基于Python和事务级的Meta-HVL验证框架

华南理工大学 林学
华南理工大学 赖晓铮
芯华章科技 杨晔

本文提出了两种基于 Python 的芯片验证方案，旨在提高芯片验证的效率和灵活性。其中，一种方案基于 Cocotb+VPI 接口，将时序部分下沉到硬件设计部分；另一种方案基于 pybind11+DPI 接口，对性能进行了极大的优化。本文在不同仿真框架下测试并比较了仿真性能，结果表明，该框架能够大幅提高仿真性能，并保持使用高级语言的一定生产力优势。

在基于 Cocotb+VPI 接口的方案中，Cocotb 提供了一种高效的方式来编写基于 Python 的硬件验证环境。与传统基于高级语言的芯片验证方案相比，该方案具有相当的性能优化。通过使用 VPI 接口，Cocotb 可以与 Verilog 仿真器进行通信，将 Python 代码与仿真器中的代码进行集成。

此外，将时序部分下沉到硬件设计部分，将通信建立在事务级别，能够极大的减少通信上的开销来提高仿真效率。该方案可以更好地利用 Python 层丰富的生态，为验证工程师提供更高效、灵活的芯片验证方法。

另一方面，基于 pybind11+DPI 接口的方案则更加注重性能优化。pybind11 提供了一种将 Python 代码集成到 C++ 环境中的方式，而使用 DPI 接口则可以实现 SV 层与 C++ 层的数据交互，使 HDL 代码可以使用 Python 层的能力，从而实现更高效的芯片验证。该方案具有更高的性能，同时保持了使用高级语言的一定生产力优势。通过将 Python 代码与 C++ 代码进行集成，验证工程师可以使用 Python 的高级特性来编写代码，同时利用 C++ 代码实现更高效的仿真

高层次设计和验证
基于Cocotb的原有Python验证方案在很多场景性能受限

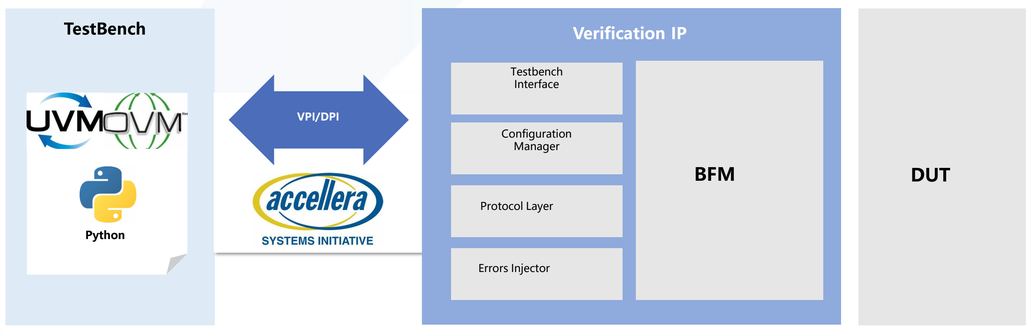
- 专注于敏捷前端流程中的敏捷验证部分 – Meta HVL，验证对象兼容现有Verilog和Meta-HDL设计流程
- **Meta HVL不仅是HVL语言，还是HVL语言的框架，工具和平台。**
- **所以Meta HVL项目既包含HVL语言(python)，也有基于HVL语言的UVM框架(python-UVM)和验证平台(cocotb)，还要引入软件领域验证工具！**

项目价值

- 面向软件和应用背景工程师，用单一python语言的验证生态系统，降低验证学习和编码难度，提高验证效率
- 通过提高Testbench与DUT之间通信的抽象级别，以事务级别建立通信，从而减少通信开销并实现加速
- 可以无缝在多种软硬件仿真平台之间通用，并保持使用高级语言的一定生产力优势

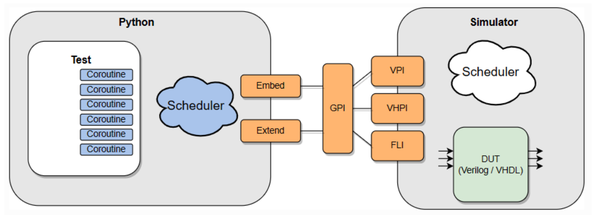
研究目标

- 让大部分验证工程师也尽可能与HDL电路实现分离，降低验证工作的门槛
- 使用更易学、易用、面向应用、有丰富生态的语言
- 更完善的仿真工具支持和仿真性能
- 能与现有的Verilog、SystemVerilog生态集成



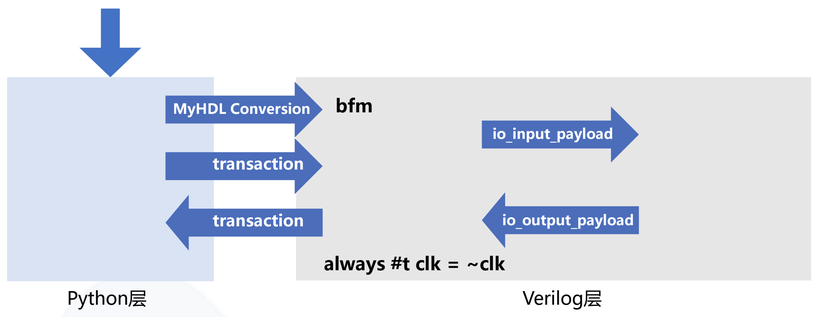
基于Cocotb的原有Python验证方案在很多场景性能受限

传统基于Cocotb的仿真验证框架中，Cocotb 按照时序将激励驱动到 DUT 的端口上，并直接从 Python 监测输出，如下图。由于Testbench与DUT之间的通信基于时钟周期，频繁的进程间通信会对性能产生的极大的损耗。



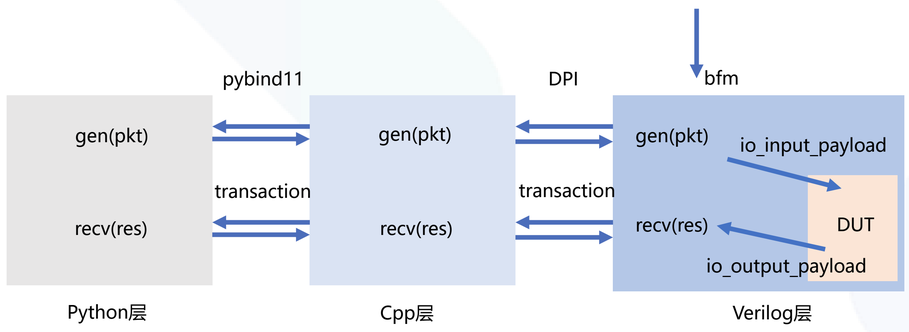
传统基于Cocotb的仿真验证框架

基于Cocotb+MyHDL的性能优化方案



使用 MyHDL 实现 bfm，然后将其转换为 Verilog

基于pybind11+DPI的性能优化方案



Corundum ARP仿真框架性能测试

	cocotb+iverilog			cocotb+Galaxsim			Galaxsim	pybind11+dpi+Galaxsim
	Python层	时钟下沉	bfm下沉	Python层	时钟下沉	bfm下沉		
1万次激励所需时间/s	118.37	83.81	15.82	111.78	76.83	9.11	0.306	6.23
时间比/%	100	70.8	13.36	100	68.73	8.15	0.27	5.57

基于Cocotb+VPI 接口的芯片验证优化方案，与传统基于时钟的交互方式相比，对于Corundum ARP，将时钟激励的产生部分下沉到HDL层，可以将时间缩减为原方法的约70%；而将与DUT相关的时序操作部分都下沉到HDL层，对于iverilog，可以将时间缩减为原方法的约13%，对于Galaxsim，可以将时间缩减为原方法的约8%，极大地减少了与传统System Verilog的差距。基于pybind11+DPI 接口的芯片验证优化方案，对于Corundum ARP，对于Galaxsim，可以将时间缩减为原方法的约5.6%，极大地减少了与传统System Verilog的差距。

Poseidon哈希仿真框架性能测试

	sv+iverilog	cocotb+iverilog			Galaxsim	cocotb+Galaxsim			pybind11+dpi+Galaxsim
		Python层	时钟下沉	bfm下沉		Python层	时钟下沉	bfm下沉	
一个package的hash激励次数		1	1	10		1	1	10	
一个package的bit位数				7650				7650	
100次hash激励所需时间/s	106.19	109.36	108.14	108.11	5.08	11.29	9.85	7.83	5.16
时间比/%	97.1	100	98.88	98.86	45	100	87.25	69.35	45.7

基于pybind11+DPI 接口的芯片验证优化方案，对于Poseidon哈希，对于Galaxsim，可以将时间缩减为原方法的约46%，已经与传统System Verilog性能差距不大。