БУДУЩЕЕ
В НАШИХ
РУКАХ

Обзор фреймворка PyUVM

Егор Ковалев

Инженер по верификации, YADRO - Департамент разработки СнК.

- Выпускник-магистр НИУ МИЭТ МПСУ, 2022.

Верификация цифрового дизайна

Цель верификации — убедиться, что результат некоторого преобразования является предполагаемым или ожидаемым

Верификация — это согласование с помощью различных средств спецификации и выхода

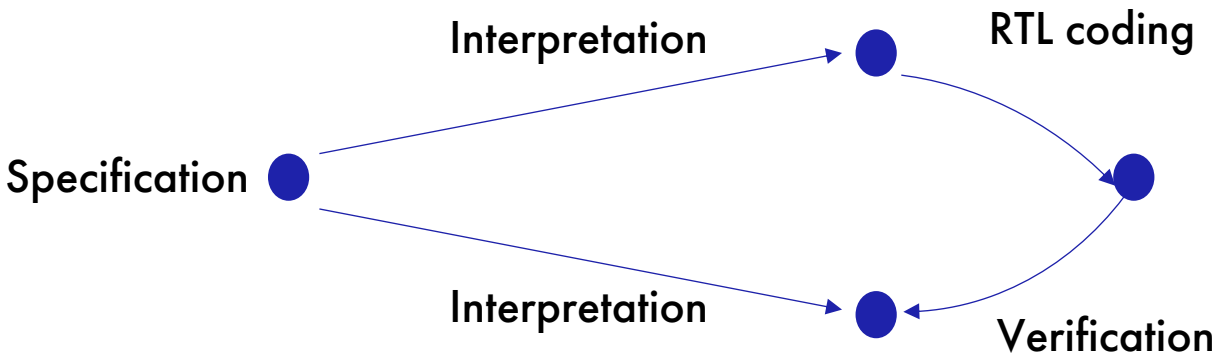
Верификация преобразования может быть выполнена только через второй реконвергентный путь с общим источником

Реконвергентные пути в неоднозначной ситуации

- Верификация собственного дизайна подтверждает его соответствие вашей интерпретации спецификации, а не самой спецификации

Верифицировать должен другой

- Верификация возможна при наличии избыточности в сомнительной ситуации



Что нужно для верификации

План тестирования

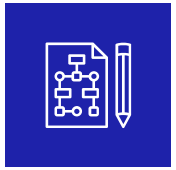
Что и как мы будем проверять,
как мы будем контролировать
прогресс

Система сборки проекта

Надо уметь запустить проект на моделирование,
получить результаты, уметь их воспроизвести

Тестовое окружение проекта

То, что будет подавать тестовые
воздействия и смотреть
за ответами системы



Тесты

Собственно то, что
мы будем запускать

Инструменты отладки

Механизм воспроизведения
ошибок и отладки

Механизм контроля и измерения прогресса

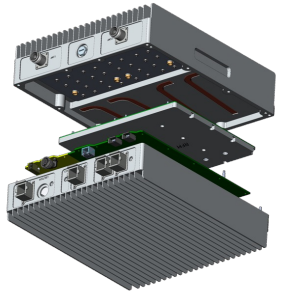
Надо разработать модель покрытия,
уметь собирать и анализировать его

Про YADRO

Мы создаём современные технологические продукты:

Серверы
VESNIN и VEGMAN

Системы хранения
данных TATLIN



Телеком
оборудование

Сетевое и коммутационное
оборудование KORNEFELD

Клиентские устройства
KVADRA

Почему не хватает рабочих кадров?

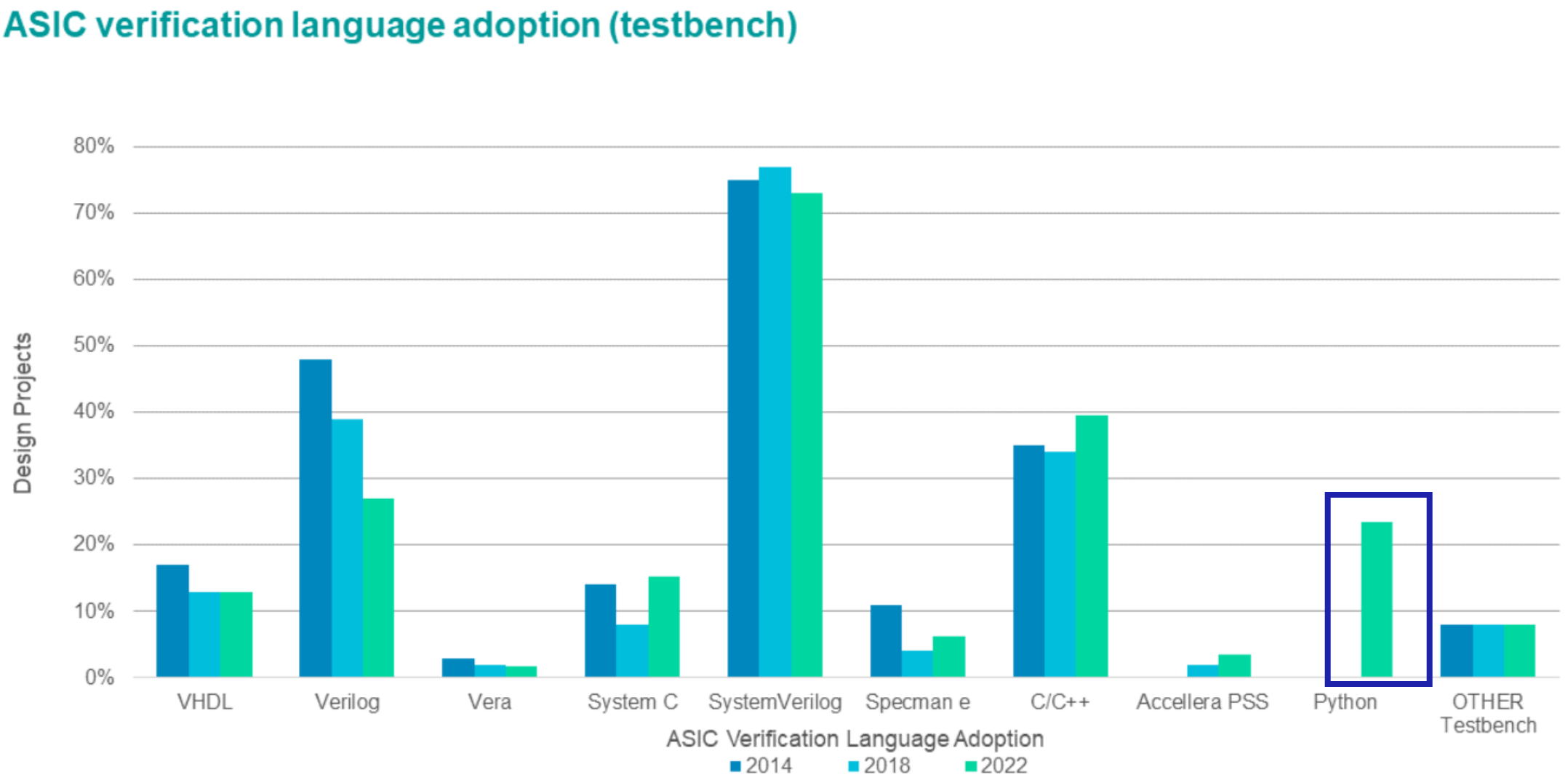
- Общая **сложность** данной области
  1. Большой конвейер работ для выпуска продукта на рынок ;
  2. Знание языков описания аппаратуры;
  3. Методологий, проприетарные инструменты.
- Мало специалистов

Решения от YADRO

Компания YADRO развивает обширную образовательную инициативу, в рамках которой расширяет сотрудничество с российскими вузами по ключевым направлениям: полному циклу разработки систем на кристалле (СнК), телекоммуникационной разработке и программной инженерии, в том числе для систем хранения данных.

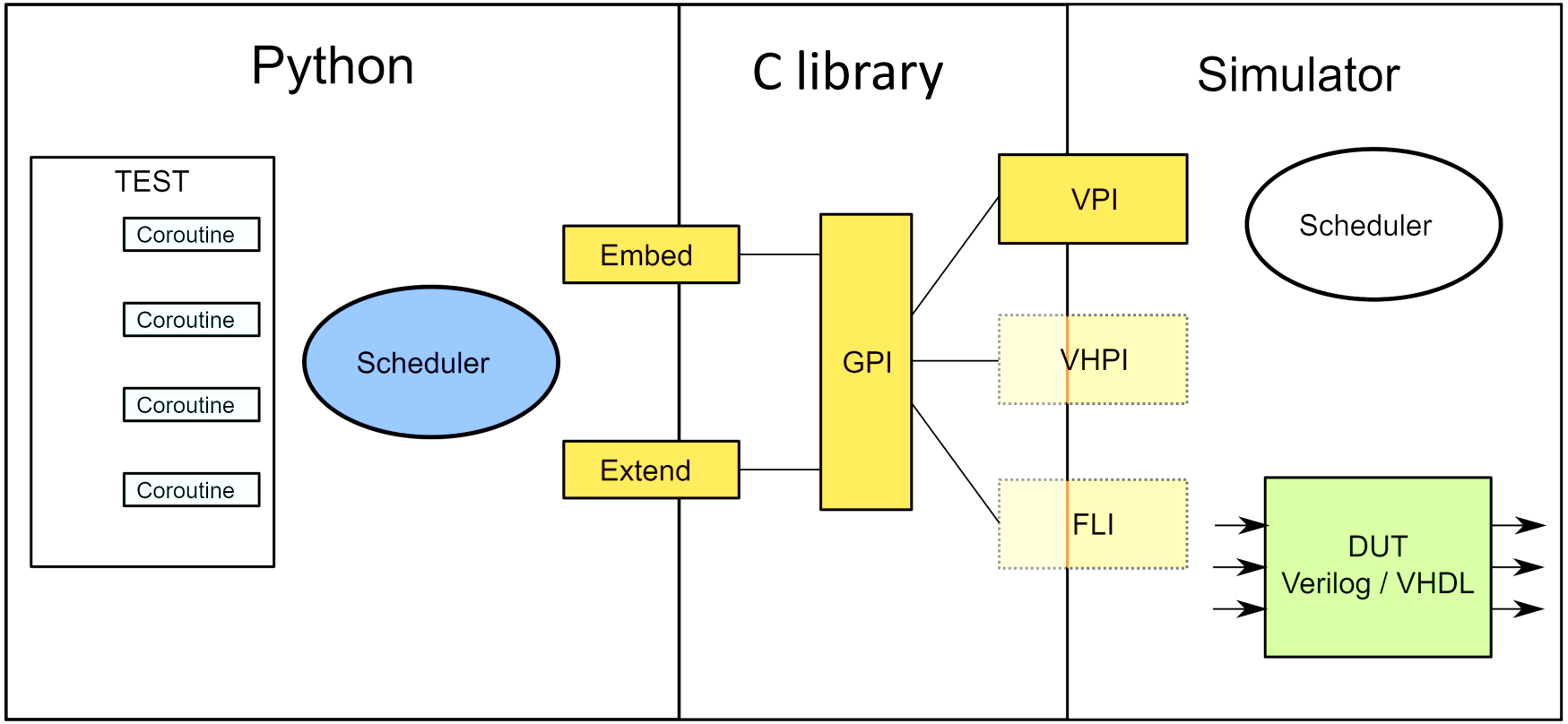
- Появление новых кафедр в образовательных учреждениях с соответствующими программами обучения;
- Стажировки ;
- Внедрение популярных языков в верификацию цифрового дизайна.

Язык Python самый популярный в мире



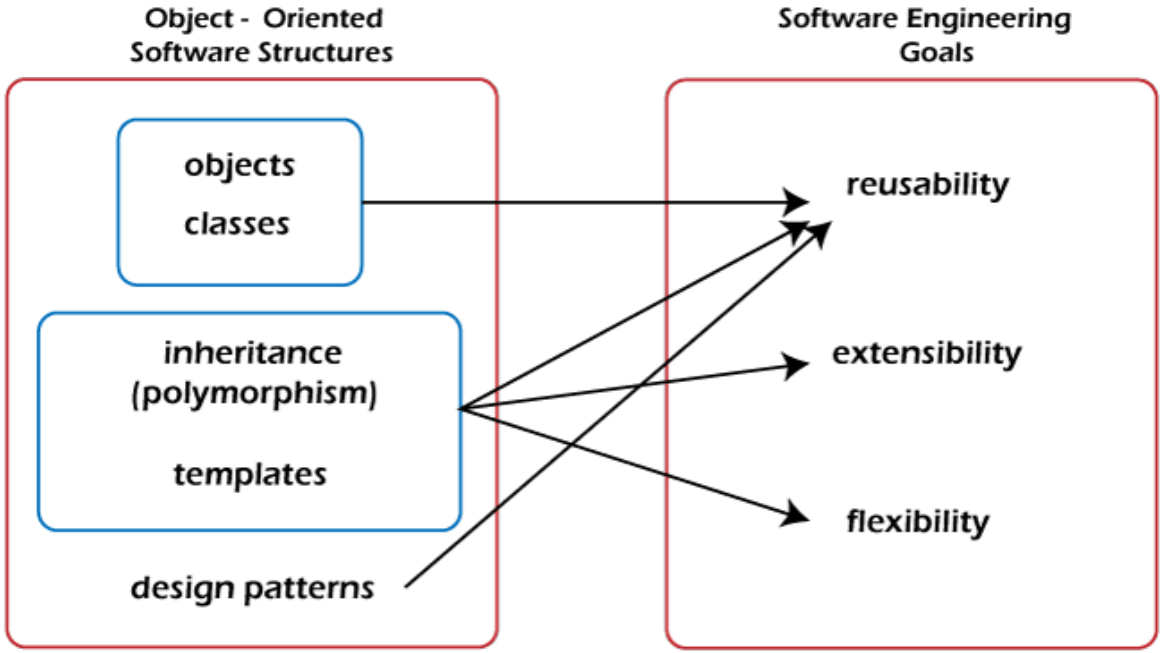
Фреймворк Cocotb

Cocotb is a coroutine based cosimulation library



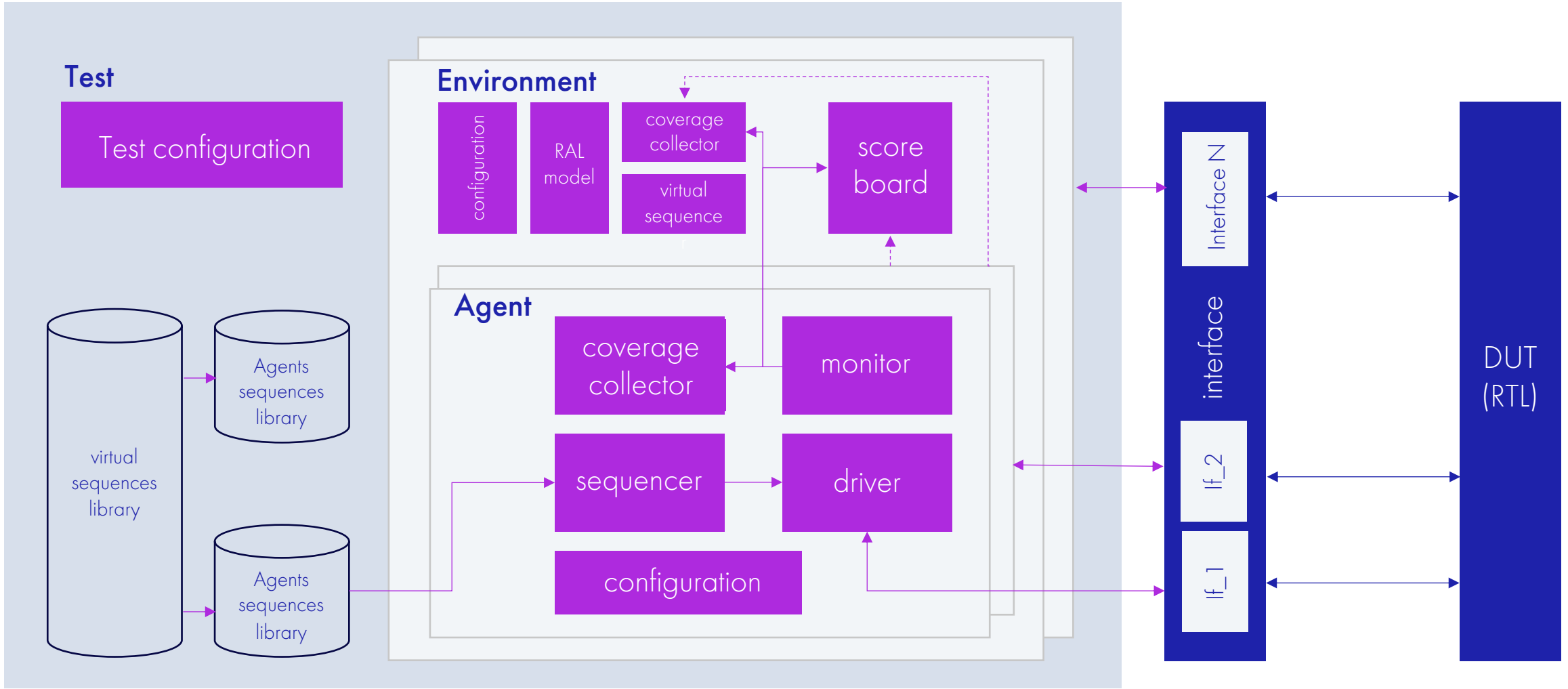
Фреймворк PyUVM – надстройка над cocotb

- Повторное использование кода;
- Стандартизация;
- Гибкость и конфигурируемость ;
- Структурированный подход ;
- Масштабируемость.



https://www.chipverify.com/tutorials/uvm

Архитектура UVM Testbench



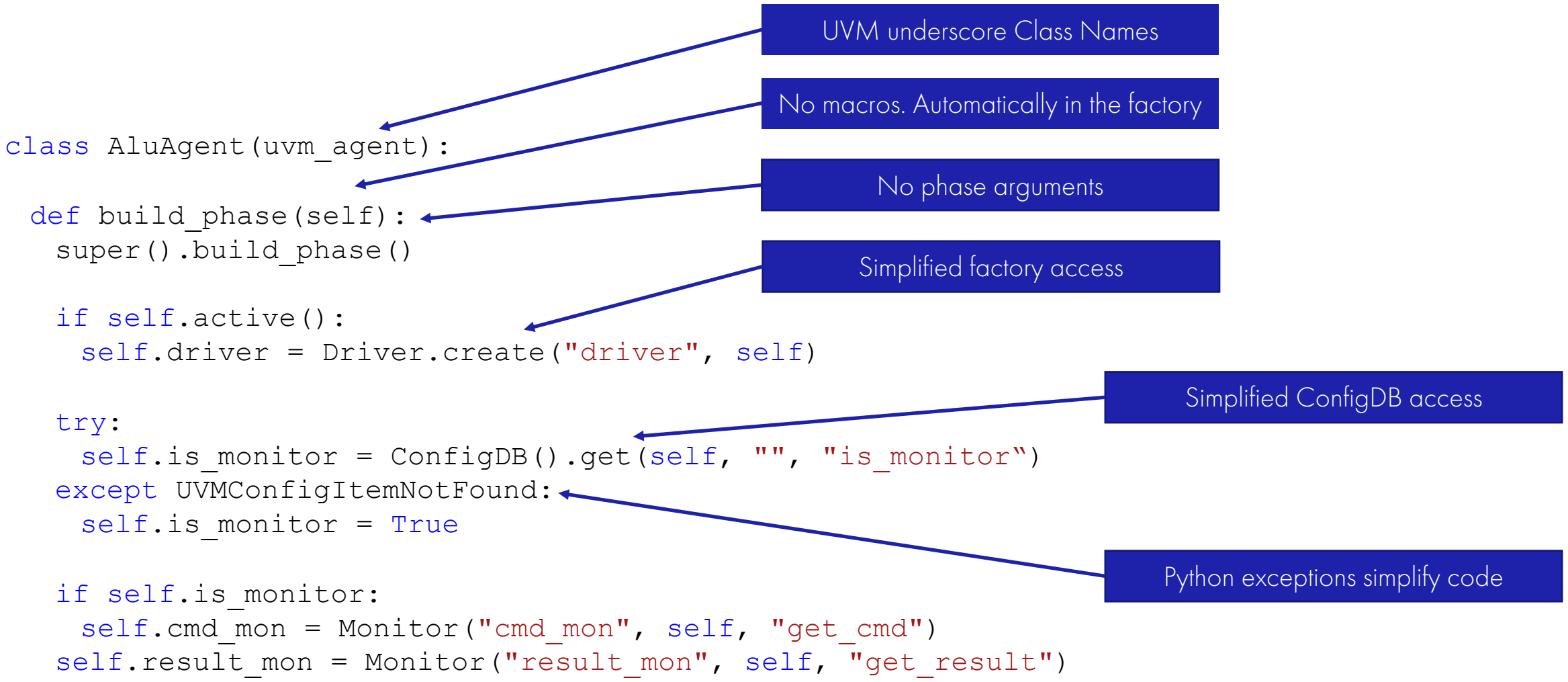
Фреймворк PyUVM

По мотивам стандарта UVM 1.2 IEEE Specification:

- Имплементирована общая, наиболее часто применяемая функциональность ;
- Не имплементированы Recording Classes, uvm_resource_db, _imp classes ...
- **Имплентирован RAL** 19/02/24

Section UVM 1.2 IEEE Specification	Name	Description
5	Base Classes	uvm_object does not capture transaction timing information
6	Reporting Classes	Leverages logging, controlled using UVM hierarchy
8	Factory Classes	All uvm_void classes automatically registered
9	Phasing	Simplified to only common phases. Supports objection system
12	UVM TLM Interfaces	Fully implemented
13	Predefined Component Classes	Implements uvm_component with hierarchy, uvm_root singleton, run_test(), simplified ConfigDB, uvm_driver, etc
14 & 15	Sequences, sequencer, sequence_item	Refactored sequencer functionality leveraging Python language capabilities. Simpler and more direct implementation

Фреймворк PyUVM



Фреймворк PyUVM

Использование фабрики:

SystemVerilog UVM

```
function void build_phase(uvm_phase phase);
     driver_h    = driver::type_id::create("driver_h",this);
     coverage_h  =  coverage::type_id::create ("coverage_h",this);
     scoreboard_h = scoreboard::type_id::create("scoreboard_h",this);
```

Python UVM

```
def build_phase(self):
    self.driver = Driver.create("driver", self)
    self.coverage = Coverage.create("coverage", self)
    self.scoreboard = Scoreboard.create("scoreboard", self)
```

Фреймворк PyUVM

ConfigDB - примитивная “singleton” реализация, без параметризации

SystemVerilog UVM

```
config_db#(uvm_sequencer)::set(null, "*", "SEQR", seqr)
if(!uvm_config_db #(uvm_sequencer)::get(this, "","SEQR",seqr))
  `uvm_fatal("Could not find sequencer") # Error out if not there
```

Python UVM

```
ConfigDB().set(None, "*", "SEQR",self.seqr)
self.seqr = ConfigDB().get(self, "", "SEQR") # Automatically errors out
```

Фреймворк PyUVM

Фазирование:

- Имплементированы только общие фазы с их стандартным порядком работы. UVM 1.2 Spec Section 9.8.1

```
uvm_common_phases = [
uvm_build_phase,
uvm_connect_phase,
uvm_end_of_elaboration_phase,
uvm_start_of_simulation_phase,
uvm_run_phase,
uvm_extract_phase,
uvm_check_phase,
uvm_report_phase,
uvm_final_phase]
```

Фреймворк PyUVM

Логирование:

- Фильтрация сообщений по уровню;
- Выбор фильтра сообщений для любого уровня иерархии;
- Запись в файл.

Level	Numeric value
CRITICAL	50
ERROR	40
WARNING	30
INFO	20
DEBUG	10
FIFO_DEBUG	5
NOTSET	0

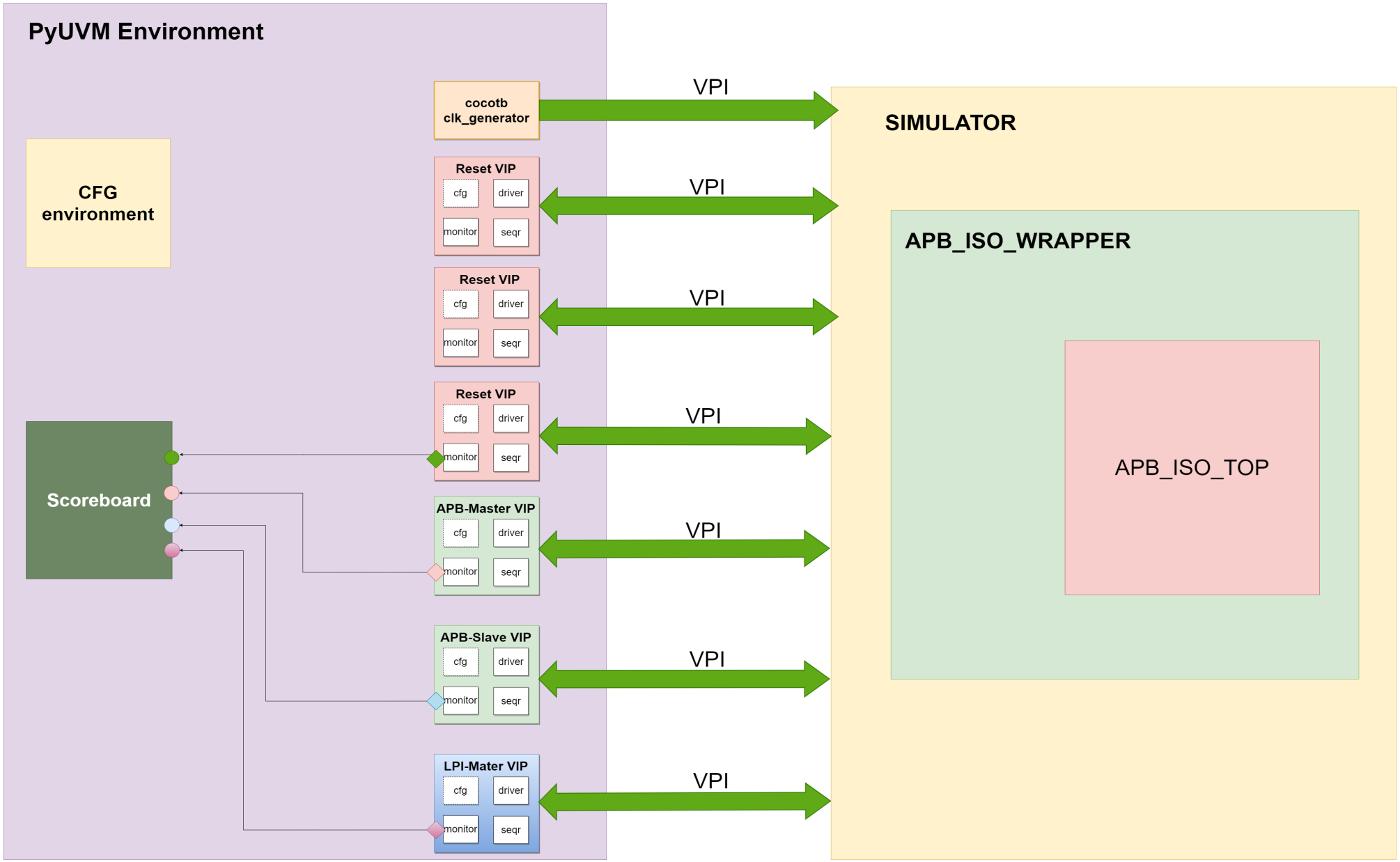
pyuvm addition

Every component has `self.logger`

```
class LogComp(uvm_component):
    def run_phase(self):
        self.raise_objection()
        self.logger.debug("This is debug")
        self.logger.info("This is info")
        self.logger.warning("This is warning")
        self.logger.error("This is error")
        self.logger.critical("This is critical")
        self.logger.log(FIFO_DEBUG, "This is a FIFO message")
        self.drop_objection()
```

```
DEBUG: <src_file>(6)[uvm_test_top.comp]: This is debug
INFO: <src_file>(7)[uvm_test_top.comp]: This is info
WARNING: <src_file>(8)[uvm_test_top.comp]: This is warning
ERROR: <src_file>(9)[uvm_test_top.comp]: This is error
CRITICAL: <src_file>(10)[uvm_test_top.comp]: This is critical
FIFO_DEBUG: <src_file>(11)[uvm_test_top.comp]: This is a FIFO message
```

Пример apb_iso_env на PyUVM



Фреймворк PyUVM: lpi_agent_cfg

SV:

```
class ymp_lpi_q_agent_cfg extends uvm_object;

  virtual ymp_lpi_q_if vif;
  bit is_active;
  bit print_debug_info = 1'b0;
  rand bit qreqn_state_during_rst;

  extern function new(string name = "ymp_lpi_q_agent_cfg");
  extern function void build();
  `uvm_object_utils(ymp_lpi_q_agent_cfg)

endclass


function ymp_lpi_q_agent_cfg::new(string name = "ymp_lpi_q_agent_cfg");
  super.new(name);
endfunction
```

Python:

```
@vsc.randobj
class lpi_master_agent_cfg (uvm_object):
  def __init__(self, name):
  super().__init__(name)
  self.is_active = 0
  self.print_debug_info = 0
  self.qreqn_state_during_rst = vsc.rand_bit_t()
```

Фреймворк PyUVM: lpi_agent

SV:

```
class ymp_lpi_q_agent extends uvm_agent;
   ymp_lpi_q_agent_cfg cfg;
   ymp_lpi_q_monitor   monitor;
   ymp_lpi_q_driver    driver;
   ymp_lpi_q_sequencer sequencer;

   extern function new(string name, uvm_component parent);
   extern function void build_phase(uvm_phase phase);
   extern function void connect_phase(uvm_phase phase);

   `uvm_component_utils(ymp_lpi_q_agent)
endclass

function ymp_lpi_q_agent::new(string name, uvm_component parent);
   super.new(name, parent);
endfunction

...

endclass
```

Python:

```
class lpi_master_agent(uvm_agent):
   num_id = 0

   def __init__(self, name, parent, *args, **kwargs):
      lpi_master_agent.num_id +=1
      self.num_id = lpi_master_agent.num_id
      super().__init__(name, parent)
```

Фреймворк PyUVM: lpi_agent

Python:

```
def build_phase(self):
    super().build_phase()
    self.cfg = ConfigDB().get(self, "", "cfg")
    if (self.cfg.is_active):
        self.sequencer = uvm_sequencer("lpi_master_seqr",self)
        self.driver = lpi_master_driver("lpi_master_driver",self,num_id = self.num_id)
        self.driver.set_cfg(self.cfg)
    self.monitor = lpi_monitor("lpi_monitor", self, num_id = self.num_id)
    self.monitor.set_cfg(self.cfg)
def connect_phase(self):
    if (self.cfg.is_active):
        self.driver.seq_item_port.connect(self.sequencer.seq_item_export)
```

Фреймворк PyUVM: lpi_driver

Python:

```
class lpi_master_driver(uvm_driver):
  def __init__(self, *args, **kwargs):
    self.num_id = kwargs["num_id"]
    del kwargs["num_id"]
    self.__flag_transaction_completed = 0
    self.__flag_transaction_started   = 0
    super().__init__(*args, **kwargs)

  def connect_phase (self):
    pre_index = "lpi" + "_vip_" + "m" + str(self.num_id - 1)
+ "_"
    self.qreqn  = getattr(cocotb.top, (pre_index + "qreqn"))
    self.qclk   = getattr(cocotb.top, (pre_index + "qreset"))
    self.qreset = getattr(cocotb.top, (pre_index + "qclk"))

  def __init_flag_to_zero(self):
    self.__flag_transaction_started = 0
    self.__flag_transaction_compl   = 0

  def set_cfg(self, cfg):
    self.__cfg = cfg
```

RTL-wrapper:

```
// LPI VIP master[0] wires
wire lpi_vip_m0_qreqn     ;
wire lpi_vip_m0_qreset    ;
wire lpi_vip_m0_qclk      ;
wire lpi_vip_m0_qacceptn  ;
wire lpi_vip_m0_qdeny     ;
wire lpi_vip_m0_qactive   ;
```

Фреймворк PyUVM: lpi_driver

SV: run_phase

```
task
ymp_lpi_q_driver::run_phase(uvm_phase
phase);
 forever begin
  @(negedge vif.rst);
  fork
   forever begin
     <...>
   end

   begin
    <...>
   end
  join_any

   disable fork;

   <...>
 end
endtask
```

Python: run_phase

```
async def __main_task(self):
  while True:
    <...>
```

```
async def __rst_task(self):
  <...>
```

```
async def run_phase(self):
  <...>
  while True:
    first_task = cocotb.start_soon(self.__main_task())
    second_task = cocotb.start_soon(self.__rst_task())
    await cocotb.triggers.First(first_task, second_task)
    <...>
```

Фреймворк PyUVM: driver

PyUVM: нет поддержки try_next_item()

```
class uvm_driver(uvm_component):
  def __init__(self, name, parent):
    super().__init__(name, parent)
    self.seq_item_port = uvm_seq_item_port("seq_item_port", self)
```

S14_15_python_sequnces.py

```
class uvm_seq_item_port(uvm_port_base):
  def connect(self, export):
    self.check_export(export)
    super().connect(export)

  async def put_req(self, item):
    ...

  def put_response(self, item):
    ...

  async def get_next_item(self):
    ...

  def item_done(self, rsp=None):
    ...

  async def get_response(self, transaction_id=None):
    ...
```

Фреймворк PyUVM: sequence_item

SV:

```
class ymp_lpi_q_item extends uvm_sequence_item;

   rand int pre_delay_clk;
   rand int post_delay_clk;
   ...



   extern function new(string name = "ymp_lpi_q_item");
   ...

   `uvm_object_utils(ymp_lpi_q_item)
endclass

function ymp_lpi_q_item::new(string name = "ymp_lpi_q_item");
   super.new(name);
endfunction

function void ymp_lpi_q_item::do_print(uvm_printer printer);
   printer.print_field_int("pre_delay_clk", pre_delay_clk,
$bits(pre_delay_clk));
   ...
endfunction

function ymp_lpi_q_states_e ymp_lpi_q_item::get_lpi_state();
   case ({state_req, state_resp})
      {NO_REQUEST, IDLE} : get_lpi_state = Q_RUN;
      ...
      default : get_lpi_state = ILLEGAL;
   endcase
endfunction
```

Python:

```
@vsc.randobj
class lpi_q_item(uvm_sequence_item):
   def __init__(self):
      super().__init__("lpi_q_item")

      self.pre_delay_clk = vsc.rand_int32_t()
      self.post_delay_clk = vsc.rand_int32_t()
      ...

   def __eq__(self, other): #do_compare analog
      return (self.state_req    == other.state_req  and
              self.state_resp   == other.state_resp and
              self.state_active == other.state_active)



   def __str__(self): #do_print analog
      return ((f"\nName: {self.get_full_name()} \
      ...
      \nstate_active: {self.state_active}"))

   def get_lpi_state(self):
      if (self.state_req == lpi_q_req_e.NO_REQUEST and
         self.state_resp == lpi_q_resp_e.IDLE):
         return lpi_q_state_e.Q_RUN.name
      ...
      else:
         return lpi_q_state_e.ILLEGAL.name
```

Фреймворк PyUVM: sequence

Python:

```
class lpi_master_direct_sequence(uvm_sequence):

 def __init__(self, name):
   super().__init__(name)

 async def body(self):
   item = lpi_q_item()
  with item.randomize_with() as it:
   it.state_req == lpi_q_req_e.REQUEST
  await self.start_item(item)
  await self.finish_item(item)
```

Фреймворк PyUVM: scoreboard

Python:

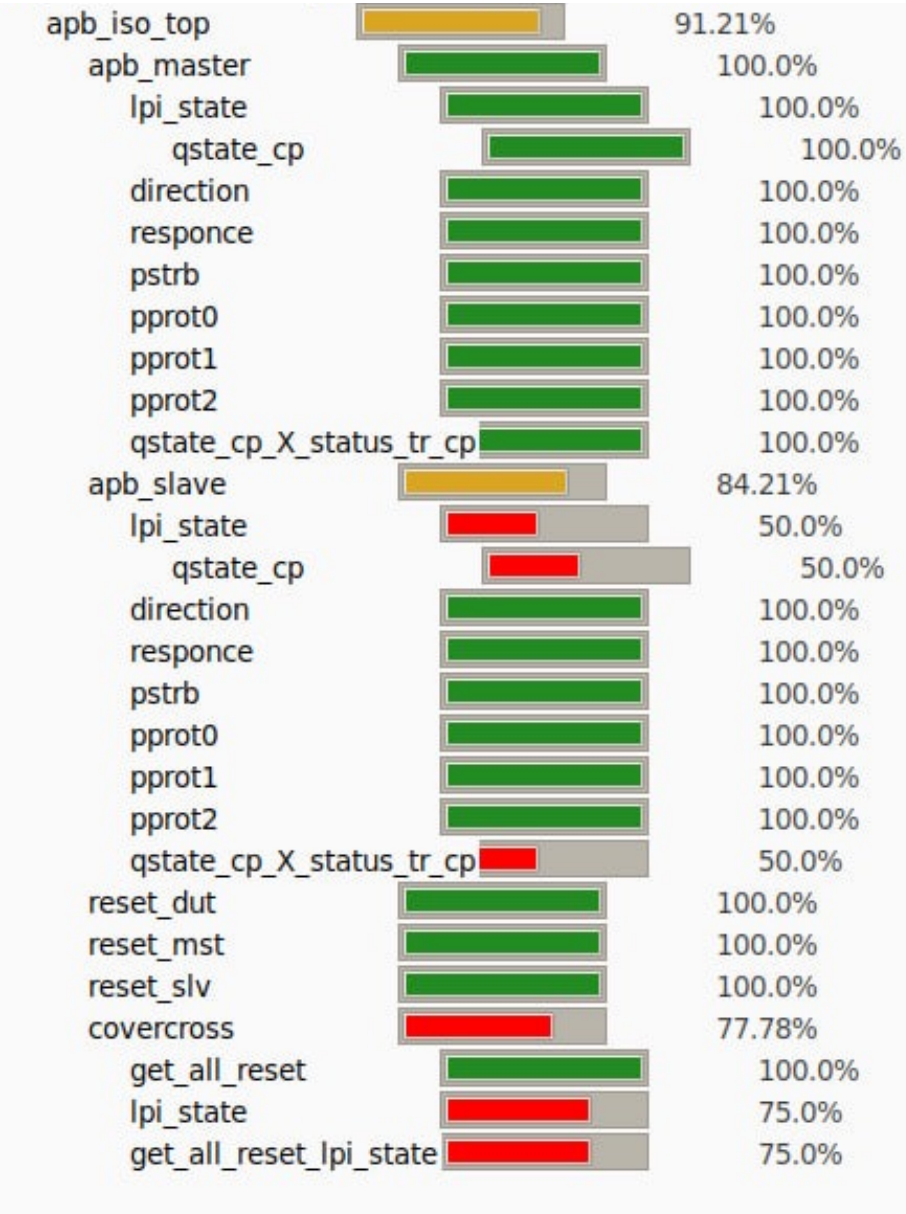
```
from apb_iso_pkg import *
class apb_iso_scoreboard (uvm_component):

 def __init__(self, name, parent):
  super().__init__(name, parent)
  self.passed = None
  self.apb_slv_trn = None

 def build_phase(self):
  self.dut_fifo_reset = uvm_tlm_analysis_fifo("dut_fifo_reset", self)
  self.slv_fifo_reset = uvm_tlm_analysis_fifo("slv_fifo_reset", self)
  self.mst_fifo_reset = uvm_tlm_analysis_fifo("mst_fifo_reset", self)
  self.mst_fifo_trans = uvm_tlm_analysis_fifo("mst_fifo_trans", self)
  self.slv_fifo_trans = uvm_tlm_analysis_fifo("slv_fifo_trans", self)
  self.lpi_fifo_trans = uvm_tlm_analysis_fifo("lpi_fifo_trans", self)

  <…>
```

Фреймворк PyUVM & cocotb_coverage

- Используются стандартные классы-сборщики покрытия - uvm_subscriber.
- Sample-функции обернуты в декораторы библиотеки cocotb_coverage.

```
class apb_iso_cov_collector(uvm_component):
  class uvm_AnalysisImp(uvm_analysis_export):
    def __init__(self, name, parent, write_fn):
      super().__init__(name, parent)
      self.write_fn = write_fn
    def write(self, tt):
      self.write_fn(tt)
  def __init__(self, name, parent):
    super().__init__(name, parent)
    self.analysis_export_master_rst = self.uvm_AnalysisImp("analysis_export_master_rst",self,
self.write_master_rst_cov)
    <…>
    def write_master_rst_cov(self, master_rst_item):
      self.sample_rst_mst(master_rst_item.rst_st.name)
    async def run_phase(self):
    <…>
    @CoverPoint("apb_iso_top.reset_mst", bins = [rst_state.RST_ASSERT.name, rst_state.RST_DEASSERT.name])
      def sample_rst_mst(self, a):
        pass
    <…>
```

Фреймворк PyUVM & cocotb_coverage

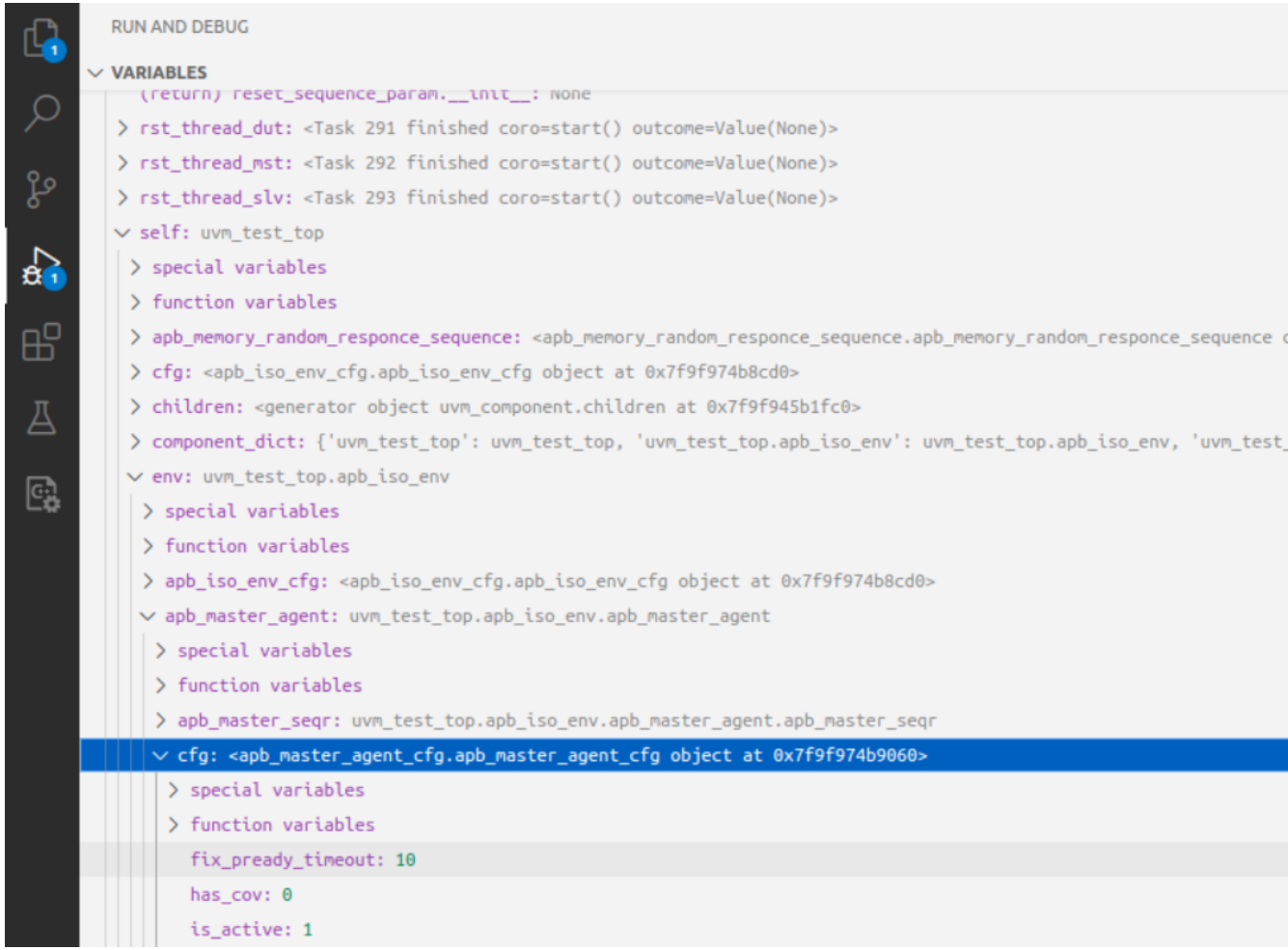


Python debugger in vscode

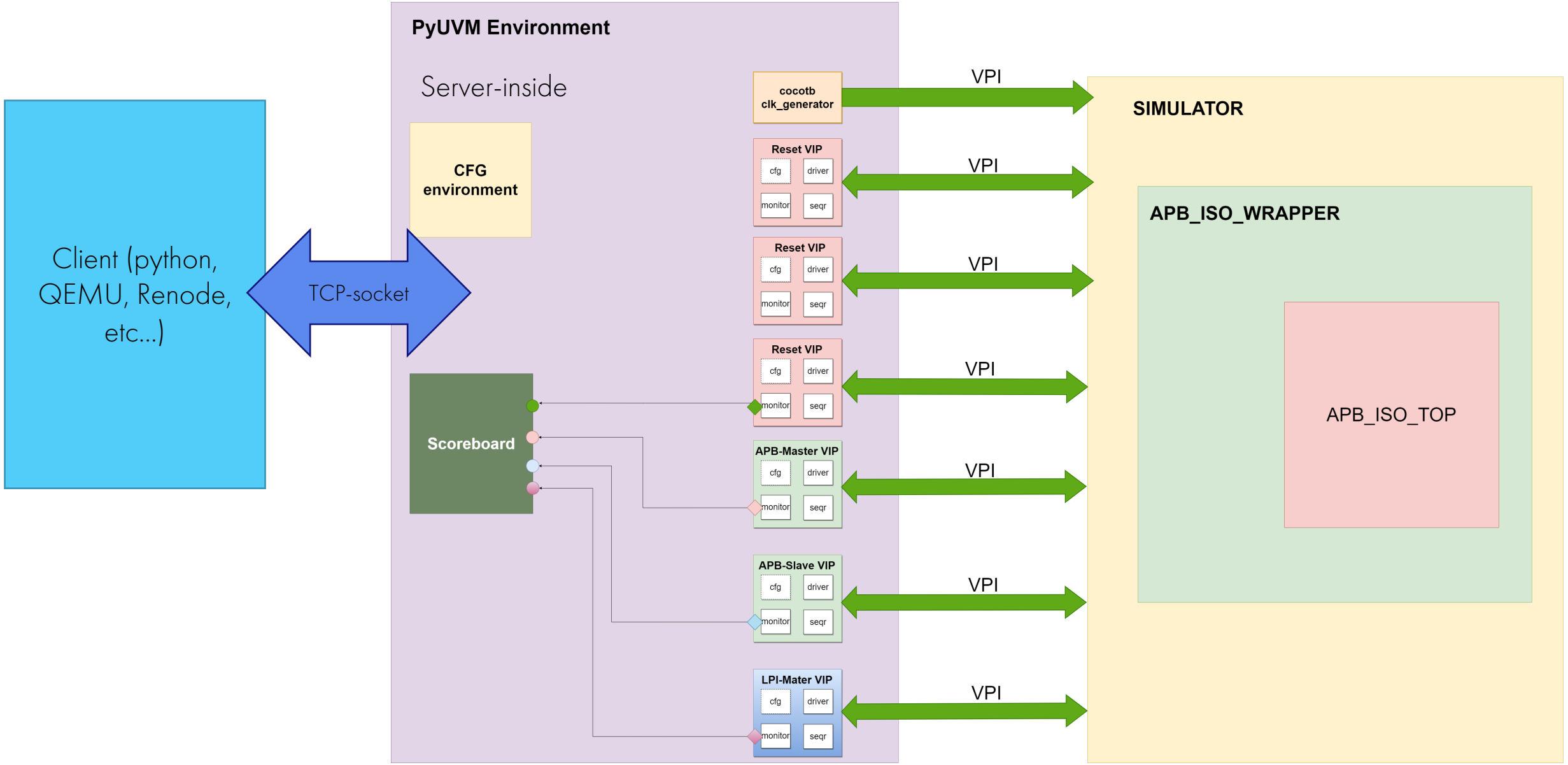
Стандартный отладчик vscode - debugpy

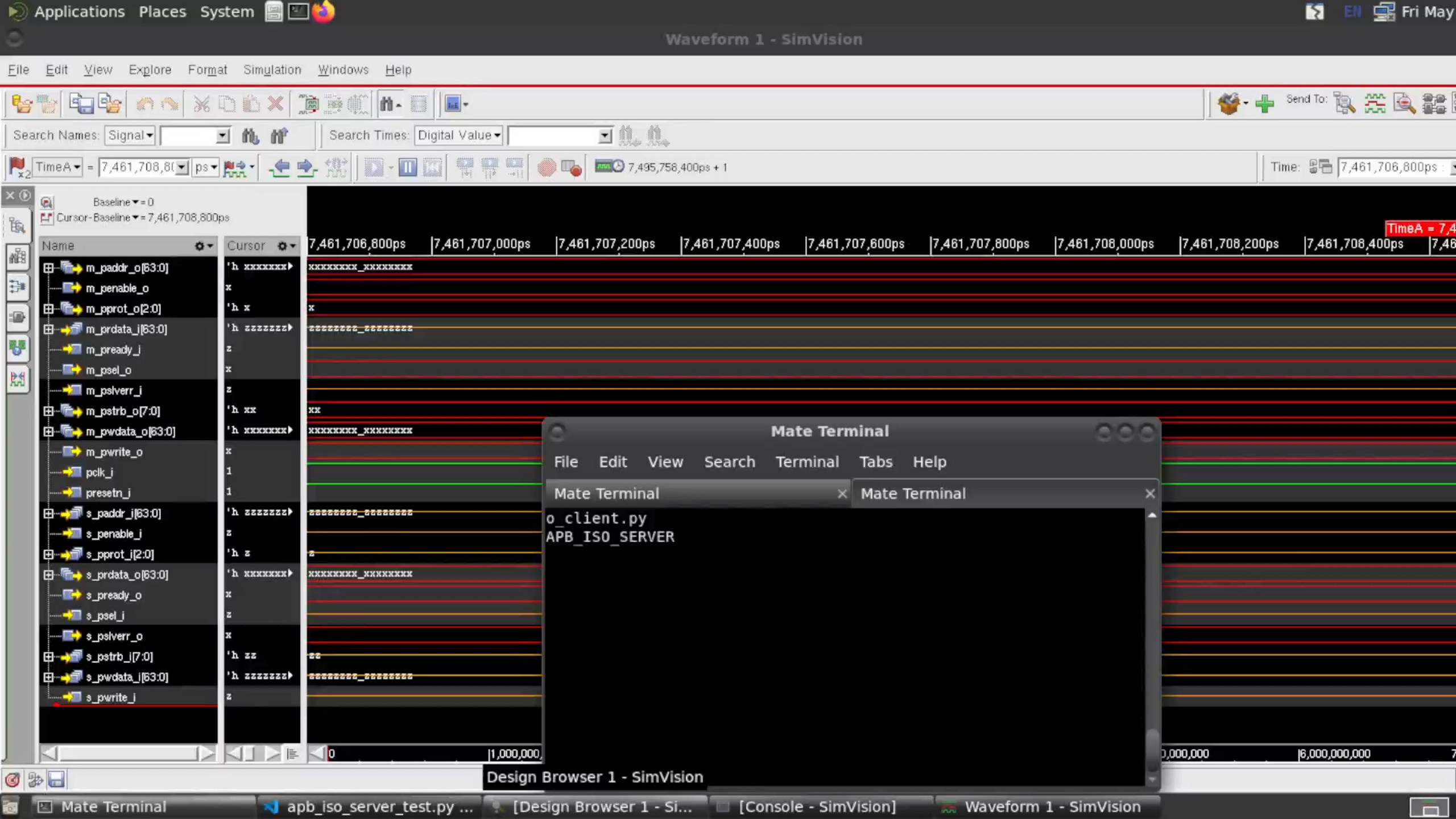
```
77
78      async def run_phase(self):
79
80          listen_host, listen_port = debugpy.listen(("localhost", 5679))
81          # Suspend execution until debugger attaches
82          debugpy.wait_for_client()
83          # Break into debugger for user control
84          breakpoint()  # or debugpy.breakpoint() on 3.6 and below
85          self.raise_objection()
86          print(cocotb.SIM_NAME)
87          cocotb.start_soon(self.apb_slave_responce())
88          cocotb.top.test_en.value = 0
                                                    "cocotb_clock")
                                                = 30
                                                = 20
93          min_delay_trn = 0
94          max_delay_trn = 2000
95
96          try:
97              num_pkts = int(cocotb.plusargs["num_pkts"])
98          except KeyError:
99              num_pkts = 1
100
```

Add Breakpoint
Add Conditional Breakpoint...
Add Logpoint...



QEMU/Renode with cocotb




Applications Places System
Waveform 1 - SimVision
File Edit View Explore Format Simulation Windows Help
Search Names: Signal
Search Times: Digital Value
Send To:
TimeA
7,495,758,400ps + 1
Time: 7,461,706,800ps
Baseline = 0
Cursor-Baseline = 7,461,708,800ps
Name
Cursor
m_paddr_o[63:0]
m_penable_o
m_pprot_o[2:0]
m_prdata_i[63:0]
m_pready_i
m_psel_o
m_pslverr_i
m_pstrb_o[7:0]
m_pwdata_o[63:0]
m_pwrite_o
pclk_i
presetn_i
s_paddr_i[63:0]
s_penable_i
s_pprot_i[2:0]
s_prdata_o[63:0]
s_pready_o
s_psel_i
s_pslverr_o
s_pstrb_i[7:0]
s_pwdata_i[63:0]
s_pwrite_i
7,461,706,800ps
7,461,707,000ps
7,461,707,200ps
7,461,707,400ps
7,461,707,600ps
7,461,707,800ps
7,461,708,000ps
7,461,708,200ps
7,461,708,400ps
TimeA = 7,4
Mate Terminal
File Edit View Search Terminal Tabs Help
Mate Terminal
Mate Terminal
o_client.py
APB_ISO_SERVER
Design Browser 1 - SimVision
Mate Terminal
apb_iso_server_test.py ...
[Design Browser 1 - Si...
[Console - SimVision]
Waveform 1 - SimVision

Опыт использования:

Проблемы с которыми столкнулся:

- Отсутствие try_next_item() в driver class (nonblocking) ;
- Отсутствие покрытия, рандомизации, констрейнов, как встроенного функционала.

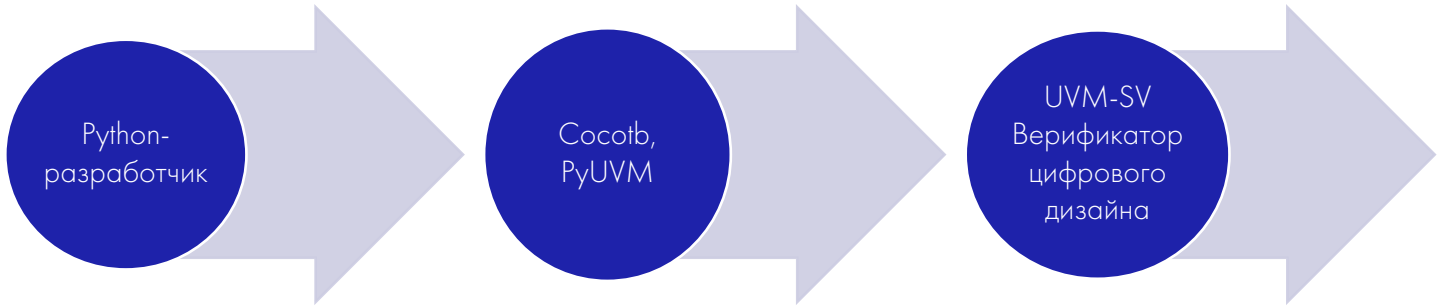
  Решение – добавить фреймворк **cocotb-coverage** - (https://github.com/mciepluc/cocotb-coverage) и **pyvsc** - (https://github.com/fvutils/pyvsc/);

- Низкая производительность.

Тест - apb_iso_all_random_reset SIM_OPTS="+num_pkts=50"		
	RAM	execution time
Pyuvm + Cadence	90620 KB	0m:40.56s
UVM SV + Cadence	719680 KB	0m:24.55s

Выводы, преимущества PyUVM:

1. Кроссплатформенность;
2. Open-source friendly – использование как бесплатных симуляторов (Icarus Verilog, Verilator), так и проприетарных (Synopsys, Cadence, Mentor Graphics), множество открытых библиотек;
3. Легкий старт с языком Python ;
4. Возможность переиспользования окружения на системном уровне;
5. Развитие Python-разработчика до UVM-SV верификатора.



ИМПУЛЬС

Инженер по разработке системного ПО для систем на кристалле (SoC/BSP) архитектуры RISC-V

Инженер-исследователь по разработке алгоритмов ЦОС (AME)

DevOps инженер

Инженер по проектированию аналогово-цифровых печатных плат

Инженер по верификации СнК (Python)

Инженер по оптимизации платформ под ПО искусственного интеллекта

Инженер по верификации СнК (SystemVerilog/UVM)

Инженер по разработке ПО для базовых станций (Go, C++, Python)

Инженер по машинному обучению

Инженер по разработке программных средств анализа производительности

Инженер по системному программированию СнК

Инженер по разработке ПО компьютерного зрения

Инженер по разработке ПО BMC

Инженер по разработке ПО BIOS UEFI

Инженер-схемотехник цифровых устройств

Инженер по RTL проектированию

Инженер-конструктор печатных плат

Инженер по FPGA прототипированию

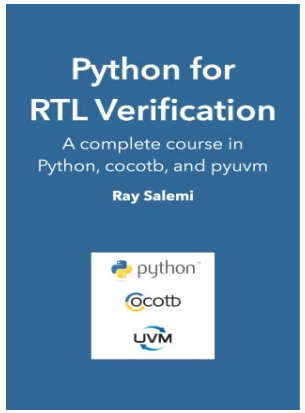
Инженер по автоматизации тестирования

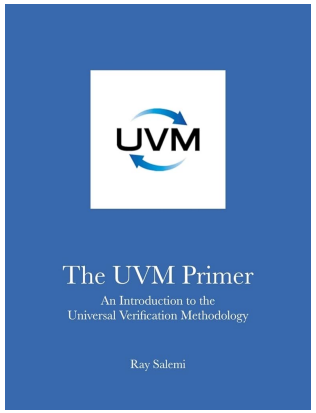
//careers.yadro.com/impulse

Git repo:

https://github.com/hurisson/pyuvm_primer

Литература:





- https://pythonworld.ru/samouchitel-python
- https://docs.cocotb.org/en/stable/
- Python for RTL Verification: A complete course in Python, cocotb, and pyuvm, Ray Salemi

Спасибо за внимание!


БУДУЩЕЕ
В НАШИХ
РУКАХ

Партнеры конференции

Наши ресурсы

Как найти сообщество

FPGA-Systems.ru

FPGA-Systems Magazine (FSM)

@fpgasystems

admin@fpga-systems.ru

Youtube

@fpgasystems