Coroutine Co-Simulation Test Bench

NIGIL MOHRA

Introduction

Coroutine Co-Simulation Test Bench (**COCOTB**) is an open-source framework hosted on GitHub, designed for digital logic verification using Python. It provides a Python interface to control standard RTL simulators and offers an alternative to traditional verification frameworks like Verilog, SystemVerilog, or VHDL.

Current Testbenches

Hardware design (synthesizable) and verification (testbenches are non-synthesizable) are distinct processes. Using the same language for both may not be optimal, as higher-level language concepts, such as object-oriented programming (OOP), are particularly useful when writing complex testbenches.

One solution is to add higher-level programming features to hardware description languages. SystemVerilog is an initial approach, incorporating simulation-only OOP language features. The Universal Verification Methodology (UVM) libraries, written in SystemVerilog, follow this path. UVM offers a well-defined set of coding guidelines, with a structured testbench design and a SystemVerilog base class library for creating reusable verification testbenches.

However, there are challenges with this approach. SystemVerilog is a complex language with a specification spanning nearly a thousand pages and 221 keywords, compared to just 83 in C++. While powerful, it requires significant time to master. Additionally, the SV-UVM framework, though similarly powerful, has over 300 keywords and also takes time to master.

Verification using Python based Test Benches

COCOTB was developed by **Chris Higgs** and **Stuart Hodgson**, who took a different approach to hardware verification. Instead of using traditional hardware description languages, they chose to use a high-level, general-purpose, object-oriented programming (OOP) language for developing testbenches, selecting Python as their language of choice. Python is simple, with only 23 keywords, making it easy to learn, yet very powerful. It also boasts a large standard library and a vast ecosystem, offering numerous existing libraries.

Co-simulation (Triggers)

In co-simulation the design and testbench are simulated separately but interact with each other. Communication between the two occurs through VPI / VHPI interfaces, which are generated by the COCOTB triggers. When a trigger is yielded or awaited, the testbench waits until the triggered condition is satisfied before resuming the execution.

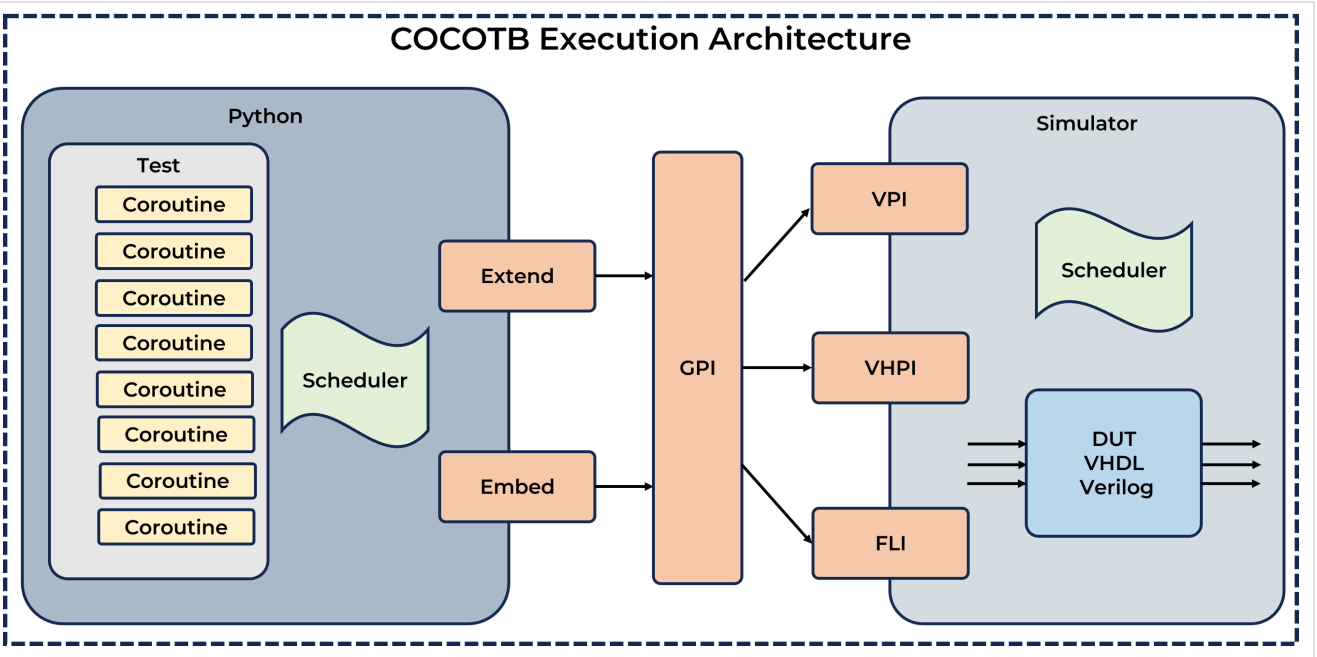
Coroutines

In COCOTB, coroutines are simply functions that follow two key properties: they are decorated using the `@cocotb.coroutine` decorator, and they contain at least one `yield` or `await` statement, yielding another coroutine or trigger. Coroutines can not only be yielded but also forked to run in parallel, enabling the creation of structures similar to a Verilog always block.

Available Triggers

Name	Type	Functionality
Timer	Time, Unit	Waits for a certain amount of simulation time to pass
Edge	Signal	Waits for a signal to change state - *rising or falling edge*
Rising Edge	Signal	Waits for the rising edge of a signal
Falling Edge	Signal	Waits for the falling edge of a signal
Clock Cycles	Signal, Number	Waits for some number of clocks - *transition from low to high*

Architecture



Architecture of COCOTB Execution