Coroutine Co-Simulation Test Bench (COCOTB)

Created By

NIGIL MOHRA

COCOTB

Introduction

1. COCOTB is an opensource framework hosted on GitHub.
2. COCOTB is a library for digital logic verification in Python.
3. Provides Python interface to control standard RTL Simulator.
4. Offers an alternative to using Verilog/System Verilog/VHDL framework for verification.

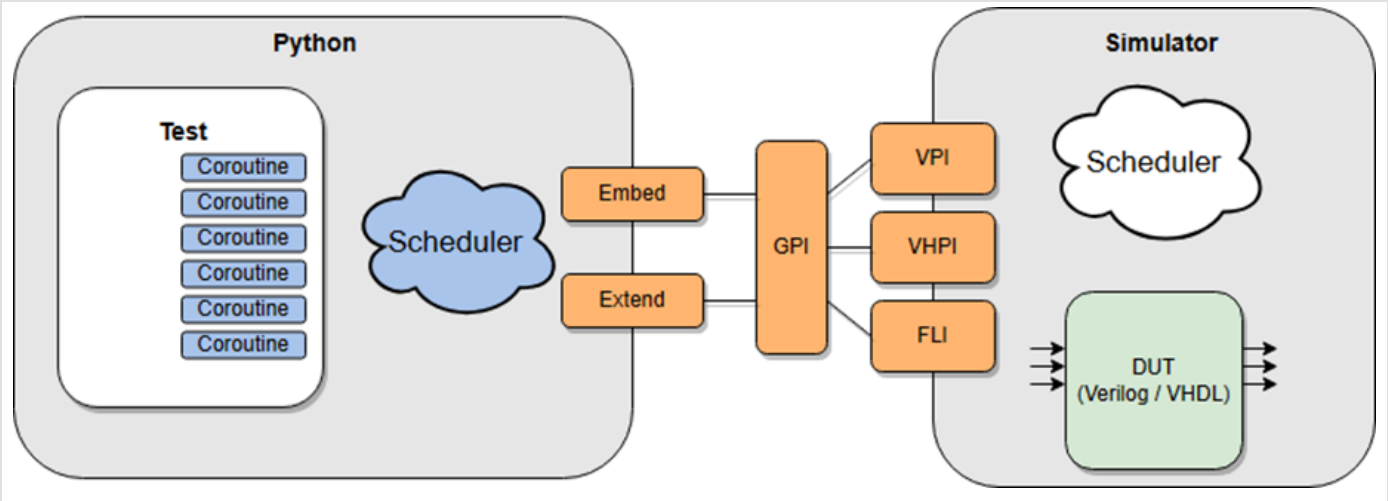

Figure 1. Architecture of Cocotb

Problems with Current Testbenches

1. Hardware design (Synthesizable) and verification (TB are Non Synthesizable) are different.
2. Using the same language for both might not be optimal. Higher level language concepts (like OOP) are useful when writing complex testbenches.
3. Add higher level programming features to the hardware description language. System Verilog is the first approach: simulation-only OOP language features.
4. UVM (Universal Verification Methodology) libraries written in System Verilog.
5. UVM is a well-defined set of coding guidelines with well-defined testbench structure, It's written in System Verilog and comes with System Verilog base class library for creating advanced reusable verification TB's.
6. Use an existing high level object oriented general purpose language for verification.
7. But, there are problems with this,
    1. The System Verilog is a fairly complex programming language.
    2. Specification is almost a thousand pages long.
    3. There are 221 keywords in the language, to 83 in C++.
8. It is a powerful, but it takes some time to master.
9. There is also SV-UVM but, it also has more than 300 keywords and though it is powerful, but it takes some time to master.

Verification using Python

1. COCOTB was developed by **Chris Higgs** and **Stuart Hodgson**, tried a different approach.
2. Use-a high level, general purpose, OOP language for developing test benches, they picked Python as their language of choice.
3. Python is simple (only 23 keywords) and easy to learn, but very powerful
4. Python has a large standard library and a huge ecosystem; lots of existing libraries.
5. Python is well documented and popular: lots of resources online.

Cosimulation (Triggers)

1. Design and TB simulated independently: this is called Cosimulation.
2. Communication through VPI/VHPI interfaces, generated by COCOTB "triggers".
3. When a trigger is yield/await, the testbench waits until the triggered condition is satisfied before resuming execution.

Coroutines

1. In COCOTB coroutines are just functions that obey two properties.
    1. Decorated using the @cocotb.coroutine decorator
    2. Contains at least one yield/await statement yielding another coroutine or trigger.
    3. Coroutine can be yielded, but they can also be forked to run in parallel. This allows the creation of something like a Verilog always block.

Available Triggers

Name	Type	Function
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, Num	Waits for some number of clocks (transitions from 0 to 1)