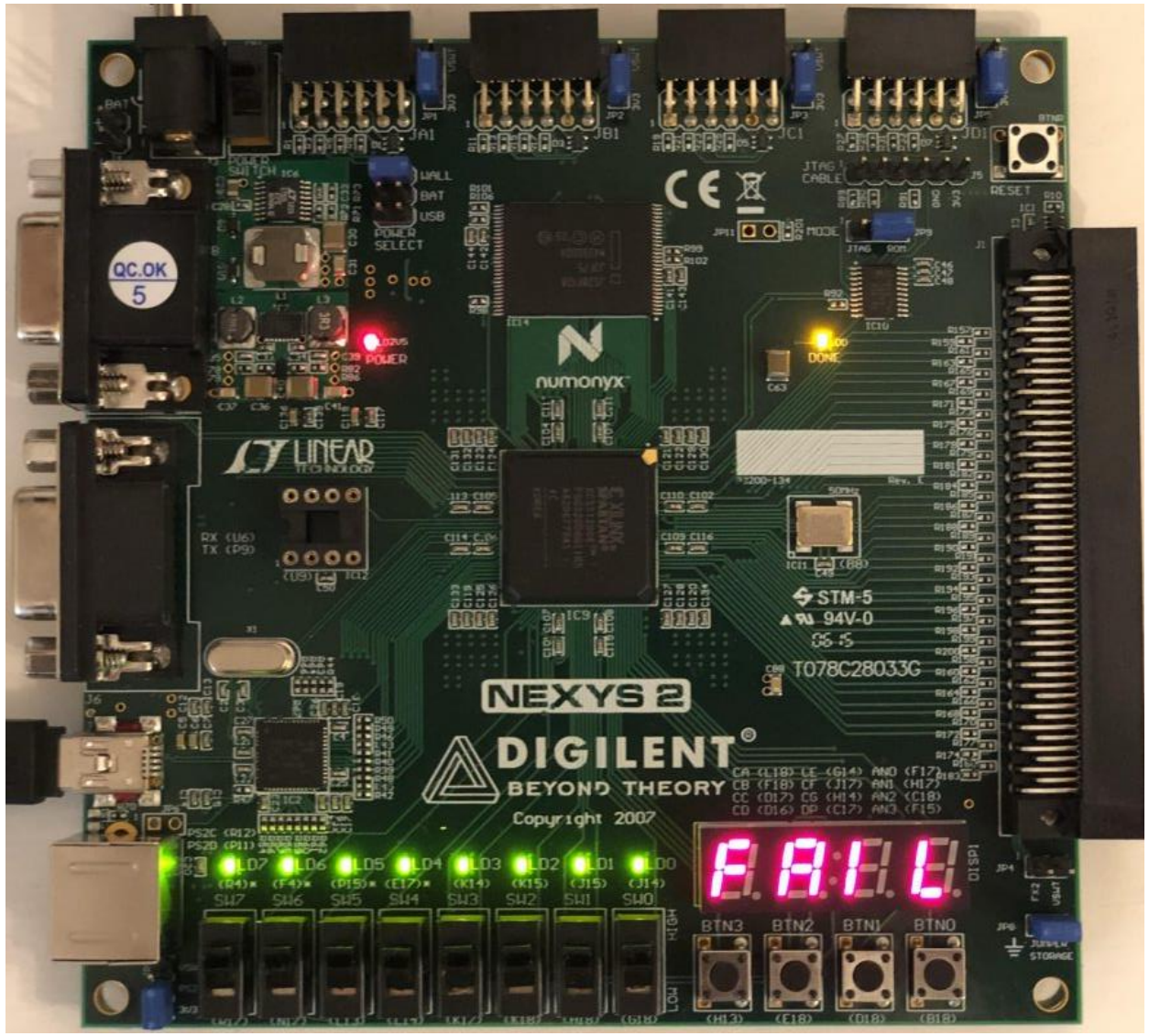
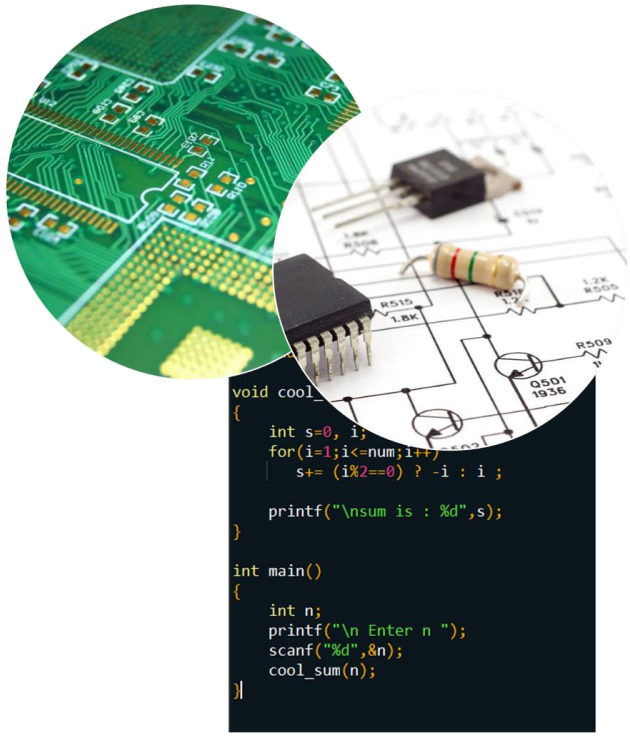
Where's the logic in VHDL verification?



ALTEN

17 MEI 2022
EVOLUON EINDHOVEN
D&E event 2022

Introduction



Electronics Engineering
Saxion Enschede

2012

Measurement System Specialist
High Power Laboratory
DNV KEMA

2013

Embedded Systems Engineer
BD Kiestra

2015

Consultant
Alten

2015

Panalytical
Firmware Engineer

2016

Kulicke & Soffa
Embedded System Engineer

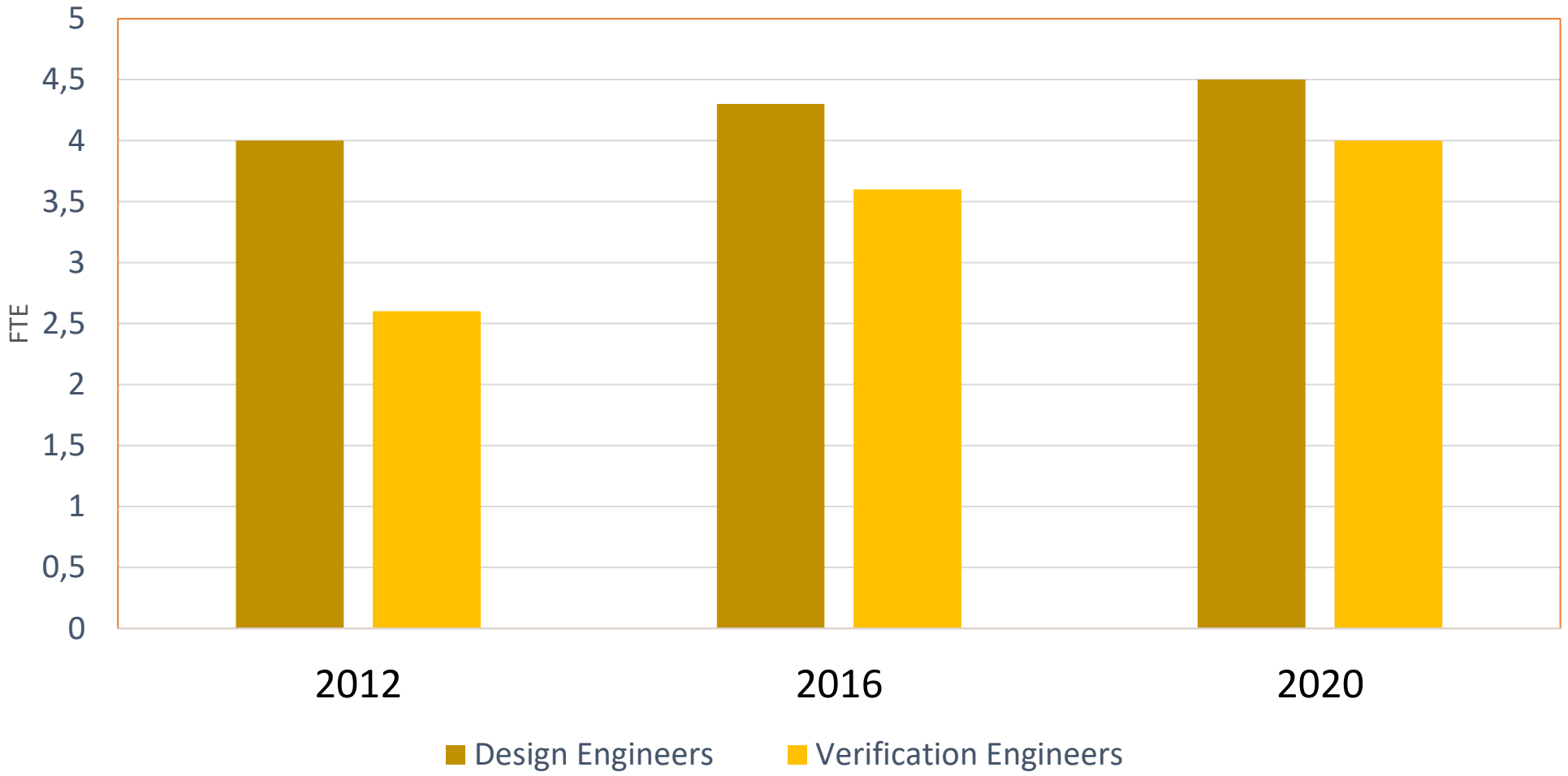
2021

HBK
Digital designer

ALTEN

VHDL verification methodologies, making good verification easy!

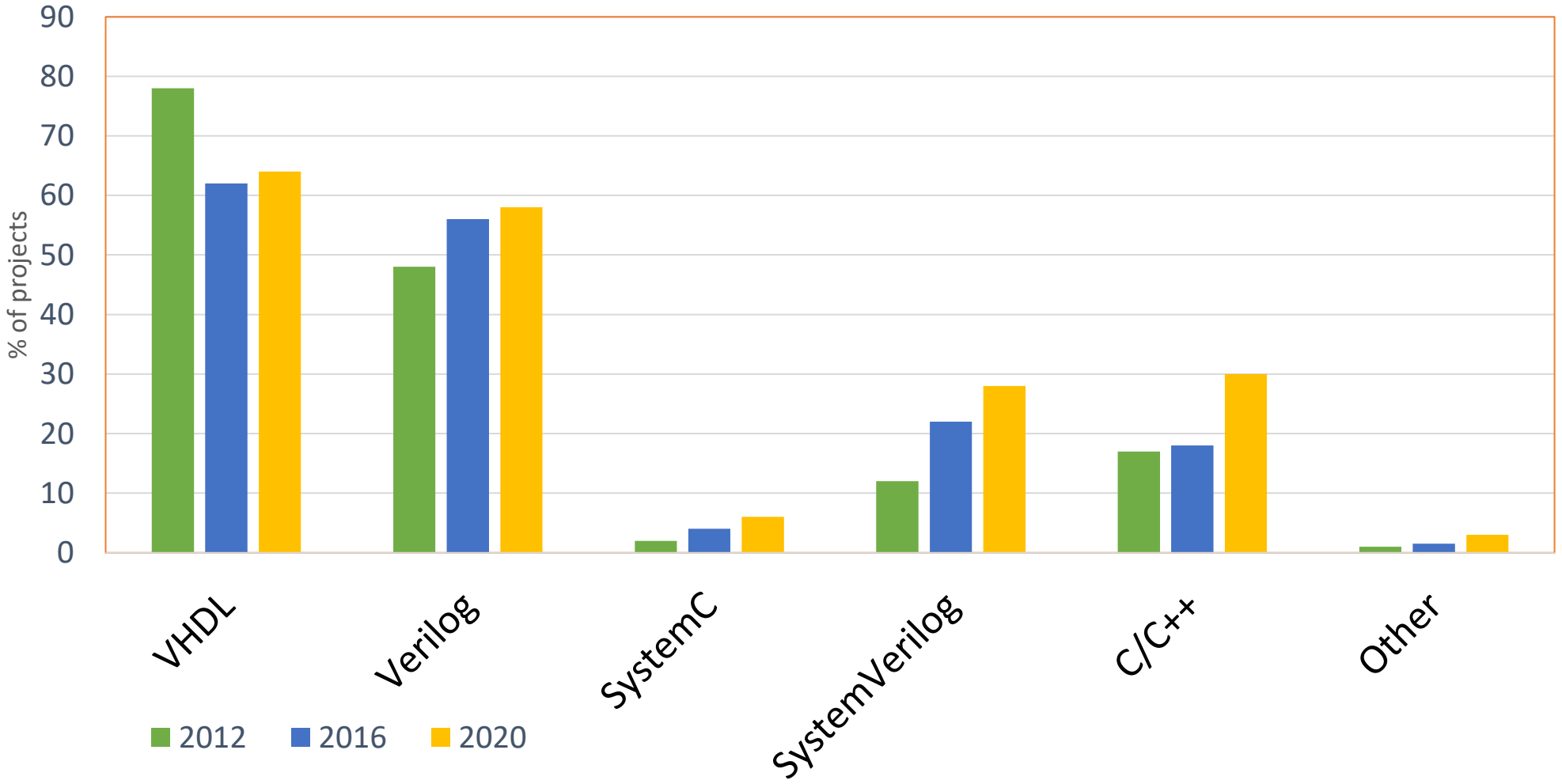
FPGA Project resource trend



Source: The 2020 Wilson Research Group Functional Verification Study

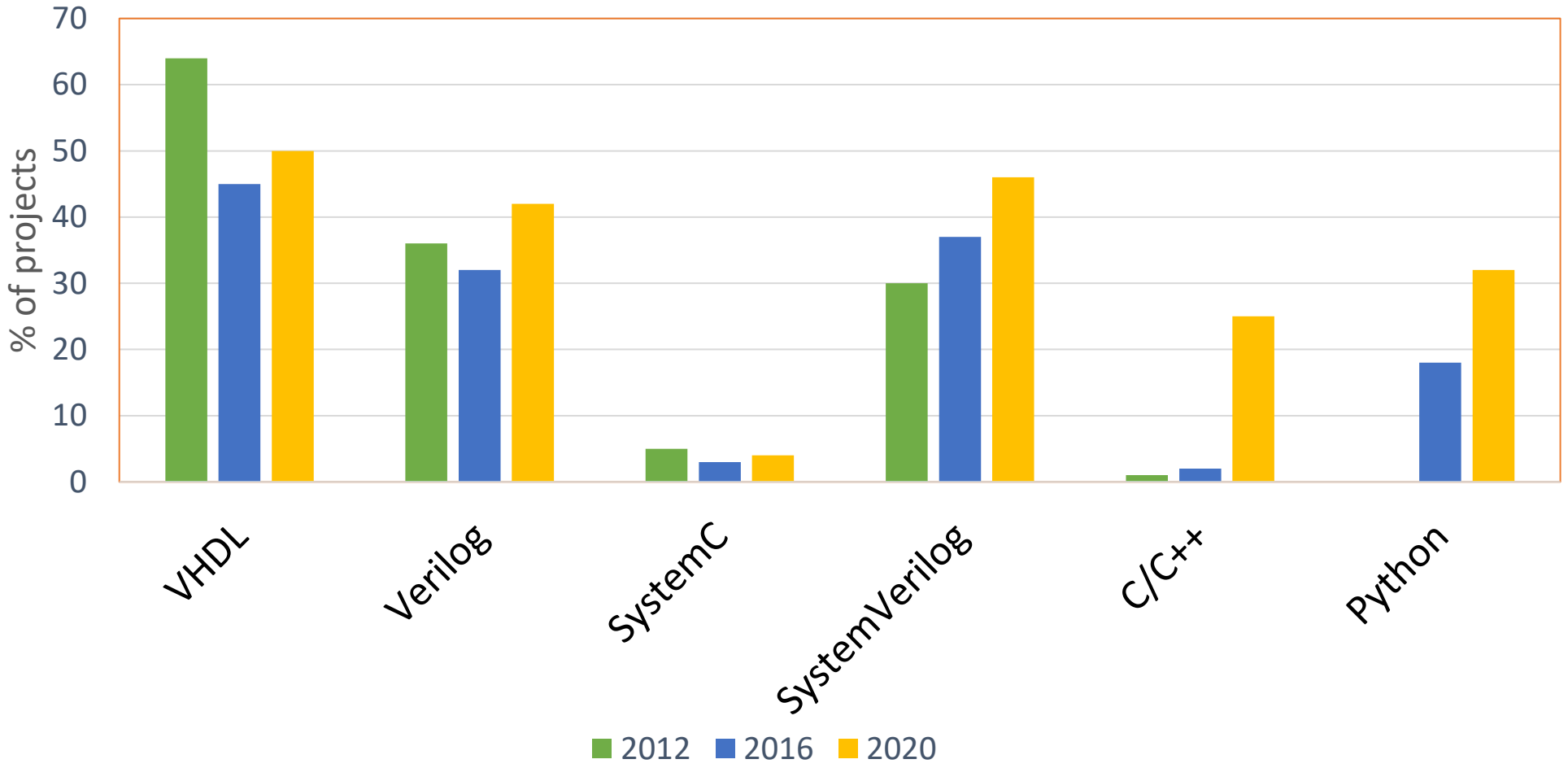
ALTEN

FPGA Design Languages



Source: The 2020 Wilson Research Group Functional Verification Study

FPGA Verification languages

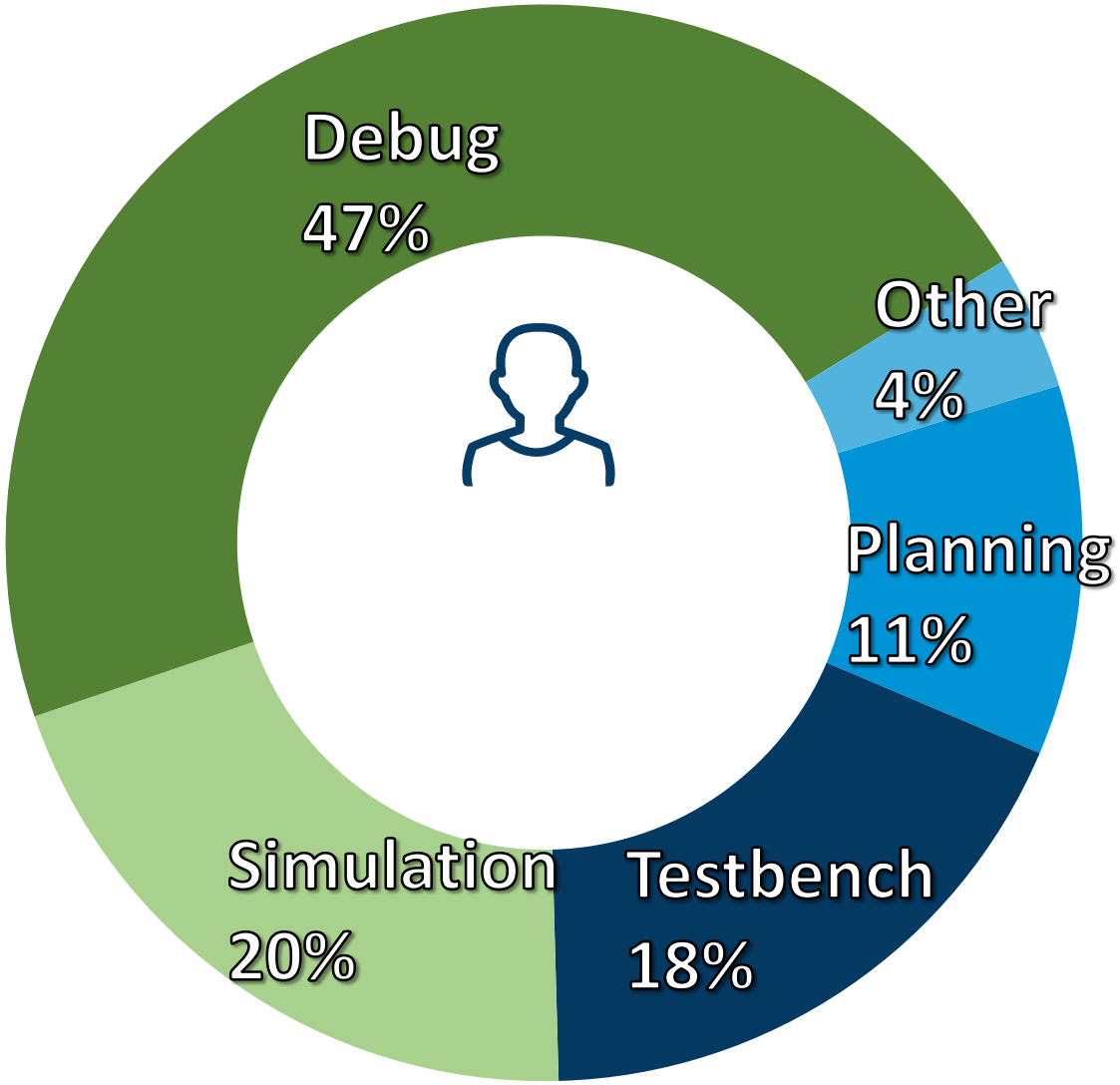


Source: The 2020 Wilson Research Group Functional Verification Study

ALTEN

17 MEI 2022 D&E event 2022
EVOLUON EINDHOVEN

Verification tasks

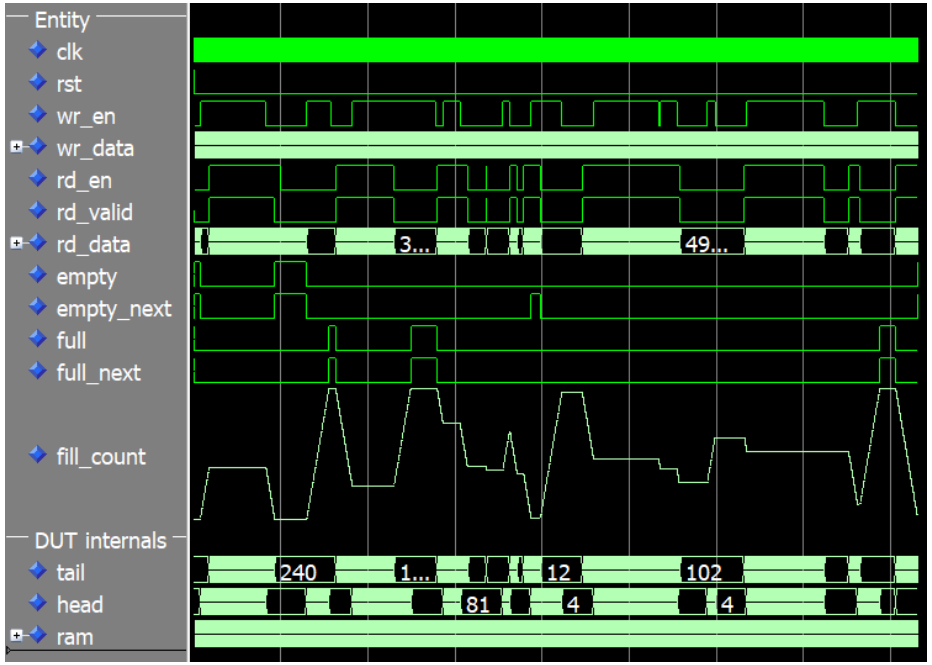


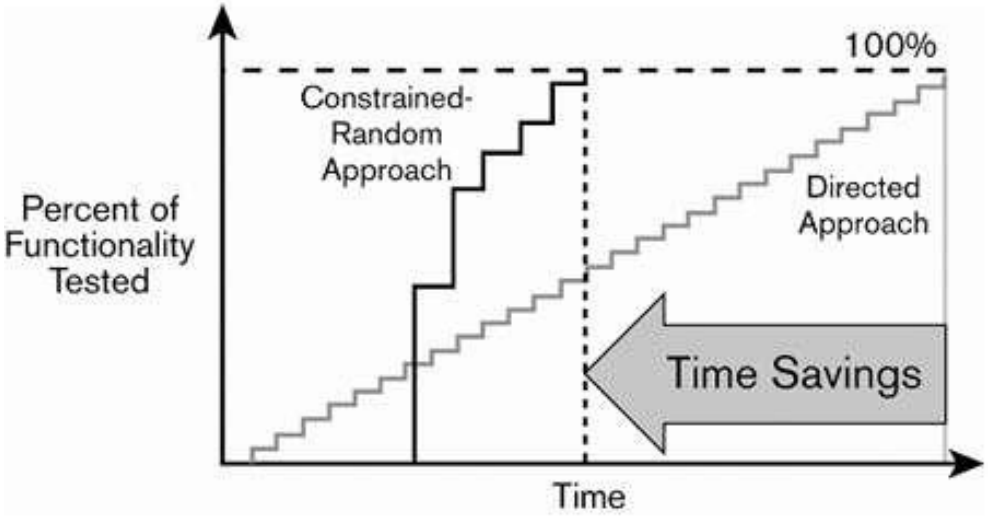
Source: The 2020 Wilson Research Group Functional Verification Study

ALTEN

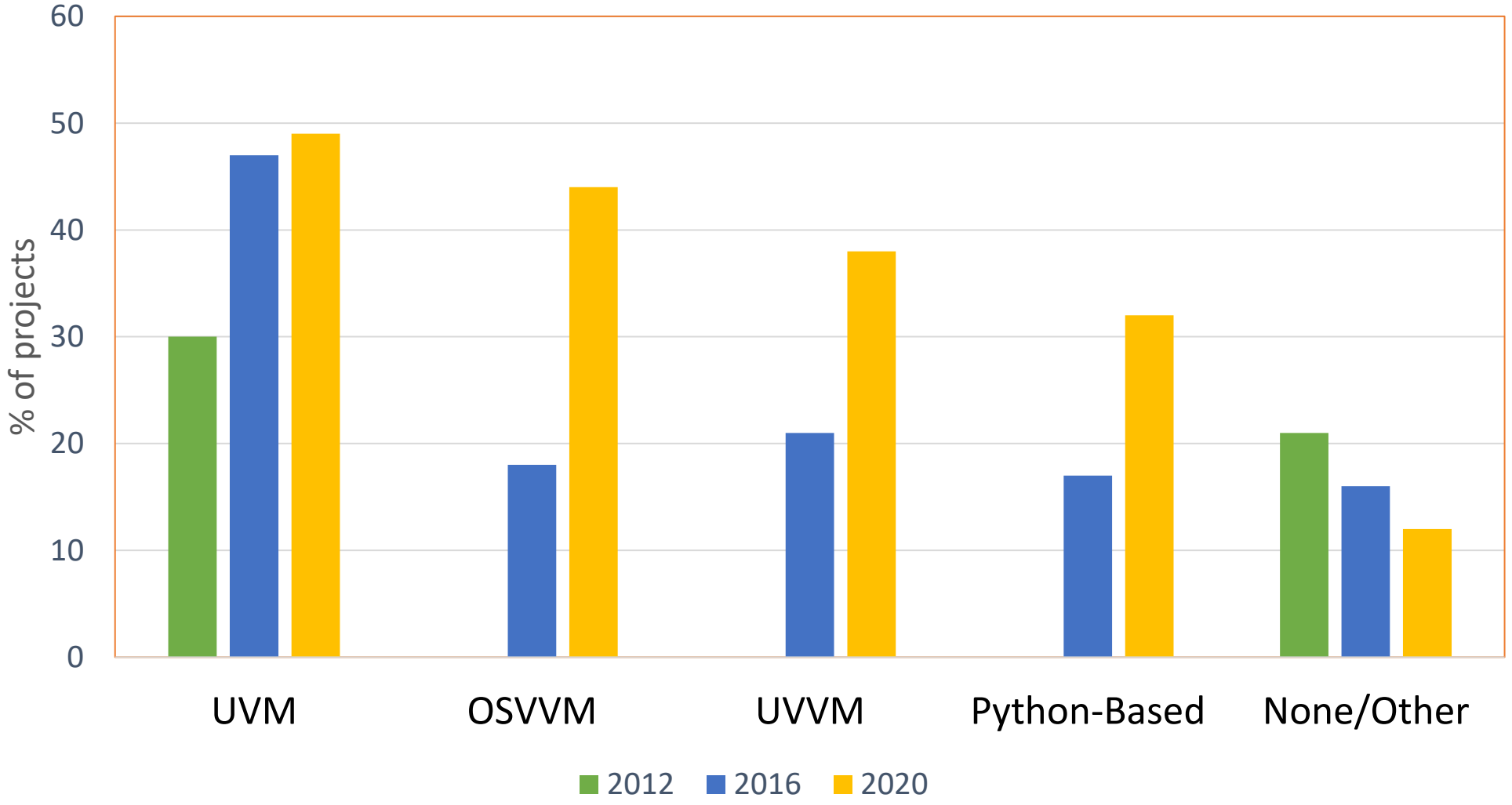
17 MEI 2022
EVOLUON EINDHOVEN
D&E event 2022

Constrained random verification





Verification Methadologies



Source: The 2020 Wilson Research Group Functional Verification Study

VHDL Verification Methodologies

UVM

- IEEE standard
- SystemVerilog based
- Complex to implement
- Cannot be integrated into existing testbenches

OSVVM/UVVM

- VHDL-2008 packages
- Easy to implement
- Easy to integrate into existing testbenches

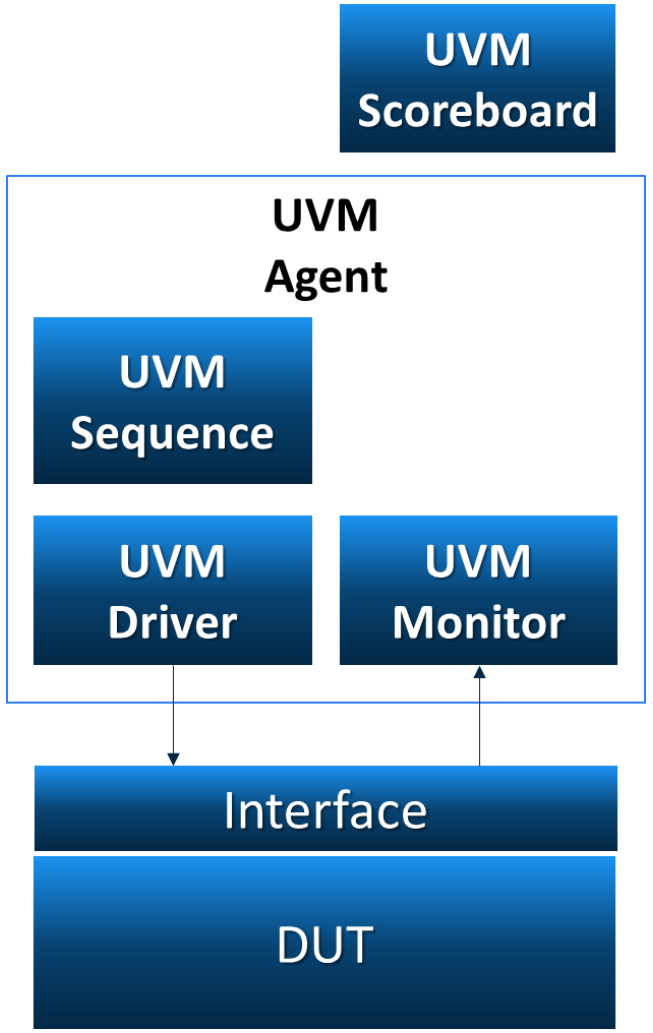
Cocotb

- Python based
- Easy to implement
- Cannot be integrated into existing testbenches
- Python packages can be used to create advanced self-checking testbenches

ALTEN

UVM

```
class driver extends uvm_driver #(Item);
  `uvm_component_utils(driver)
  function new(string name = "driver", uvm_component parent=null);
    super.new(name, parent);
  endfunction

  virtual des_if vif;

  virtual function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    if (!uvm_config_db#(virtual des_if)::get(this, "", "des_vif", vif))
      `uvm_fatal("DRV", "Could not get vif")
  endfunction

  virtual task run_phase(uvm_phase phase);
    super.run_phase(phase);
    forever begin
      Item m_item;
      `uvm_info("DRV", $sformatf("Wait for item from sequencer"), UVM_HIGH)
      seq_item_port.get_next_item(m_item);
      drive_item(m_item);
      seq_item_port.item_done();
    end
  endtask

  virtual task drive_item(Item m_item);
    @(vif.cb);
      vif.cb.in <= m_item.in;
  endtask
endclass
```



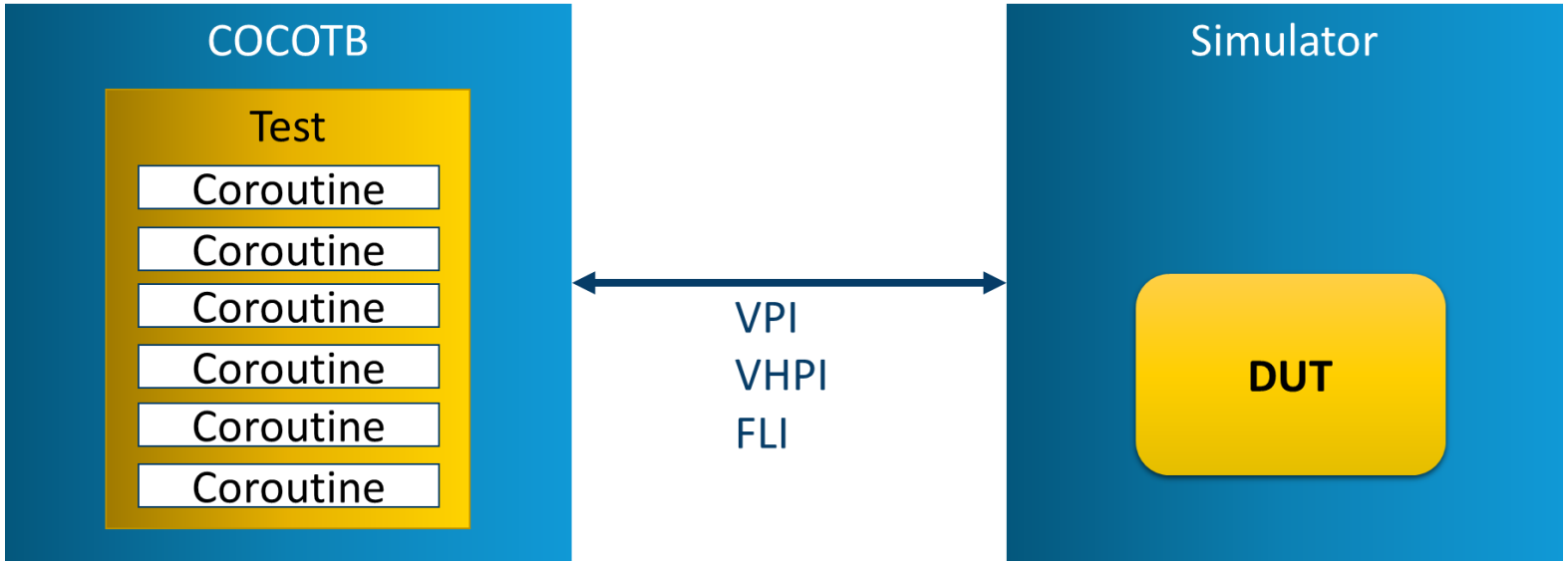
ALTEN

OSVVM

```
library ieee;
  use ieee.std_logic_1164.all;
  use ieee.numeric_std.all;

use std.env.all;

library OSVVM;
  use OSVVM.RandomPkg.all;
  use OSVVM.CoveragePkg.all;
```

ALTEN

COCOTB

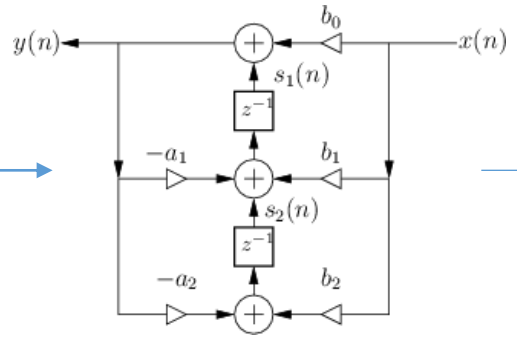



```python
@cocotb.test()
async def dff_simple_test(dut):
    """Test that d propagates to q"""

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

    await FallingEdge(dut.clk)  # Synchronize with the clock
    for i in range(10):
        val = random.randint(0, 1)
        dut.d.value = val  # Assign the random value val to the input port d
        await FallingEdge(dut.clk)
        assert dut.q.value == val, f"output q was incorrect on the {i}th cycle"
```

ALTEN

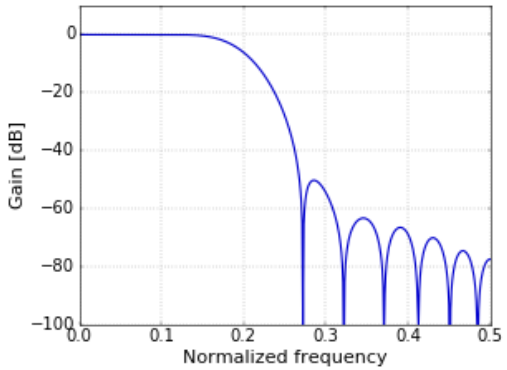
COCOTB

model



011
010
101
001

DUT



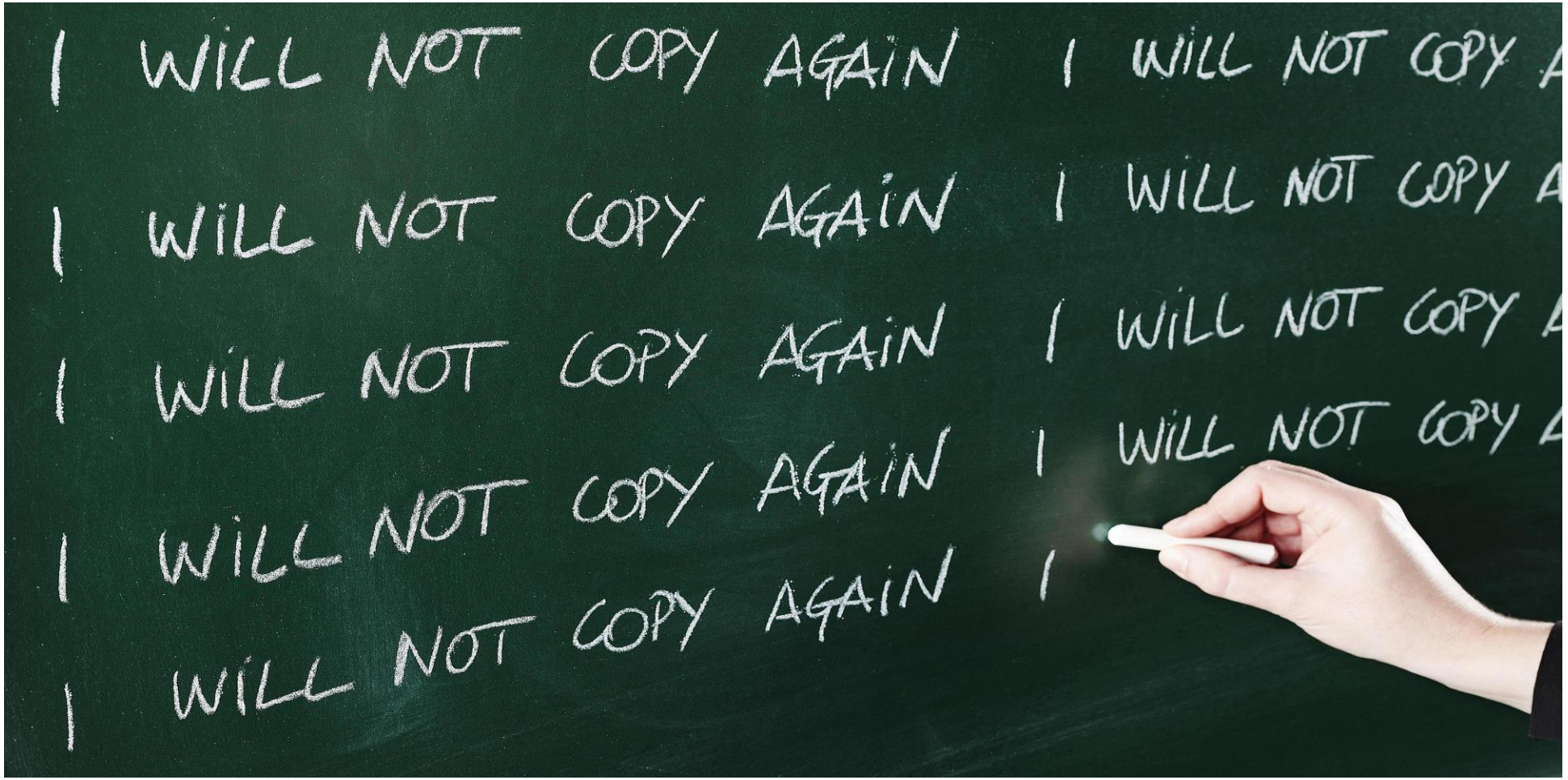
ALTEN

17 MEI 2022
EVOLUON EINDHOVEN
D&E event 2022

Quality

“If a jobs worth doing it’s worth doing well”

Productivitiy



Investment

ALTEN

17 MEI 2022
EVOLUON EINDHOVEN
D&E event 2022

Standarization

ALTEN

17 MEI 2022
EVOLUON EINDHOVEN
D&E event 2022

Conclusion

VHDL Verification Methodologies

making good quality verification easy

- Quality
- Productivity
- Standarization

WW: www.alten.nl
email: robert.wijma@alten.nl