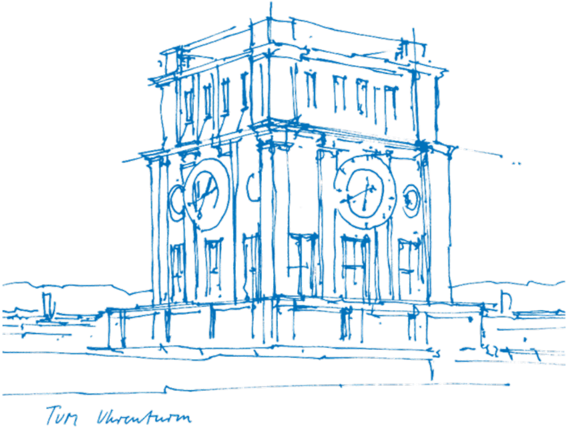
Chair of Theoretical Information Technology
Department of Electrical and Computer Engineering
Technical University of Munich

General Quantum Hardware Simulation

QuaSi Framework: Proposal



Simon Sekavcnik

08.05.2023

Outline

Outline

Introduction

- Quantum mechanics is notoriously difficult to simulate on classical hardware.
- Small quantum systems, can require an impractical amount of computational resources to simulate.
- Depending on the goal of the simulation, certain aspects of quantum mechanic may be emulated.
- Developing software that can simulate quantum mechanics helps advance quantum technologies.
- Quantum hardware simulation has applications in quantum cryptography, communication, computing and sensing.
- The topic of this talk is to present the proposal for simulation framework.

Outline

1 Introduction

2 QuaSi Framework: Idea

3 Quasi Framework: Structured Approach

4 Quasi Framework: Discussions

Goals of the QuaSi framework

1. Simulation framework for quantum hardware benchmarking
2. Generic tool for modeling experiments and quantum devices;
3. Cost-effective approach to test ideas prior to resource commitment;
4. Potential to improve efficiency of experimentation;
5. Enables exploration of new hardware possibilities;
6. Facilitates collaboration between researchers and hardware developers;
7. Potential to drive innovation in the development of quantum technology.

Python

Why we chose Python:

1. Python will be used as the primary programming language for the project.
2. Python is widely adopted by academia as a programming language for scientific research and data analysis.
3. Python has extensive testing and documentation support, with a vast selection of libraries which considerably reduces development time.
4. Python is open-source, ensuring that the project does not require an expensive license, and the community can contribute to continuously enhance it.

Python: Potential Drawbacks

1. Python may not always be the go-to language, in this case we may offer assistance when it comes to implementations.
2. Python can be slower at runtime compared to compiled languages like C++ or Java, which may be an issue for tasks that require high performance.
3. With its dynamically-typed nature, Python doesn't have strong enforcement of variable types, which could cause unexpected errors in large or complex codebases.
4. For memory-intensive tasks, Python's garbage collection can cause performance issues and may not be as efficient as manual memory management in other languages.

Mojo

1. Mojo is a superset of Python, allowing for seamless execution of Python code.
2. Mojo introduces stronger types than Python, providing better type safety and reducing unexpected errors.
3. The language will be released under an open-source license, encouraging contributions from the community.
4. Mojo promises significant performance improvements of up to 4000x for certain tasks, such as algebraic manipulations.
5. The language is currently in closed beta, but developers can request early access to try Mojo and provide feedback.

Outline

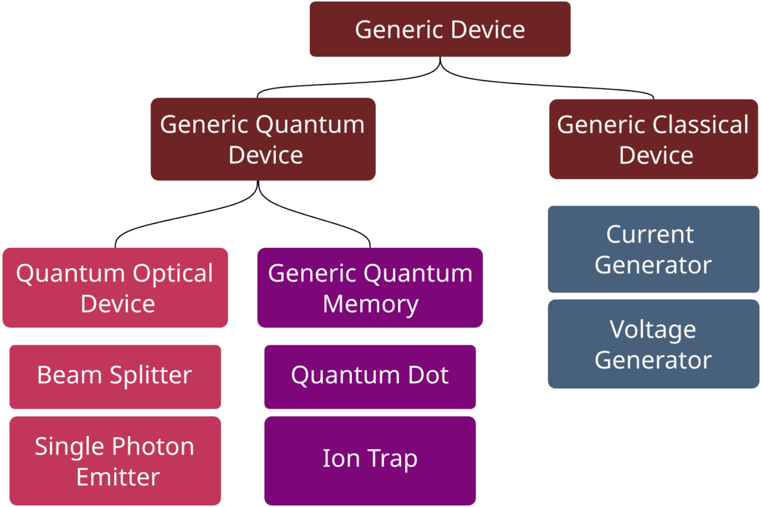
QuaSi Framework: Abstractions

- Simulation Framework Setup:
  - Devices: Structured simulation of device behavior.
  - Signals: Representation of device-produced or fed signals.
  - Simulation: Management of simulation tasks, such as data collection and analysis.
- Different types and implementations of devices and signals all inherit from some `AbstractBaseClass()`:
  - to ensure that important behaviors are implemented,
  - to ensure that they are implemented in the fashion that adheres to the rest, of the framework,
  - to ensure interoperability.

Devices

- Represent structured behavior of different devices
- All devices inherit the structure from already defined `GenericDevice(ABC)`, so any device included in the framework must include the following attributes and methods:
  - Ports: `Dictionary`, labeling different `input` and `output` ports through which device receives or emits a specific type of `Signal`
  - Properties: `Dictionary`, documenting operating properties of the devices (power requirements, operating temperature, reference to published paper, CAD model, icon)
    - **Brainstorming is needed for the complete list of required properties**
    - Entries could be mandatory or optional
  - Behavior: `method`, every device will need to implement a `method` called `compute_outputs()`.
    - The method operates in a way (decorators are already defined), where the inputs are already computed and present in the ports.
    - The device could also have implemented some internal state.

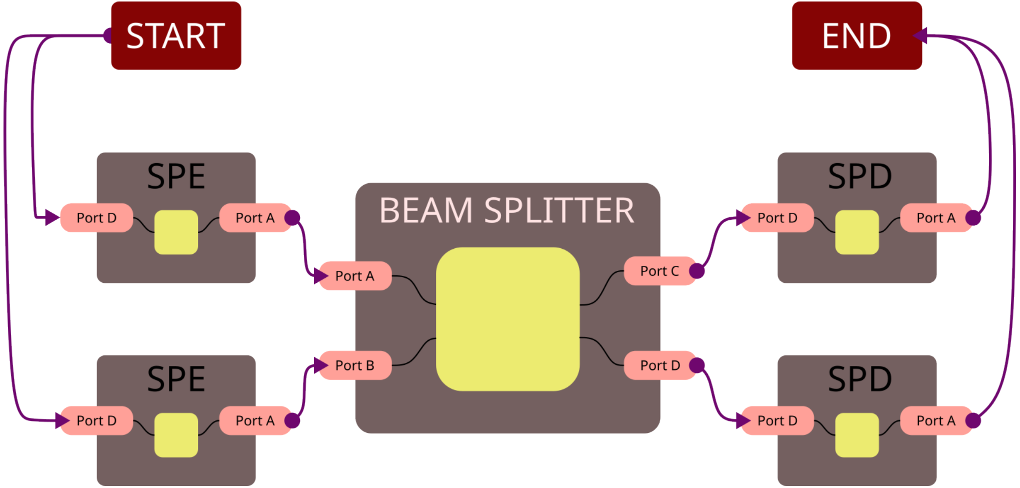
Devices: Hierarchy



Signals

- Signals encode type of Signal that is being produced or consumed by any device.
- Currently only the structure for the base signal is implemented.
- Different types of signals implemented could be:
  - Discrete Signal
  - Analog Signal
  - Generic Quantum Signal
    - Quantum Optical Signal
- Quantum signals will need to have a mechanism to track the correlations (entanglement).

Example: Simulation Flow



Extras: Bibliography

- Optional property of the devices will be reference.
- The reference should point to published work describing the operation of the device.
- The simulation will have an ability to compile the list of references for the devices used in one specific simulation.

Repository

- Hosted on GitHub
- Currently Private Repository:
  - Access available after invitation (Closed pre-alpha)

Contributions

- Due to the complexity of the topic in question the code quality must be monitored and maintained
- Python linting is configured
- Strict testing and broad testing will be implemented
- Code is added to repository through so called **pull requests**
  - If code meets quality expectations (linting and testing) it will be merged with the master branch.

Outline

Questions and Comments

- We are open to all comments, suggestions and possible pull requests.
- Some open questions:
  - Confidentiality:
    - If confidentiality is a concern, the base repository can be forked and developed in parallel, so confidential components remain confidential.
  - Important properties describing quatnum devices.
  - Possible depth of the simulation.
    - How well are devices understood, can the operations be described using qunatum mechanical descriptions?