Cocotb

An alternative verification solution

Summary

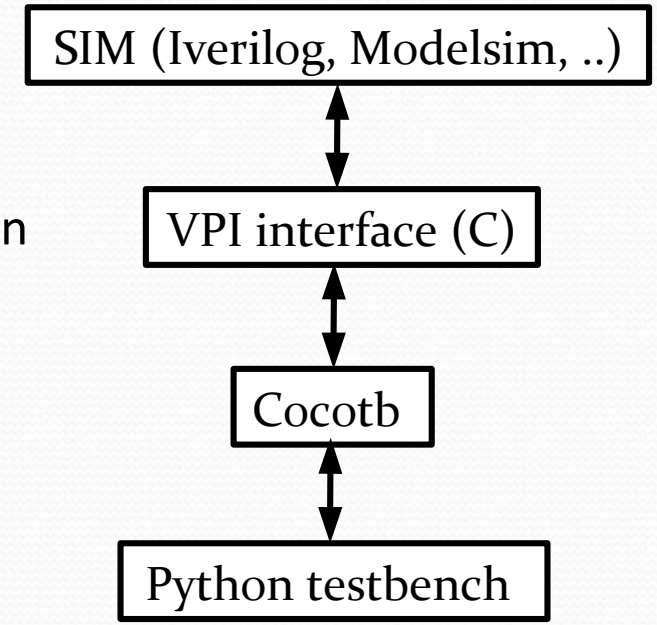
- Concept
- How it work
- Base syntax

Concept

- Use standard simulators to manage the RTL simulation
- Use Python to write your testbench

How that work

1. The simulator boot
2. Via the VPI interface it run Cocotb
3. Via annotation, cocotb run your testbench
4. By using the Cocotb API, your testbench can interact with the simulator



How the testbench work

- You can read and write all signals of your RTL (all !)
- You can fork new coroutines (Kind of simulation thread)
- All coroutines can use triggers to wait time or wait RTL edges
- Coroutines are Python's generator which use the yield statment to wait a triggers and inner coroutines

Testbench main exemple

```
import cocotb
from cocotb.result import TestFailure
from cocotb.triggers import Timer

@cocotb.test()
def myTestbench(dut):
    print("START")

    dut.reset = 1
    yield Timer(1000)

    dut.reset = 0
    yield Timer(1000)

    if dut.io_value != 42:
        raise TestFailure("io_value mismatch")

    print("SUCCESS")
```

Coroutine example

```
@cocotb.coroutine
def genClockAndReset(dut):
    dut.reset = 1
    dut.clk = 0
    yield Timer(1000)
    dut.reset = 0
    yield Timer(1000)
    while True:
        dut.clk = 1
        yield Timer(500)
        dut.clk = 0
        yield Timer(500)
```

```
@cocotb.coroutine
def checker(dut):
    yield Timer(10000)
    if dut.io_value != 42:
        raise TestFailure("io_value mismatch")
```

```
@cocotb.test()
def myTestbench(dut):
    cocotb.fork(genClockAndReset(dut))
    yield checker(dut)
```

Signals

- You can do
  - dut.signal == 42
  - dut.signal != 42
  - int(dut.signal) > 42
  - queue.put(int(dut.signal))
- But you can't do
  - dut.signal > 42
  - queue.put(dut.signal)
- Don't forget that Cocotb is an library, not an language. Some signals operator are overloaded to make life easier, not all of them.