2025 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

SAN JOSE, CA, USA
FEBRUARY 24-27, 2025

Accelerating Device Sign-off through a Unified Environment for DV, SV, and ATE with PSS

M. Suckert[a], S. Khaikin[c], A. Vazhayil[q],
N. Devendra[q], A. Samudra[a], K. Hilliges[a], M. Rubin[c]

[a] Advantest, [c] Cadence, [q] Qualcomm

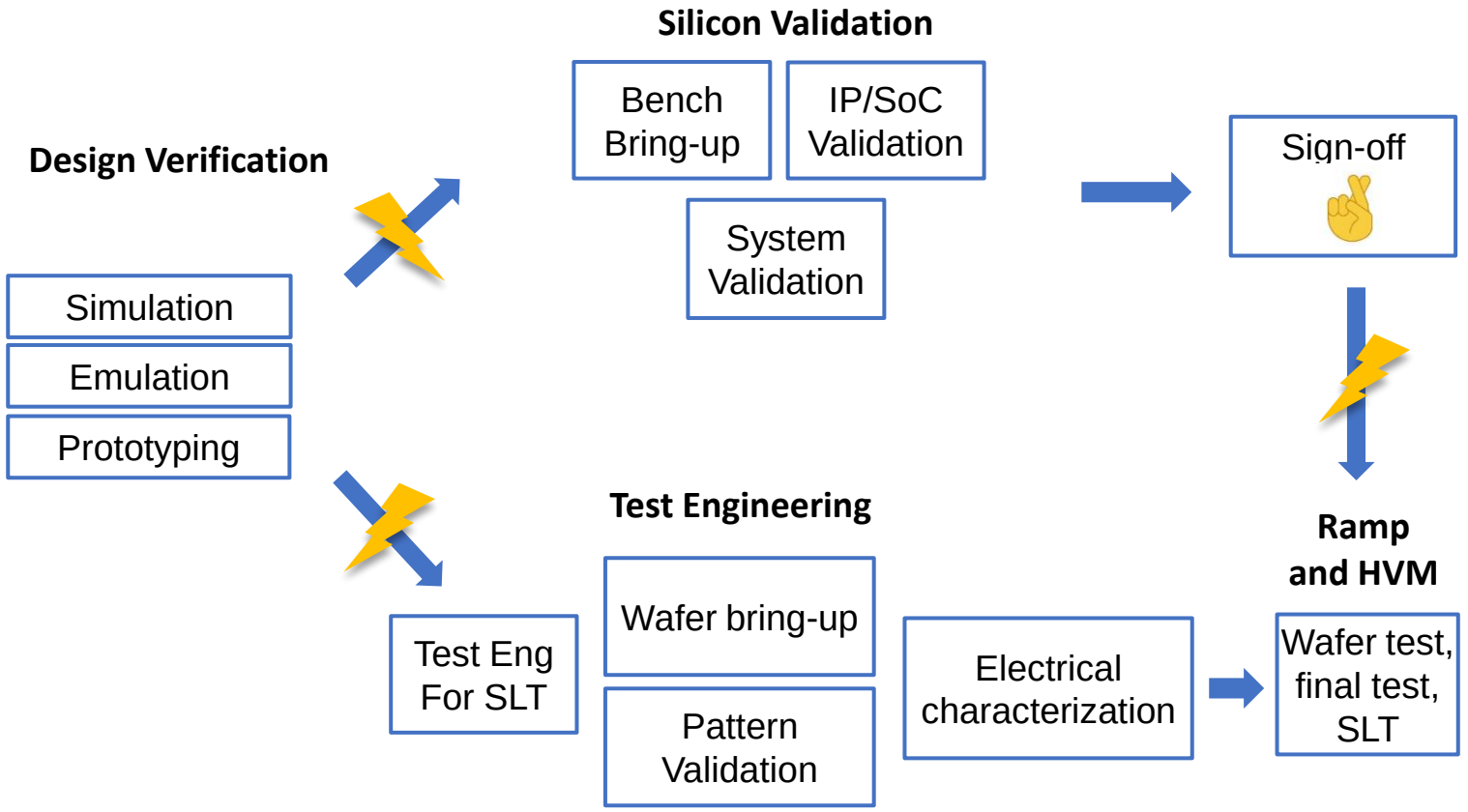
accellera SYSTEMS INITIATIVE

Break down of very long title

1. Accelerating Device Sign-off
2. through a Unified Environment
3. for DV, SV and ATE
4. with PSS

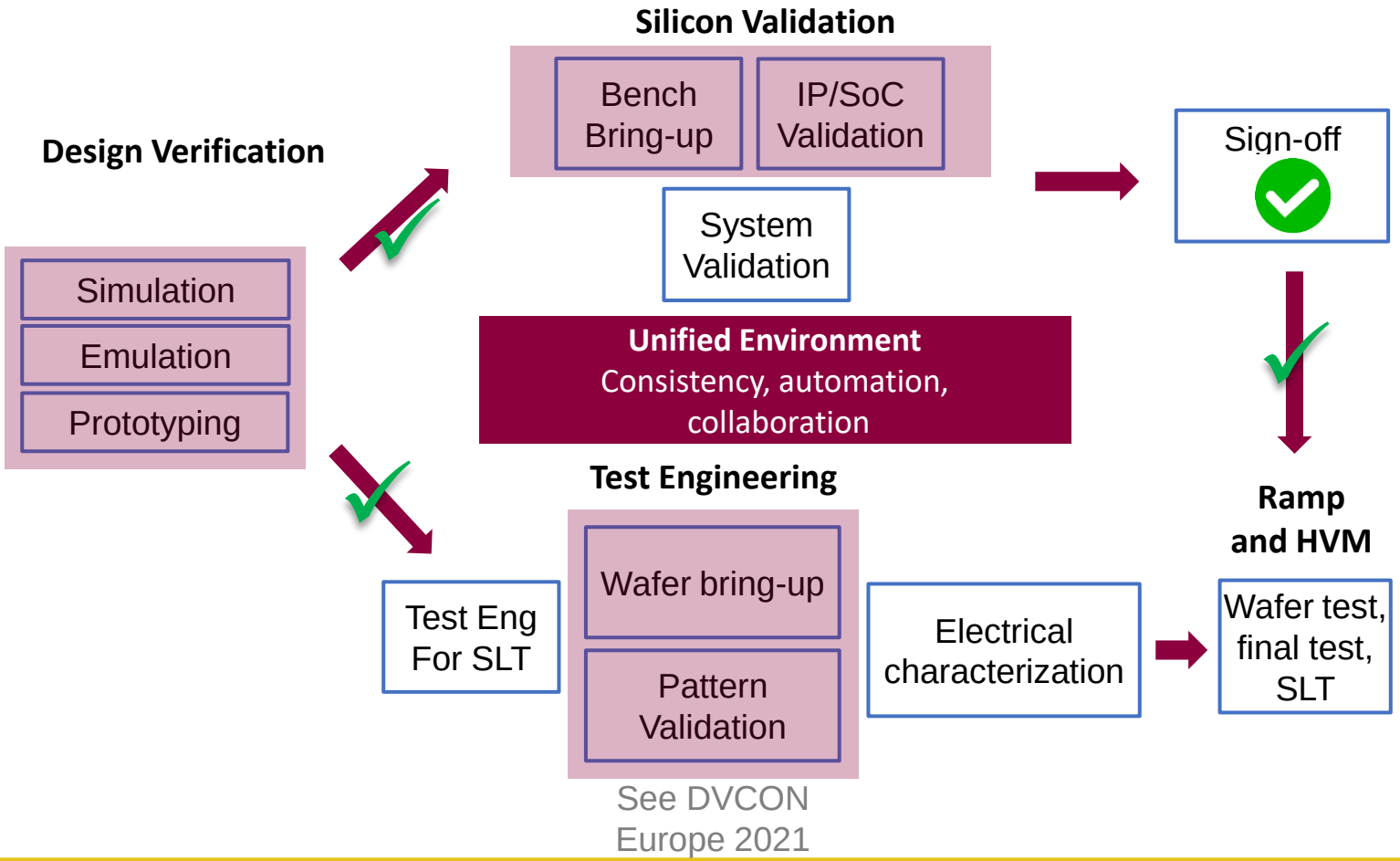
The struggle to collaborate for device sign-off

- Separate workflows
- Different tools and formats
- Complex SV benches
- Complex, shared ATEs
- Poor debug on ATE



Unified environment for device sign-off

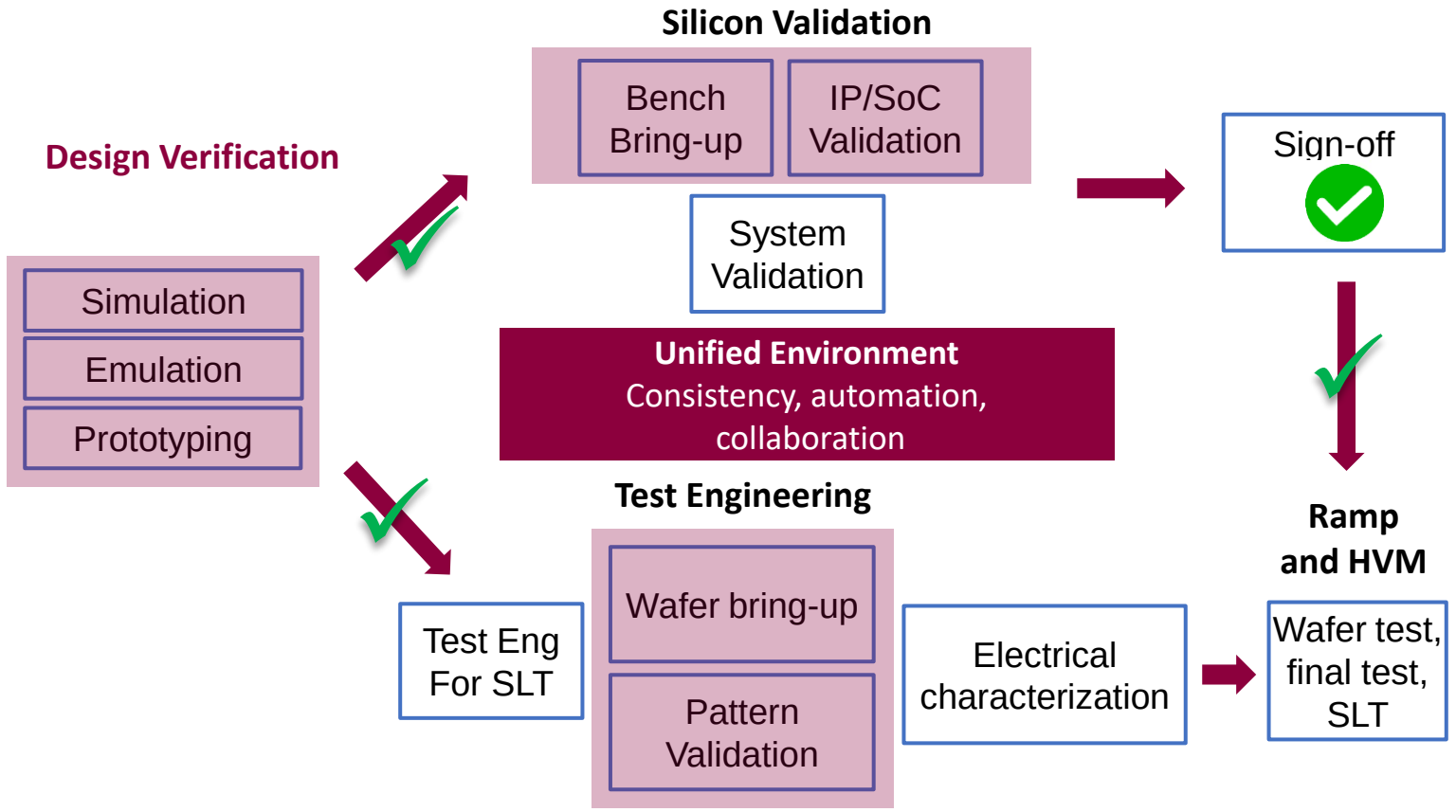
- Connected workflows
- Well defined interfaces and formats
- Bench instrument with standard interfaces and unfied software
- Native test execution
- Native software debug



See DVCON Europe 2021

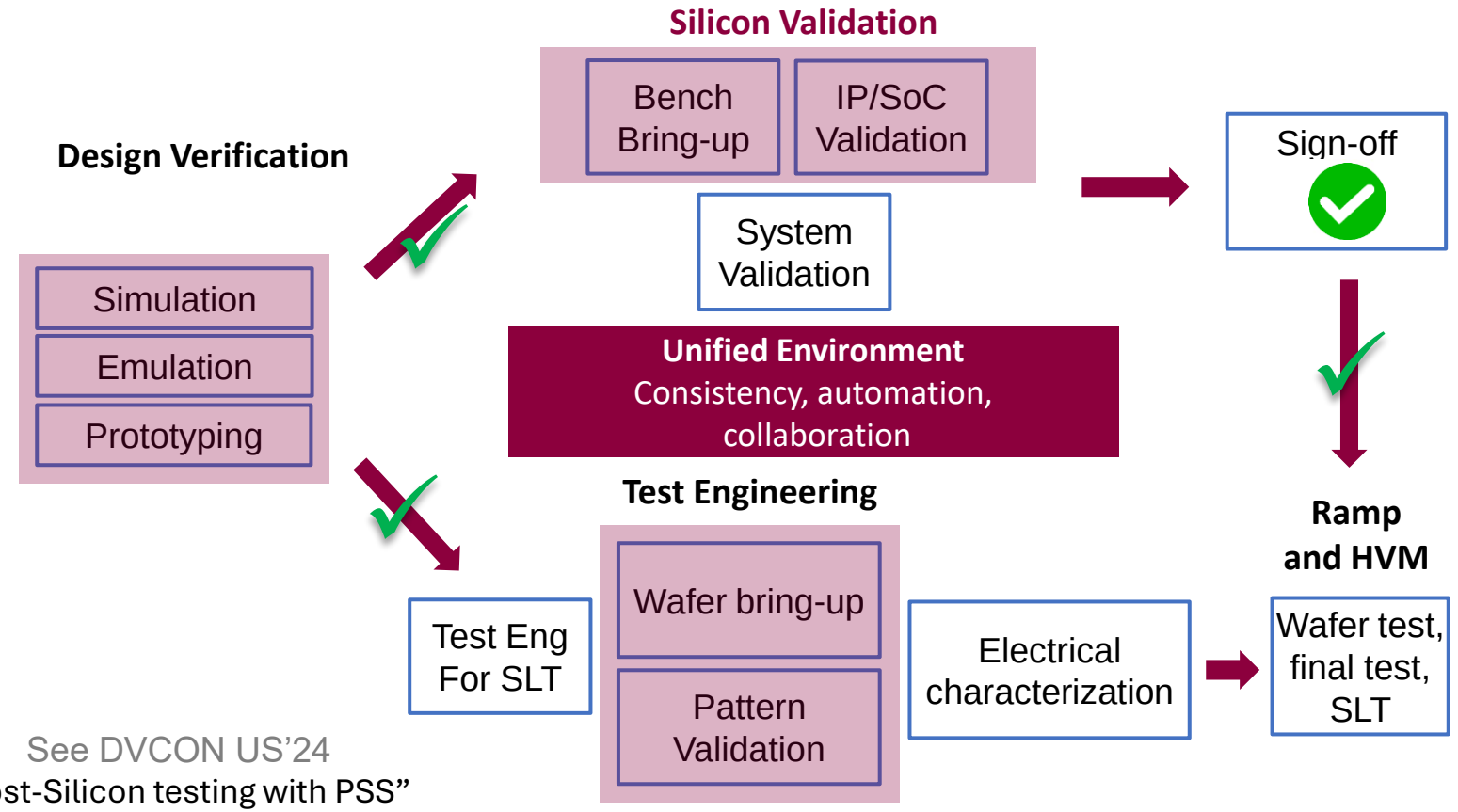
DV on silicon in unified environment

- Increase functional coverage
- Execute and explore faster on silicon
- Execute not simulatable content on silicon
- Debug tests on silicon without SV expert



Silicon Validation in unified environment

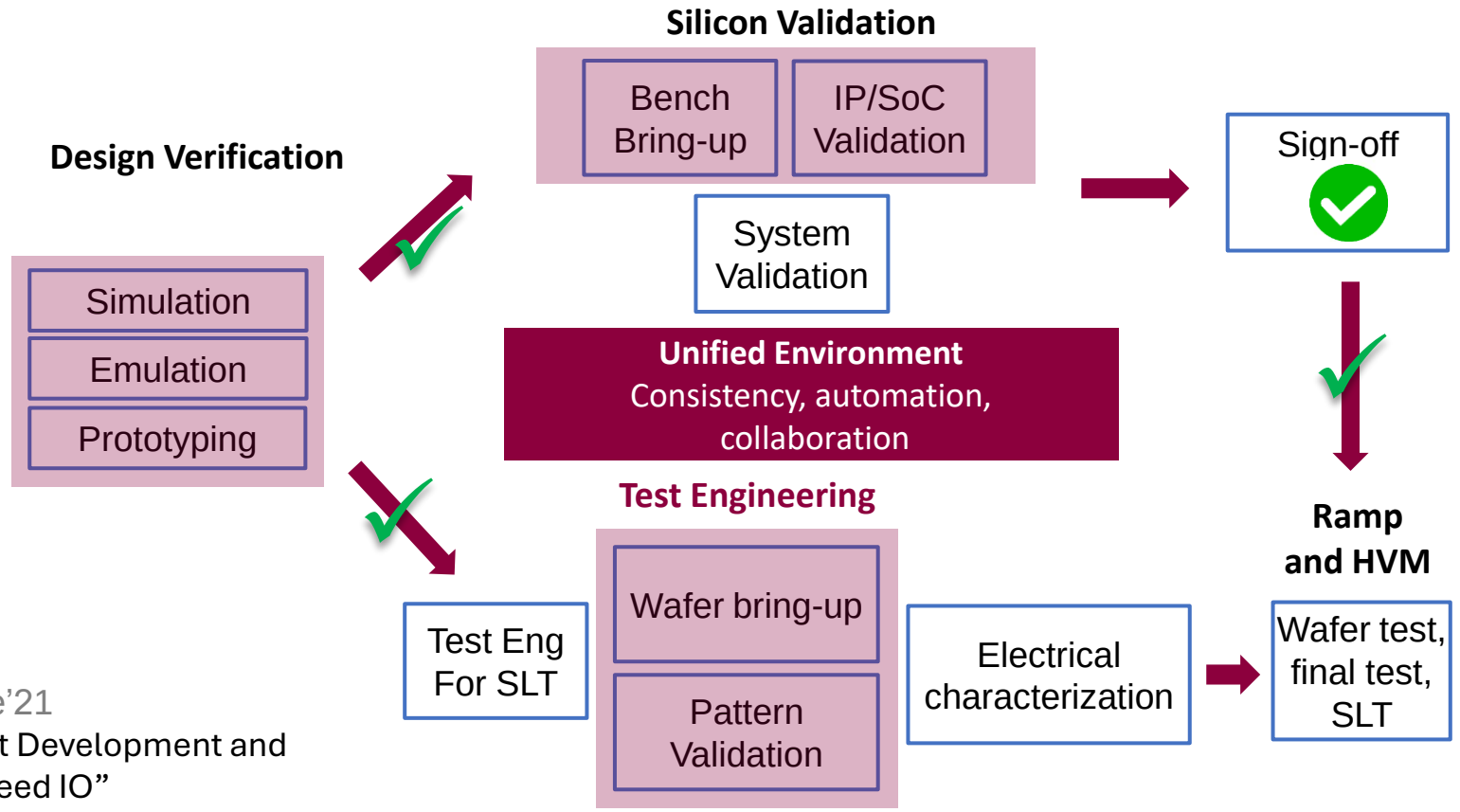
- Leverage pre-silicon coverage concepts
- Joint bringup and debug with DV
- PSS tests correct by construction and verified in simulation



See DVCON US'24
"Post-Silicon testing with PSS"
by Prabhat Gupta @AMD

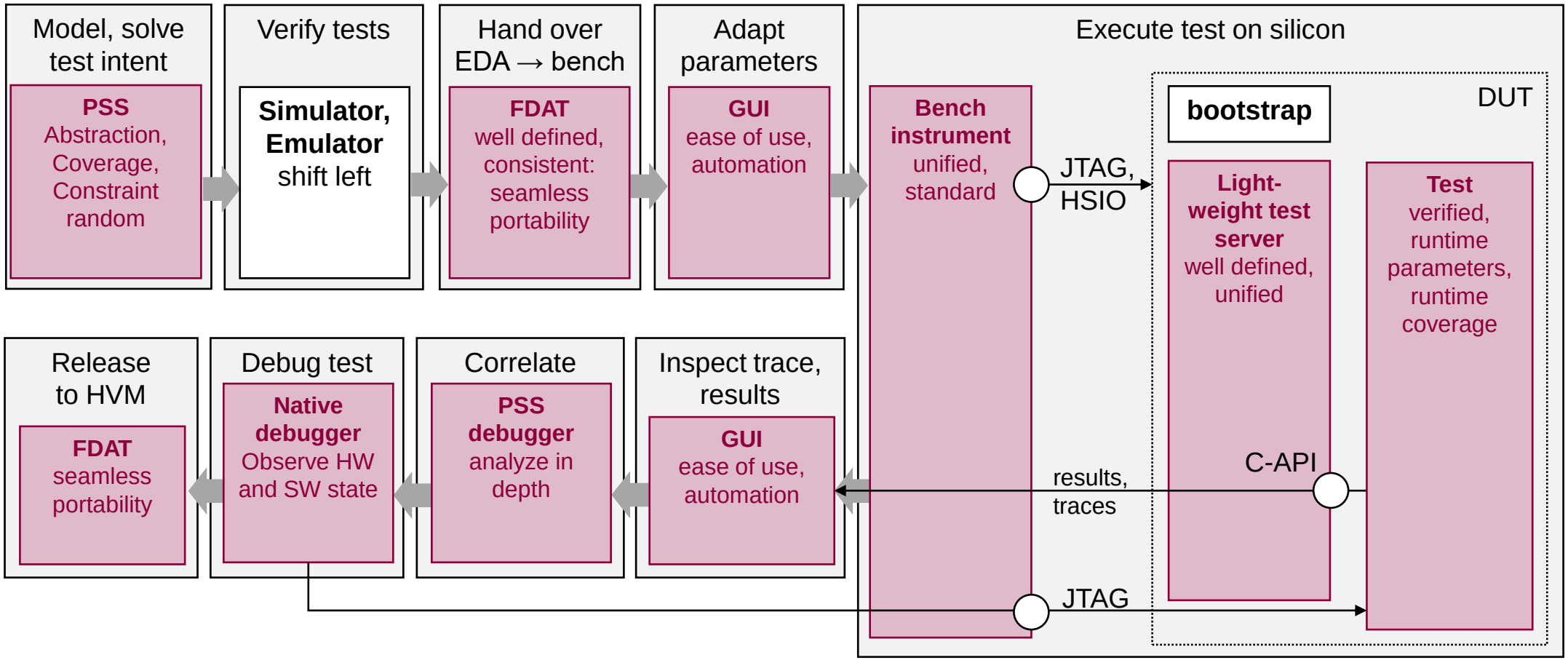
Test Engineering in unified environment

- tests signed-off with confidence
- well understood coverage from PSS
- concurrent test engineering



See DVCON Europe'21
"A Novel Approach to Functional Test Development and Execution using High-Speed IO"
by M. Schulze Westenhorst et al. @AMD, Cadence

Components of the unified environment



Native PSS 3.0 API for unified environment

Example: runtime parameter set

```
// (1) declare parameter set
struct user_param_set_algorithm_s : psv_base_param_set_s {
  //@tooltip("Algorithm type: Fast = 0, Slow = 1")          // doc comment
  rand uint8_t algorithm_p;                                 // name and type
  constraint algorithm_p_min_max { algorithm_p in [0..1]; } // value range

  //@tooltip("Seed")
  rand uint16_t seed_p;
  constraint seed_p_min_max { seed_p in [0..UINT16_MAX]; }
}

// declare parameterized component
component user_algorithm_c {
  // (2) define parameter-set component
  psv_param_set_c<user_param_set_algorithm_s> user_param_set_c;

 action initialize_a {
    activity {
      do psv_param_set_c<user_param_set_algorithm_s>::psv_initialize_param_set_a with {
        comp == this.comp.user_param_set_c;
        // (3) initialize parameter with default value
        param_set.algorithm_p == 0;
      };
    }
  }

  action run_test_pointer_a {
    exec body {
      // (4) retrieve pointer to parameter set and forward to target function
      comp.runAlgorithm( comp.user_param_set_c.get_pointer() );
    }
  }
}
```

```
// instantiate parameterized components
user_algorithm_c a1;
user_algorithm_c a2;

action main_a {
  activity {
    // initialize parameter sets
    do user_algorithm_c::initialize_a with { comp == this.comp.a1 };
    do user_algorithm_c::initialize_a with { comp == this.comp.a2 };

    // generate target code for parameter sets
    do psv_vip_parameters_c::prepare_parameters_a;

    // run test
    do user_algorithm_c::run_test_pointer_a with { comp == this.comp.a1; };
    do user_algorithm_c::run_test_pointer_a with { comp == this.comp.a2; };
  }
}
```

- abstract, portable test model
- correct by construction
- runtime coverage
- tracing
- debug and correlation

FDAT – well defined container for tests

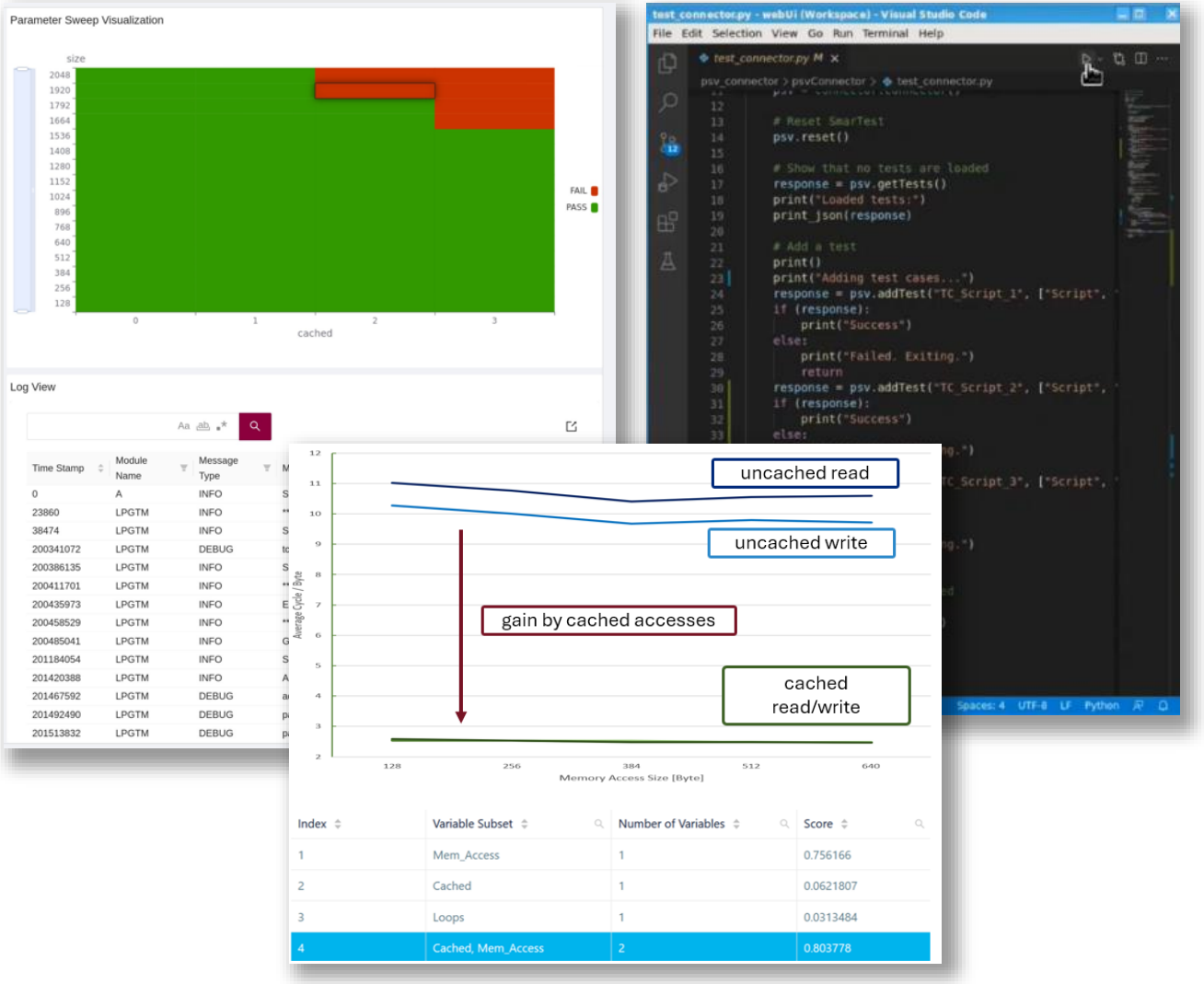
- FDAT = functional test data container
- complete description of functional test
- native binary
- zip format
- well defined directory and file structure
- well defined interface files for test parameters and test results
- open for user extensions

Unified bench instrument

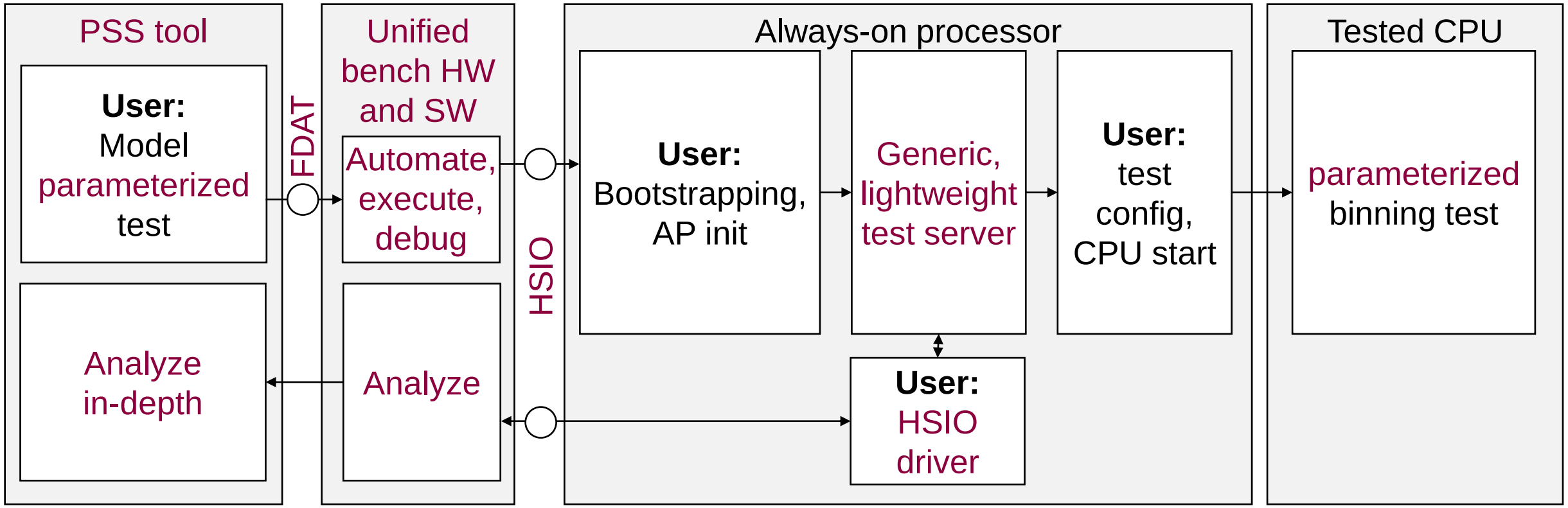
- air cooled
- standard power supply
- bench footprint
- HSIO interfaces: PCIe, USB
- debug interfaces: JTAG, SPI, I2C, UART
- industry standard connectors
- reliability of ATE

Bench software

- ease of use
- automation
- data analytics
- FDAT as input format
- native software debugging
- well-defined HSIO protocol



Use case: Acceleration of device sign-off



Use case: Acceleration of device sign-off

- flow setup using templates and example code
- seamless integration of PSS API
- seamless handover of tests with FDAT
- debug test with native software debugger
- explore and sign off test with runtime parameters
- re-use in next project

Acceleration of device sign-off

- PSS productivity gain by factor 5
- reduced churn time in DV-SV workflow
  - Native binary execution without patter conversion saves hours per iteration
  - debug without expert support saves hours to days of meeting organization per iteration
- higher coverage on silicon for test with poor accuracy in simulation
- collecting coverage on silicon faster by 10^5 to 10^6 than on simulation

Questions

- Thanks for joining!

- maximilian.suckert@advantest.com
- skhaikin@cadence.com
- aashokv@qti.qualcomm.com