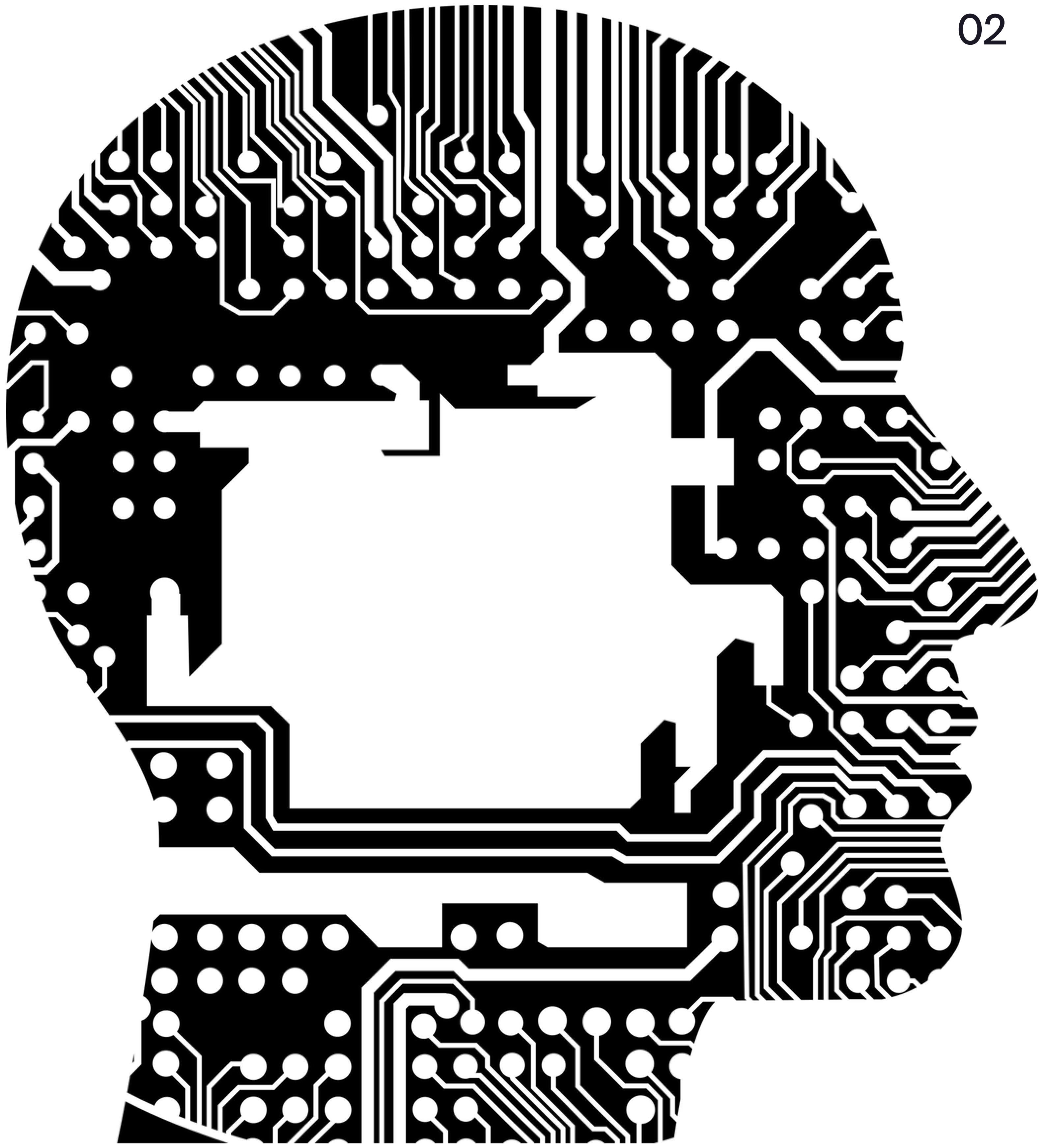
USING COCOTB AS A VERIFICATION FRAMEWORK FOR HDL DESIGNS

Magdalena Król
Supervisor: Maciej Kopeć

Introduction

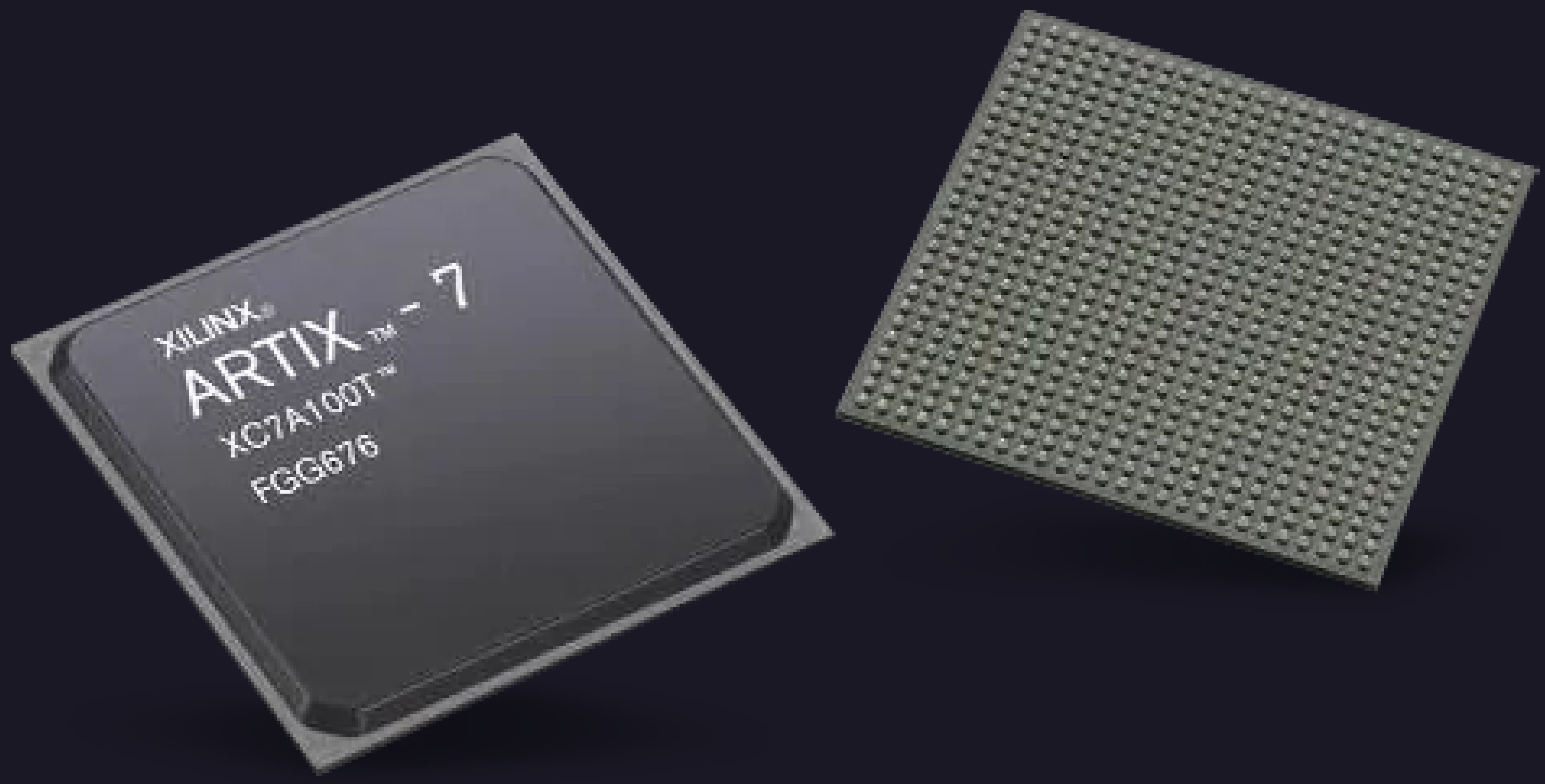
What is the project about

What were my tasks



FPGA

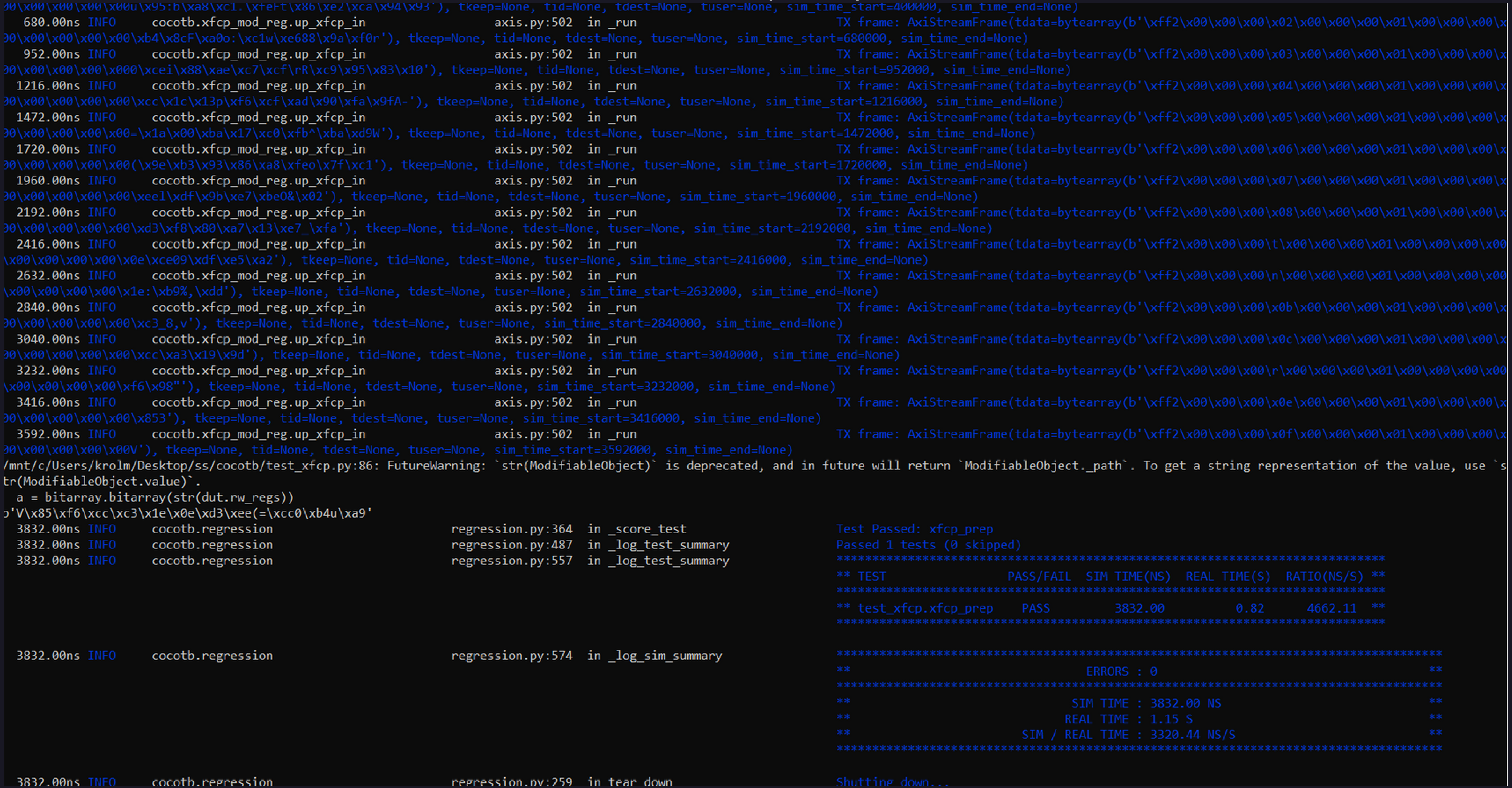
A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence the term "field-programmable".



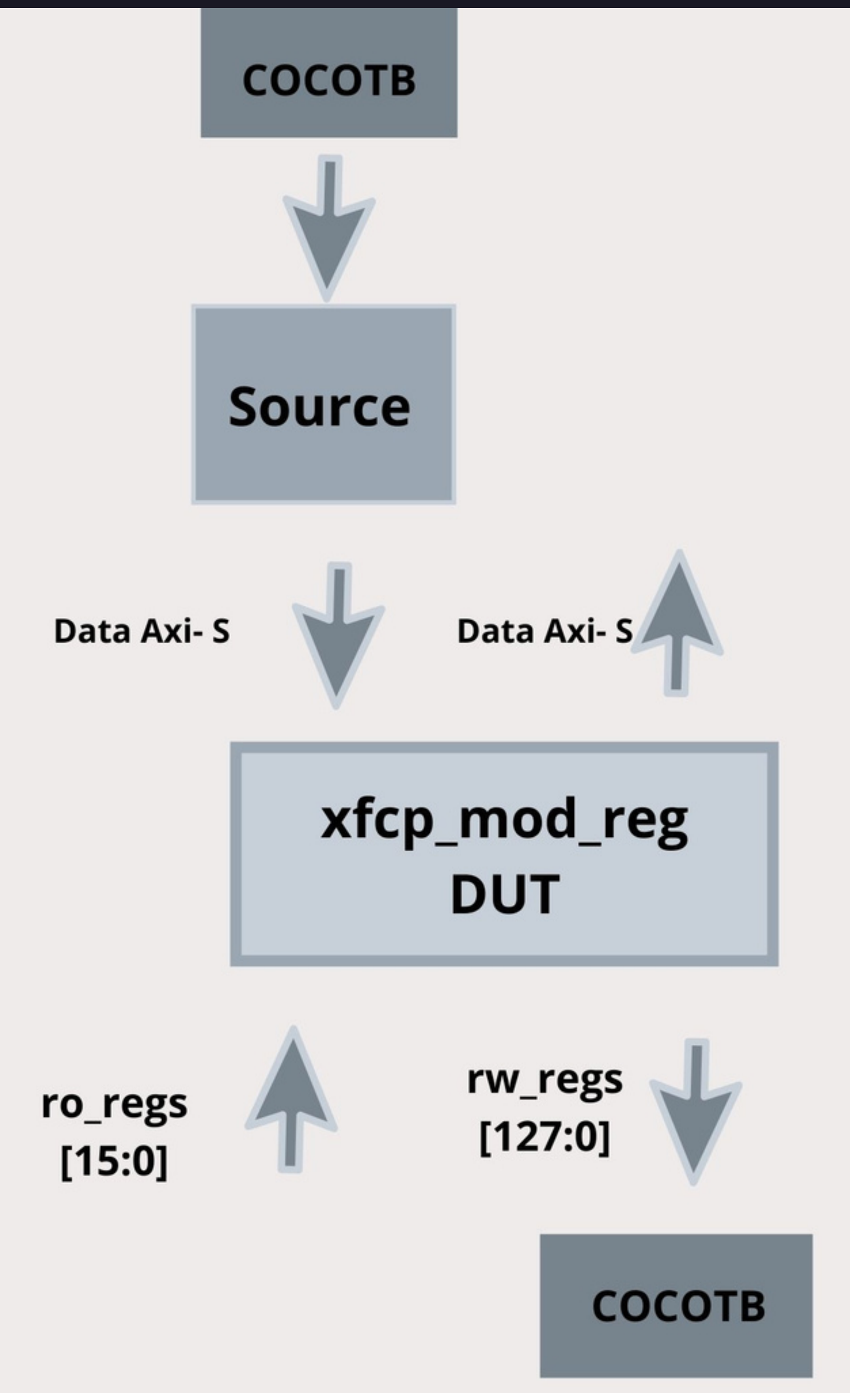
https://pl.mouser.com/images/marketingid/2019/img/148373317.png?v=022321.0507

Cocotb

cocotb is an open-source COroutine-based COsimuation TestBench (hence the name) environment, developed for the verification of hardware description languages like VHDL, Verilog and SystemVerilog, by means of Python programming language in the field of electronic design automation (EDA).

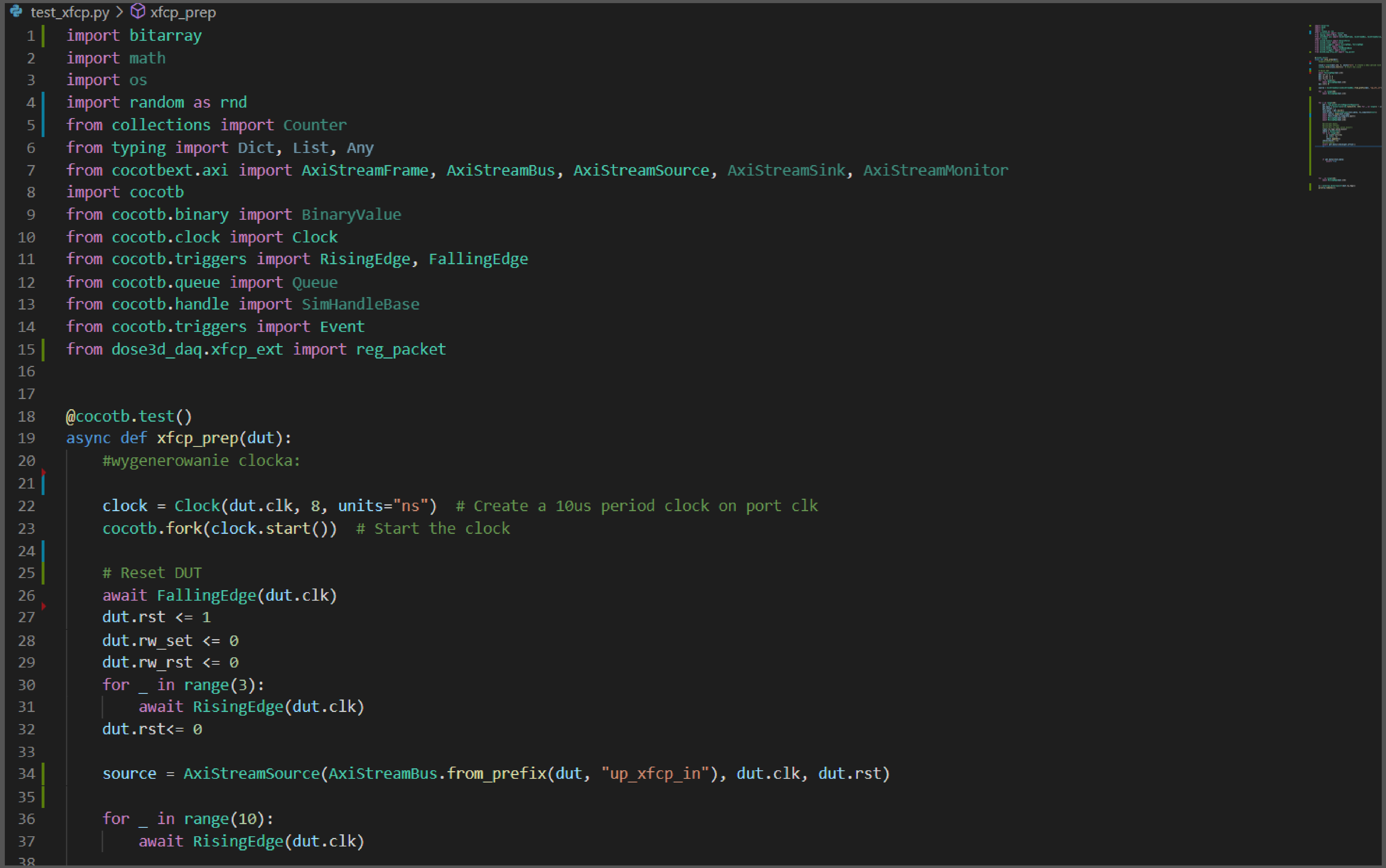


```
00\x00\x00\x00\x00u\x95:b\xa8\xc1.\xfeFt\x86\xe2\xca\x94\x93'), tkeep=None, tid=None, tdest=None, tuser=None, sim_time_start=400000, sim_time_end=None)
  680.00ns INFO     cocotb.xfcp_mod_reg.up_xfcp_in                    axis.py:502  in _run                       TX frame: AxiStreamFrame(tdata=bytearray(b'\xff2\x00\x00\x00\x02\x00\x00\x00\x01\x00\x00\x00\x
00\x00\x00\x00\x00\xb4\x8cF\xa0o:\xc1w\xe688\x9a\xf0r'), tkeep=None, tid=None, tdest=None, tuser=None, sim_time_start=680000, sim_time_end=None)
  952.00ns INFO     cocotb.xfcp_mod_reg.up_xfcp_in                    axis.py:502  in _run                       TX frame: AxiStreamFrame(tdata=bytearray(b'\xff2\x00\x00\x00\x03\x00\x00\x00\x01\x00\x00\x00\x
00\x00\x00\x00\x000\xcei\x88\xae\xc7\xcf\rR\xc9\x95\x83\x10'), tkeep=None, tid=None, tdest=None, tuser=None, sim_time_start=952000, sim_time_end=None)
 1216.00ns INFO     cocotb.xfcp_mod_reg.up_xfcp_in                    axis.py:502  in _run                       TX frame: AxiStreamFrame(tdata=bytearray(b'\xff2\x00\x00\x00\x04\x00\x00\x00\x01\x00\x00\x00\x
00\x00\x00\x00\x00\xcc\x1c\x13p\xf6\xcf\xad\x90\xfa\x9fA-'), tkeep=None, tid=None, tdest=None, tuser=None, sim_time_start=1216000, sim_time_end=None)
 1472.00ns INFO     cocotb.xfcp_mod_reg.up_xfcp_in                    axis.py:502  in _run                       TX frame: AxiStreamFrame(tdata=bytearray(b'\xff2\x00\x00\x00\x05\x00\x00\x00\x01\x00\x00\x00\x
00\x00\x00\x00\x00=\x1a\x00\xba\x17\xc0\xfb^\xba\xd9W'), tkeep=None, tid=None, tdest=None, tuser=None, sim_time_start=1472000, sim_time_end=None)
 1720.00ns INFO     cocotb.xfcp_mod_reg.up_xfcp_in                    axis.py:502  in _run                       TX frame: AxiStreamFrame(tdata=bytearray(b'\xff2\x00\x00\x00\x06\x00\x00\x00\x01\x00\x00\x00\x
00\x00\x00\x00\x00(\x9e\xb3\x93\x86\xa8\xfeo\x7f\xc1'), tkeep=None, tid=None, tdest=None, tuser=None, sim_time_start=1720000, sim_time_end=None)
 1960.00ns INFO     cocotb.xfcp_mod_reg.up_xfcp_in                    axis.py:502  in _run                       TX frame: AxiStreamFrame(tdata=bytearray(b'\xff2\x00\x00\x00\x07\x00\x00\x00\x01\x00\x00\x00\x
00\x00\x00\x00\x00\xeel\xdf\x9b\xe7\xbeO&\x02'), tkeep=None, tid=None, tdest=None, tuser=None, sim_time_start=1960000, sim_time_end=None)
 2192.00ns INFO     cocotb.xfcp_mod_reg.up_xfcp_in                    axis.py:502  in _run                       TX frame: AxiStreamFrame(tdata=bytearray(b'\xff2\x00\x00\x00\x08\x00\x00\x00\x01\x00\x00\x00\x
00\x00\x00\x00\x00\xd3\xf8\x80\xa7\x13\xe7_\xfa'), tkeep=None, tid=None, tdest=None, tuser=None, sim_time_start=2192000, sim_time_end=None)
 2416.00ns INFO     cocotb.xfcp_mod_reg.up_xfcp_in                    axis.py:502  in _run                       TX frame: AxiStreamFrame(tdata=bytearray(b'\xff2\x00\x00\x00\t\x00\x00\x00\x01\x00\x00\x00\x00
\x00\x00\x00\x00\x0e\xce09\xdf\xe5\xa2'), tkeep=None, tid=None, tdest=None, tuser=None, sim_time_start=2416000, sim_time_end=None)
 2632.00ns INFO     cocotb.xfcp_mod_reg.up_xfcp_in                    axis.py:502  in _run                       TX frame: AxiStreamFrame(tdata=bytearray(b'\xff2\x00\x00\x00\n\x00\x00\x00\x01\x00\x00\x00\x00
\x00\x00\x00\x00\x1e:\xb9%,\xdd'), tkeep=None, tid=None, tdest=None, tuser=None, sim_time_start=2632000, sim_time_end=None)
 2840.00ns INFO     cocotb.xfcp_mod_reg.up_xfcp_in                    axis.py:502  in _run                       TX frame: AxiStreamFrame(tdata=bytearray(b'\xff2\x00\x00\x00\x0b\x00\x00\x00\x01\x00\x00\x00\x
00\x00\x00\x00\x00\xc3_8,v'), tkeep=None, tid=None, tdest=None, tuser=None, sim_time_start=2840000, sim_time_end=None)
 3040.00ns INFO     cocotb.xfcp_mod_reg.up_xfcp_in                    axis.py:502  in _run                       TX frame: AxiStreamFrame(tdata=bytearray(b'\xff2\x00\x00\x00\x0c\x00\x00\x00\x01\x00\x00\x00\x
00\x00\x00\x00\x00\xcc\xa3\x19\x9d'), tkeep=None, tid=None, tdest=None, tuser=None, sim_time_start=3040000, sim_time_end=None)
 3232.00ns INFO     cocotb.xfcp_mod_reg.up_xfcp_in                    axis.py:502  in _run                       TX frame: AxiStreamFrame(tdata=bytearray(b'\xff2\x00\x00\x00\r\x00\x00\x00\x01\x00\x00\x00\x00
\x00\x00\x00\x00\xf6\x98"'), tkeep=None, tid=None, tdest=None, tuser=None, sim_time_start=3232000, sim_time_end=None)
 3416.00ns INFO     cocotb.xfcp_mod_reg.up_xfcp_in                    axis.py:502  in _run                       TX frame: AxiStreamFrame(tdata=bytearray(b'\xff2\x00\x00\x00\x0e\x00\x00\x00\x01\x00\x00\x00\x
00\x00\x00\x00\x00\x853'), tkeep=None, tid=None, tdest=None, tuser=None, sim_time_start=3416000, sim_time_end=None)
 3592.00ns INFO     cocotb.xfcp_mod_reg.up_xfcp_in                    axis.py:502  in _run                       TX frame: AxiStreamFrame(tdata=bytearray(b'\xff2\x00\x00\x00\x0f\x00\x00\x00\x01\x00\x00\x00\x
00\x00\x00\x00\x00V'), tkeep=None, tid=None, tdest=None, tuser=None, sim_time_start=3592000, sim_time_end=None)
/mnt/c/Users/krolm/Desktop/ss/cocotb/test_xfcp.py:86: FutureWarning: `str(ModifiableObject)` is deprecated, and in future will return `ModifiableObject._path`. To get a string representation of the value, use `s
tr(ModifiableObject.value)`.
  a = bitarray.bitarray(str(dut.rw_regs))
b'V\x85\xf6\xcc\xc3\x1e\x0e\xd3\xee(=\xcc0\xb4u\xa9'
 3832.00ns INFO     cocotb.regression                         regression.py:364  in _score_test                Test Passed: xfcp_prep
 3832.00ns INFO     cocotb.regression                         regression.py:487  in _log_test_summary          Passed 1 tests (0 skipped)
 3832.00ns INFO     cocotb.regression                         regression.py:557  in _log_test_summary          ******************************************************************************
                                                                                                                ** TEST                    PASS/FAIL  SIM TIME(NS)  REAL TIME(S)  RATIO(NS/S) **
                                                                                                                ******************************************************************************
                                                                                                                ** test_xfcp.xfcp_prep      PASS         3832.00          0.82       4662.11  **
                                                                                                                ******************************************************************************

 3832.00ns INFO     cocotb.regression                         regression.py:574  in _log_sim_summary           ***************************************************************************************
                                                                                                                **                                 ERRORS : 0                                      **
                                                                                                                ***************************************************************************************
                                                                                                                **                               SIM TIME : 3832.00 NS                             **
                                                                                                                **                              REAL TIME : 1.15 S                                 **
                                                                                                                **                        SIM / REAL TIME : 3320.44 NS/S                           **
                                                                                                                ***************************************************************************************

 3832.00ns INFO     cocotb.regression                         regression.py:259  in tear_down                  Shutting down...
```

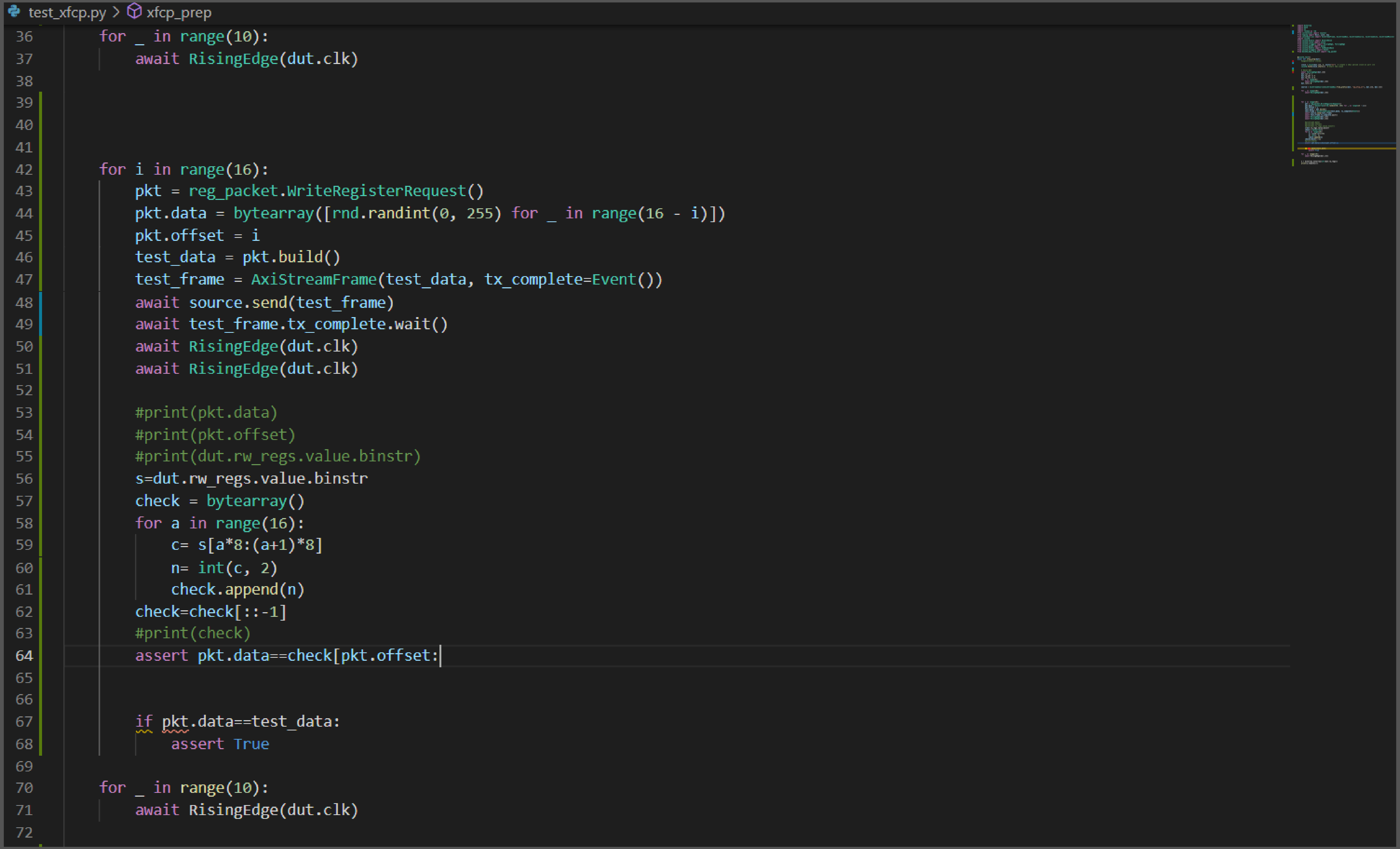


Write register requests

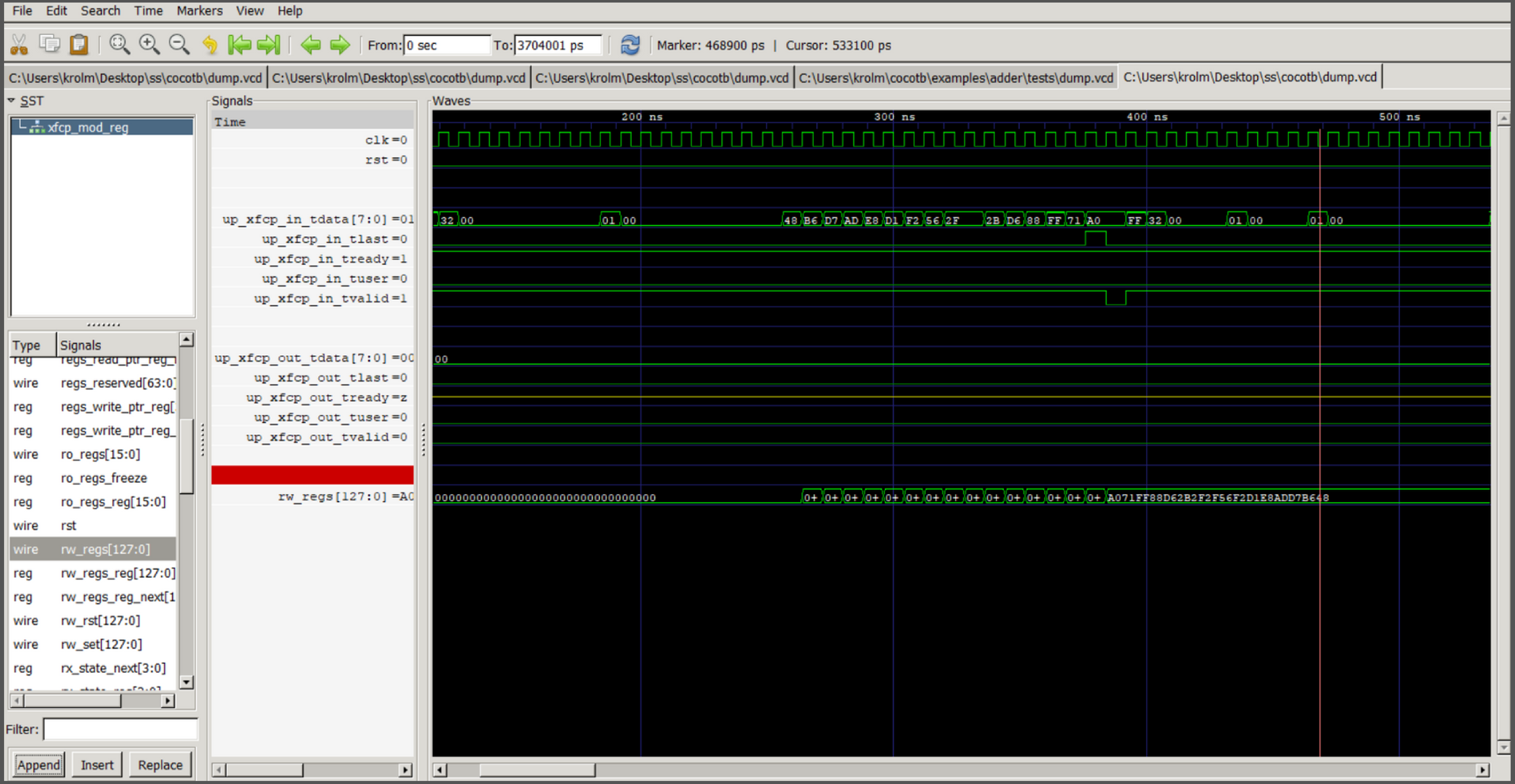
-offset
-data


test_xfcp.py > xfcp_prep

```python
import bitarray
import math
import os
import random as rnd
from collections import Counter
from typing import Dict, List, Any
from cocotbext.axi import AxiStreamFrame, AxiStreamBus, AxiStreamSource, AxiStreamSink, AxiStreamMonitor
import cocotb
from cocotb.binary import BinaryValue
from cocotb.clock import Clock
from cocotb.triggers import RisingEdge, FallingEdge
from cocotb.queue import Queue
from cocotb.handle import SimHandleBase
from cocotb.triggers import Event
from dose3d_daq.xfcp_ext import reg_packet


@cocotb.test()
async def xfcp_prep(dut):
    #wygenerowanie clocka:

    clock = Clock(dut.clk, 8, units="ns")  # Create a 10us period clock on port clk
    cocotb.fork(clock.start())  # Start the clock

    # Reset DUT
    await FallingEdge(dut.clk)
    dut.rst <= 1
    dut.rw_set <= 0
    dut.rw_rst <= 0
    for _ in range(3):
        await RisingEdge(dut.clk)
    dut.rst<= 0

    source = AxiStreamSource(AxiStreamBus.from_prefix(dut, "up_xfcp_in"), dut.clk, dut.rst)

    for _ in range(10):
        await RisingEdge(dut.clk)
```


test_xfcp.py > xfcp_prep

```python
    for _ in range(10):
        await RisingEdge(dut.clk)




    for i in range(16):
        pkt = reg_packet.WriteRegisterRequest()
        pkt.data = bytearray([rnd.randint(0, 255) for _ in range(16 - i)])
        pkt.offset = i
        test_data = pkt.build()
        test_frame = AxiStreamFrame(test_data, tx_complete=Event())
        await source.send(test_frame)
        await test_frame.tx_complete.wait()
        await RisingEdge(dut.clk)
        await RisingEdge(dut.clk)

        #print(pkt.data)
        #print(pkt.offset)
        #print(dut.rw_regs.value.binstr)
        s=dut.rw_regs.value.binstr
        check = bytearray()
        for a in range(16):
            c= s[a*8:(a+1)*8]
            n= int(c, 2)
            check.append(n)
        check=check[::-1]
        #print(check)
        assert pkt.data==check[pkt.offset:


        if pkt.data==test_data:
            assert True

    for _ in range(10):
        await RisingEdge(dut.clk)

```


File
Edit
Search
Time
Markers
View
Help
From: 0 sec
To: 3704001 ps
Marker: 468900 ps | Cursor: 533100 ps
C:\Users\krolm\Desktop\ss\cocotb\dump.vcd
C:\Users\krolm\Desktop\ss\cocotb\dump.vcd
C:\Users\krolm\Desktop\ss\cocotb\dump.vcd
C:\Users\krolm\cocotb\examples\adder\tests\dump.vcd
C:\Users\krolm\Desktop\ss\cocotb\dump.vcd
SST
xfcp_mod_reg
Signals
Waves
Filter:
Append
Insert
Replace

Type	Signals
reg	regs_read_ptr_reg_
wire	regs_reserved[63:0]
reg	regs_write_ptr_reg[
reg	regs_write_ptr_reg_
wire	ro_regs[15:0]
reg	ro_regs_freeze
reg	ro_regs_reg[15:0]
wire	rst
wire	rw_regs[127:0]
reg	rw_regs_reg[127:0]
reg	rw_regs_reg_next[1
wire	rw_rst[127:0]
wire	rw_set[127:0]
reg	rx_state_next[3:0]

```
Time                            200 ns                300 ns                400 ns                500 ns
clk=0
rst=0

up_xfcp_in_tdata[7:0]=01        32 00  01 00  48 B6 D7 AD E8 D1 F2 56 2F  2B D6 88 FF 71 A0  FF 32 00  01 00  01 00
up_xfcp_in_tlast=0
up_xfcp_in_tready=1
up_xfcp_in_tuser=0
up_xfcp_in_tvalid=1

up_xfcp_out_tdata[7:0]=00       00
up_xfcp_out_tlast=0
up_xfcp_out_tready=z
up_xfcp_out_tuser=0
up_xfcp_out_tvalid=0

rw_regs[127:0]=A0               00000000000000000000000000000000  0+ 0+ 0+ 0+ 0+ 0+ 0+ 0+ 0+ 0+ 0+ 0+ 0+ 0+ 0+  A071FF88D62B2F2F56F2D1E8ADD7B648
```

The end.

Thank you for your attention