hold meetings, and share information in real time with one another when we are not able to meet in person. Code repository and hosting services are again provided by the Department of Electrical and Computer Engineering, while other documentation and demos remain public or provided by our client and his team. The goal for our project is to utilize the current public resources and guidance to produce additional documentation, tools, and resources for other developers hoping to utilize the powerful Co-Simulation framework provided by Xilinx with QEMU + SystemC-TLM structure. All deliverables aim to be open-sourced and free for anyone to use and readily available where it makes sense in public development channels.

2.8 Financial Requirements

This project will in all require no financial requirements. The shared computing resources have been generously provided by the Department of Electrical and Computer Engineering and all of the software needed is free and open-source. Any professional consultants provided are done so free of charge though the teams of the client's company (Collin's Aerospace). Any public documentation, tools, feedback, or advice is done so free of charge due to the nature of the open-source software community on the Github platform and the public nature of the Xilinx simulation toolchain.

3 Design

3.1 Previous Work And Literature

Various different simulation technologies already exist for simulating both processor behavior and respective simulation environments individually/separately. However, the "cosim" model combines the two and though this technology exists, there lacks both sufficient documentation and demonstrations.

In essence, the cosim model as a toolchain is relatively new and therefore improving documentation and demos will be a large focus of this project in order to make the technology more approachable to prospective users.

The majority of project work will be focused on extending the usefulness of an already existing simulation environment. This means that background research is somewhat limited in its scope to learning about the technologies already being used by the system. The project group is currently focused on learning about those technologies.

Background literature for this project includes SystemC tutorials, a Xilinx emulator user guide, co-simulation documents, and any other work found on the open source forums.

Literature:

Banerjee, Amal, and Balmiki Sur. "SystemC-AMS and SystemC Combinations." *SystemC and SystemC-AMS in Practice*, 2013, pp. 449–455., doi:10.1007/978-3-319-01147-9_17.

Ammari, Ahmed Chiheb, et al. "HW/SW Co-'Esign for Dates Classification on Xilinx Zynq SoC." *2020 26th Conference of Open Innovations Association (FRUCT)*, 2020, doi:10.23919/fruct48808.2020.9087548.

Xilinx. "Xilinx Quick Emulator User Guide." 2019.

3.2 Design Thinking

Co-sim technologies exist yet are not well known within the target community. A relatively new tech, improving documentation and demos will make cosimulation tech more approachable for the community and hopefully allow for increased usage of these technologies among target constituents.

Because another large deliverable of the project is implementing further system logic and driver support, detailed and thorough documentation and demo of the tech will again be imperative to project success.

3.3 Proposed Design

Thus far, our team has been focused on getting the co-sim demo running on each of our devices. Project scope for the semester includes understanding the various concepts/technologies, project documentation, and implementing some basic extensions to the system.

The first extension our group will be focused on this semester is implementing a transmit use case on top of an existing use case that receives data. Currently, the processor only receives data from simulated firmware. We will be implementing the opposite - the processor will be able to transmit data to the simulated firmware.

The second extension our group will be implementing is a UDP bridge in order to send and receive data from the SystemC model. This bridge will be used to support dataflows in the memory map interface model.

Our team will demonstrate the functionality of a UDP connection by generating a demo of an arbitrary sensor modeled in SystemC correctly functioning with an off the shelf Linux peripheral driver. This demonstration, coupled with documentation describing the purpose of the co-sim, will be valuable to companies who are evaluating the use of the co-sim environment for testing their code.

Due to the need for replication of our project, it is also imperative that our group creates and maintains detailed documentation of work being done throughout the project timeline. Existing documentation on these technologies is difficult to read/hard to find. Therefore, creating readable and more extensive documentation is a must.

3.4 Technology Considerations

Most technology decisions regarding this project have already been made. One strength of the project is that the co-simulation model offers increased flexibility as compared to a "real world + simulation" model. Project work will potentially answer client needs more accurately because team members only need a computer with an internet connection - both of which have already been established for all members of the team. The alternative "real world + simulation" model would require work to be done in a specific location(s) as opposed to remotely using hardware such as an FPGA.

The need to collaborate remotely has led our team to utilize a common, headless Ubuntu server like mentioned above.

As previously mentioned, various simulators already exist for simulating processor and environment behavior. That in consideration, QEMU and SystemC + TLM were both adopted by the client and therefore outside of our group's scope for further consideration.

3.5 Design Analysis

Design analysis is fairly preemptive at this point in the project. Our current design analysis process will focus on risk identification and mitigation to ensure project success moving forward. We suspect the proposed design (Section 3.3) will be fairly successful because of the simple fact that project work in the "engineering department" will focus on extending an already existing system like previously mentioned.

3.6 Development Process

Though this project doesn't fit into any one "specific" development process, it most closely resembles the Agile approach because it is being completed through small, iterative progress chunks. Our group has chosen to use Trello to track progress.

This development process was selected because it allows for high client involvement and is easily applicable to the system when other development processes would make less sense logically.

3.7 Design Plan

Project focus for this semester will mainly center around improving demos and documentation to improve approachability of cosim technology. Next semester, project focus will shift to adding increased functionality to the system in the form of additional driver modules.

4 Testing

4.1 Unit Testing

We will test each individual demo component on its own. The SystemC TLM models for each demo can also be tested individually prior to attaching the appropriate Linux driver. As our project mainly

consists of implementing minor contributions to existing infrastructure, most of our work will tightly interface with surrounding components and won't be usable in isolation.

4.2 Interface Testing

The primary interface we have to worry about is the connection between QEMU and our SystemC model. As the nature and specifics of this connection are outside of the scope of our project, we don't need to substantially test it. We do however need to ensure that our SystemC work is accurately being translated into the QEMU environment using this connection. Once we build our UDP interface, we will need to ensure that data is transferred correctly between the host OS and our SystemC model. Specifically, we need to make sure read and write calls do what they intend to do, and that no data is lost when data transfer occurs.

4.3 Acceptance Testing

In the future, when we have built more parts for this project, we will functionally test all of them and give a demonstration to our client to ensure that they are up to professional standards. Along with our client, the application we create will need to be up to the community standards as well. Sending in a ticket with a documented pull request will give us an idea of if the open source community will accept our application, or if we need to bring it back for further development.

4.4 Results

Our team has been able to test that the demo given to us by the client functions correctly. This gives us a basic understanding of how the co-simulation works and now we can start researching how to create the UDP Bridge and connect it to the program.

5 Implementation

Each section of our deliverables are neatly divided into demos. For instance, our client would like us to demo a working out of the box Linux driver running on our co-sim platform. So, our team will start our implementation with a focus on our Client Demos. Our team will lay out all of the requirements for each demo, and then implement them iteratively to add the needed functionality.

To actually begin implementing each demo, we will begin by researching the existing material already in the project. Our demos seek to build upon open source repositories, and these projects already have significant amounts of documentation and publicly available correspondence regarding potential tasks.

Our team will utilize all of the available resources associated with each project to make a determination on how the feature will fit into the demo. Next, we will reach out to the maintainers to get feedback on our proposed changes. Understanding how the maintainers of the repository want our changes to fit into the existing demo is key to us properly implementing the change.

From this point, we can actually go ahead and make our changes. Much of our changes will be in the SystemC demo code that is written, in order to add additional functionality to QEMU memory interface. We can then test our changes on the demo by using basic Linux memory access commands to verify that we are adjusting memory as required.

Finally, these open sourced projects require a high amount of documentation. We are seeking to expand the documentation already present, as well as generate new documentation so that these repositories can be more accessible. So, our future implementation will need to involve growing the quantity and quality of documentation for each different demo, and then publishing these documents appropriately such that they can be used by the community in the future.

6 Closing Material

6.1 Conclusion

So far, our team has demonstrated proficiency executing a co-sim demonstration. We have begun the process of learning more about the toolchain, and will soon begin experimenting with alternate SystemC models driving our QEMU memory model. This will naturally progress us into our first major deliverable: demonstrating a working Linux Peripheral driver executing via Cosim.

Going forward, our team will continue to adopt an iterative approach in adding functionality to the Cosim system. Our team will also be reaching out to open source projects and identifying areas where we can expand the existing offering. This expansion will be done in the form of new demos, documentation, or additional features.

6.2 References

J. Komlodi and V. Garhwal, "Co-simulation," *Confluence*. [Online]. Available: https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/862421112/Co-simulation. [Accessed: 09-Mar-2021].

Xilinx, "systemctlm-cosim-demo," *GitHub*. [Online]. Available: https://github.com/Xilinx/systemctlm-cosim-demo. [Accessed: 09-Mar-2021].

6.3 Appendices

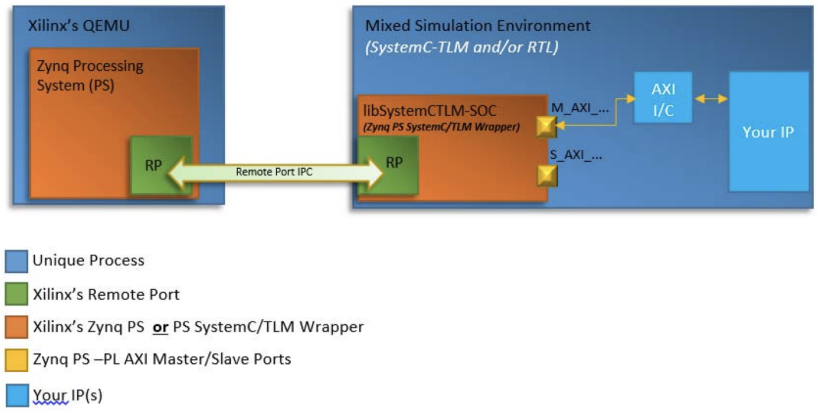


Figure 1 - Xilinx QEMU Mixed Simulation Environment

Xilinx. "Xilinx Quick Emulator User Guide." 2019, p. 37.