EmLogic

UVVM
– An introduction to the world's fastest growing FPGA verification methodology

FDF-2024, CERN

(by Espen Tallaksen, CEO EmLogic)

EmLogic

- Independent Design Centre for Embedded Systems and FPGA
- Established 1st of January 2021. **Extreme ramp up**
  - January 2021: 1 person
  - June 2023: → 43 persons (SW:19, HW:4, FPGA:18, DSP:1+)
- Continues the legacy from bitvis
  - All previous Bitvis technical managers are now in EmLogic
- Verification IP and Methodology provider **UVVM**
- Course provider within FPGA Design and Verification
  - Accelerating FPGA Design (Architecture, Clocking, Timing, Coding, Quality, Design for Reuse, …)
  - Advanced VHDL Verification – Made simple (Modern efficient verification using UVVM)
- A potential partner for ESA projects for European companies
  - Increased opportunities due to Norway's low geo return

What is UVVM?

UVVM = Universal VHDL Verification Methodology

- VHDL Verification Library & Methodology
- Free and Open Source

- Very structured infrastructure and architecture
- Significantly improves Verification Efficiency
- Assures a far better Design Quality

DOULOS

- Recommended by Doulos for Testbench architecture
- ESA projects to extend the functionality
- IEEE Standards Association Open source project

esa

IEEE SA OPEN

- Runs on any VHDL-2008 compliant simulator

SIEMENS Mentor A Siemens Business

ALDEC THE DESIGN VERIFICATION COMPANY

GHDL

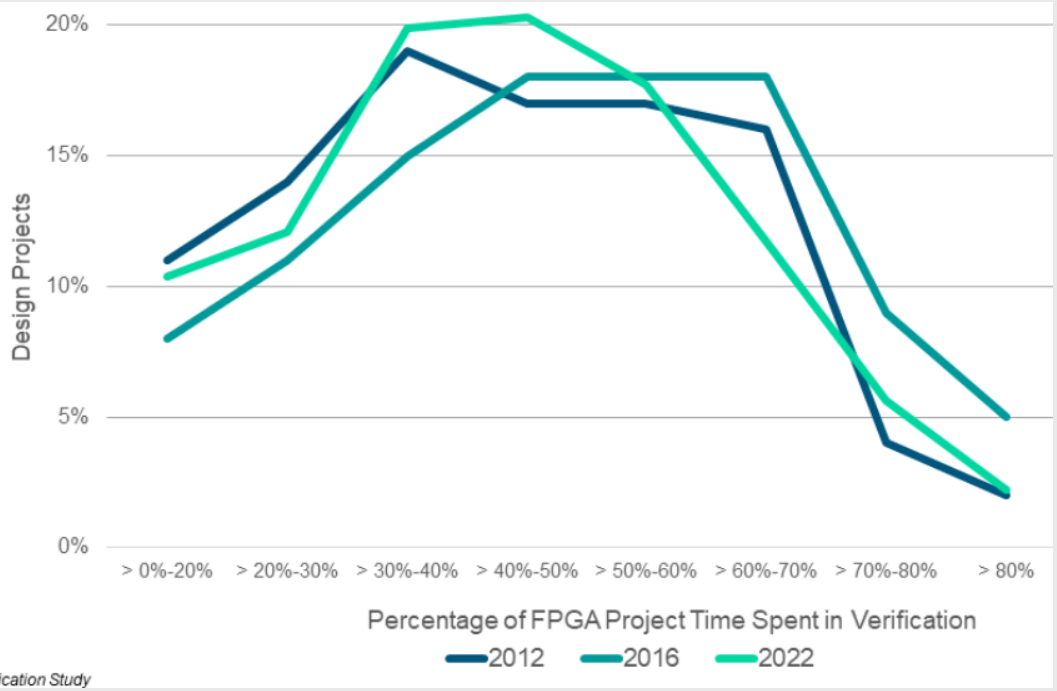
EmLogic

The 2022 Wilson Research Group Functional Verification Study (1)

Nearly half the project time is spent in verification

40%-50%

Median project time spent in verification



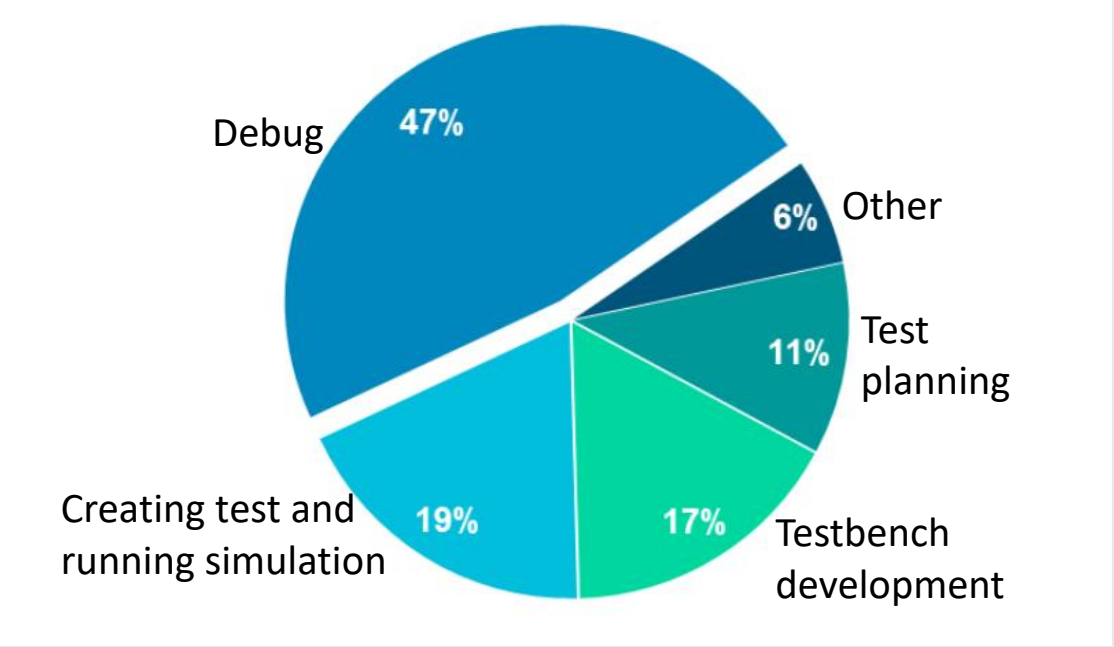
Source: Wilson Research Group and Siemens EDA, 2022 Functional Verification Study

Could verification time be reduced without reducing the quality?

EmLogic

The 2022 Wilson Research Group Functional Verification Study (2)

Half the verification time is spent on debugging



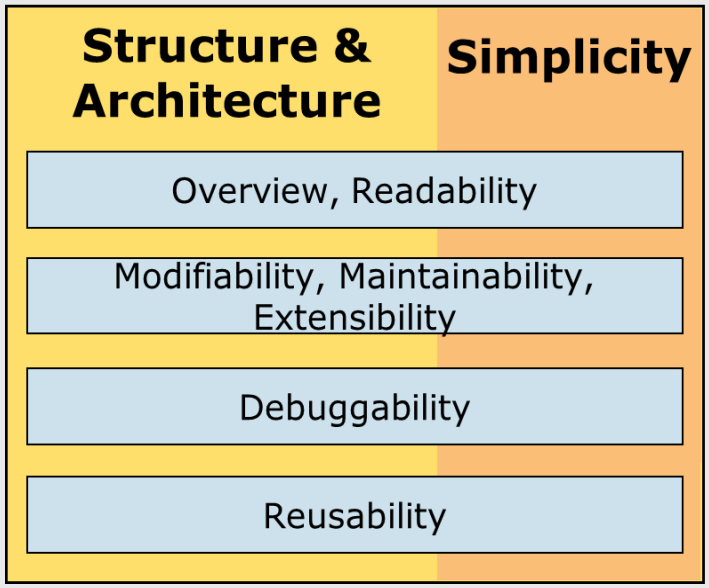
2022 WILSON RESEARCH GROUP, FUNCTIONAL VERIFICATION STUDY
FPGA FUNCTIONAL VERIFICATION TREND REPORT

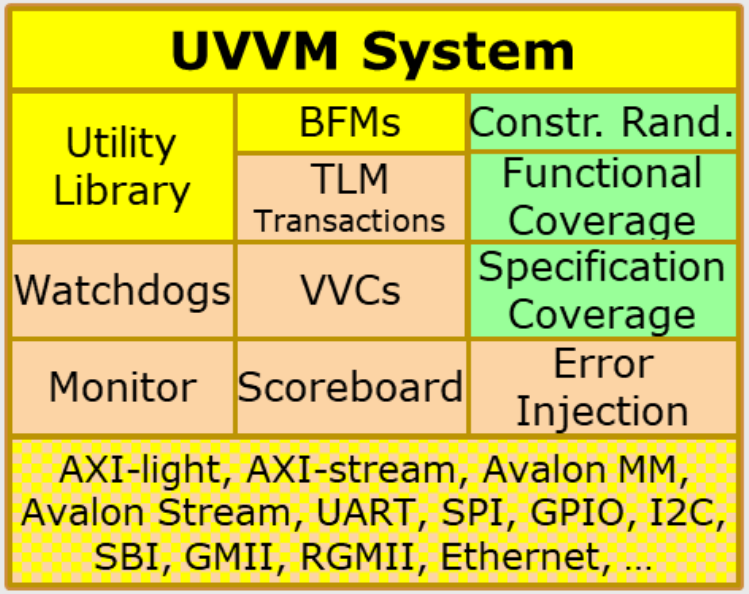
We can definitely be more efficient! - structured!

EmLogic

What enables Quality and Efficiency

- Huge improvement potential for more structured FPGA verification



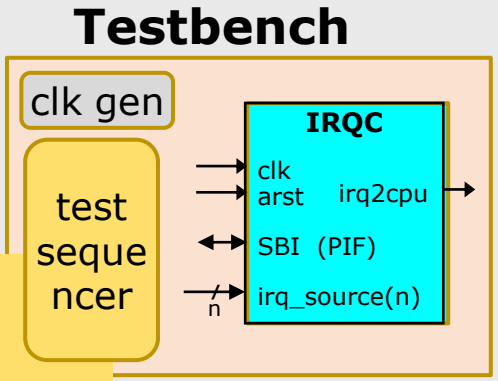


UVVM targets all of this

EmLogic

Example on test sequencer code and transcript/log

```
clock_generator(clk, GC_CLK_PERIOD);
```

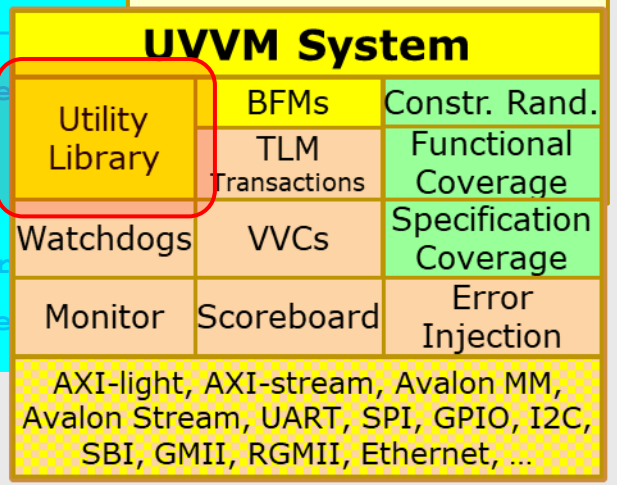



```
log(ID_LOG_HDR, "Check Interrupt trigger clear mechanism");
check_value(irq2cpu, '0', "irq2cpu default inactive");
check_stable(irq2cpu, now - v_reset_time, "Stable irq2cpu");
gen_pulse(irq_source, '1', C_CLK_PERIOD, "Set IRQ source for clock period");
await_value(irq2cpu, '1', 0 ns, 2* C_CLK_PERIOD, "Interrupt expected");
sbi_write(C_ADDR_ITR, x"AA", "ITR : Set interrupts");
```

```
2000.0 ns    Check Interrupt trigger clear mechanism
-------------------------------------------------------------------
 110.0 ns    check_value() => OK, for std_logic '0'. irq2cpu de
 727.5 ns    check_stable() => OK. Stable at 0. Stable irq2cpu
1060.0 ns    Pulsed to '1'. Set IRQ source for clock period
1117.5 ns    await_value(std_logic 1, 0 ns, 20 ns) => OK. Inter
2020.0 ns    SBI write(A:x"2", x"AA") completed. ITR : Set inte
```

All procedures with:

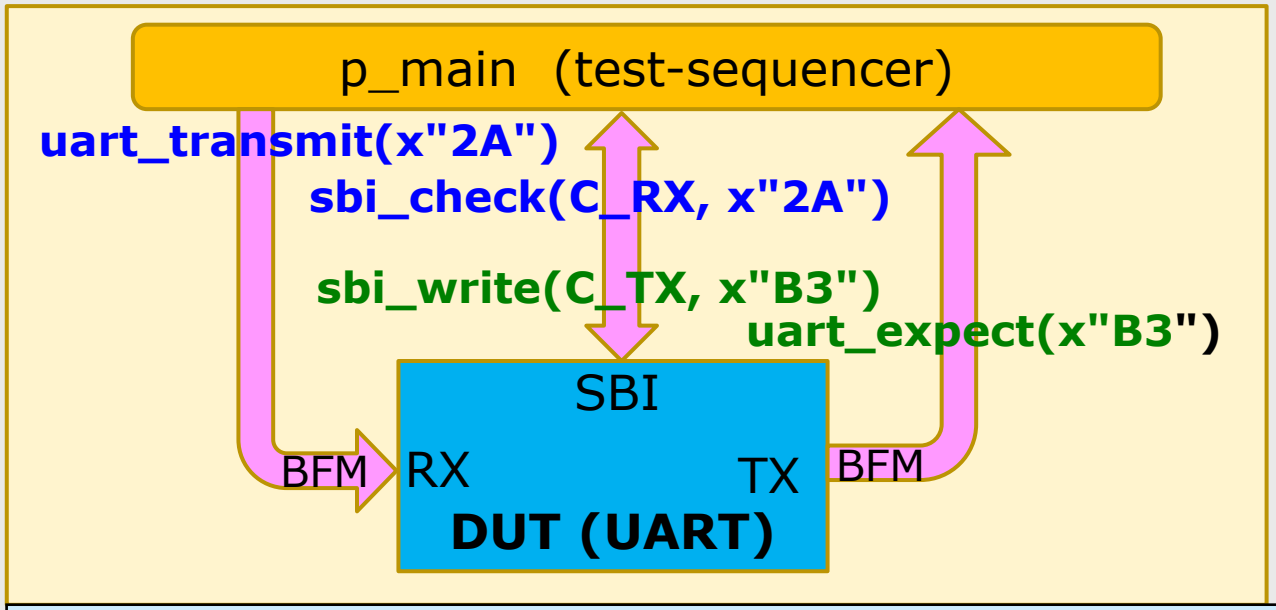
- Positive acknowledge If wanted



UVVM Utility Library for simple **and** advanced testbenches

- check_stable(), await_stable()
- clock_generator(), adjustable_clock_generator()
- random(), randomize()
- gen_pulse()
- block_flag(), unblock_flag(), await_unblock_flag()
- await_barrier()
- enable_log_msg(), disable_log_msg()
- to_string(), fill_string(), to_upper(), replace(), etc…
- normalize_and_check()
- set_log_file_name(), set_alert_file_name()
- wait_until_given_time_after_rising_edge()
- etc…

Simple data communication



May use Utility Library and provided BFMs

Free, Open source BFMs:

UART, AXI4-lite, SPI, I2C, Avalon MM, AXI4-stream, Avalon stream, GPIO, SBI, GMII, RGMII, ...

All well documented

```
TB:   172 ns. uart_tb     uart_transmit(x2A) on UART RX
TB:   192 ns. uart_tb     sbi_check(x1, ==> x2A) completed. From UART RX
```

```
TB:   192 ns. uart_tb     sbi_write(x2, ==> xB3) completed.
TB: ERROR:
TB:      192 ns. uart_tb
TB:              value was: 'xB2'.  expected 'xB3'.
TB:              (From uart_expect(xB3))
TB:=================================================
```

UVVM System

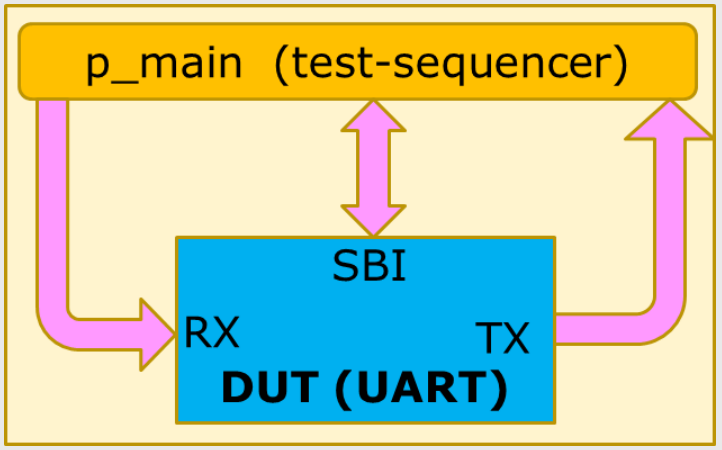
Utility Library	BFMs	Constr. Rand.
	TLM Transactions	Functional Coverage
Watchdogs	VVCs	Specification Coverage
Monitor	Scoreboard	Error Injection

AXI-light, AXI-stream, Avalon MM, Avalon Stream, UART, SPI, GPIO, I2C, SBI, GMII, RGMII, Ethernet, ...

BFM procedures are not sufficient

BFM: Defined here as a procedure only

- BFMs are great for simple testbenches
  - Dedicated procedures in a simple package
  - Just reference and call from a process
- BUT
  - A process can only do one thing at a time
    - Either execute that BFM
    - **Or** execute another BFM
    - **Or** do something else
- To do more than one thing:
  → Need an entity (or component)
  (VC = Verification Component)



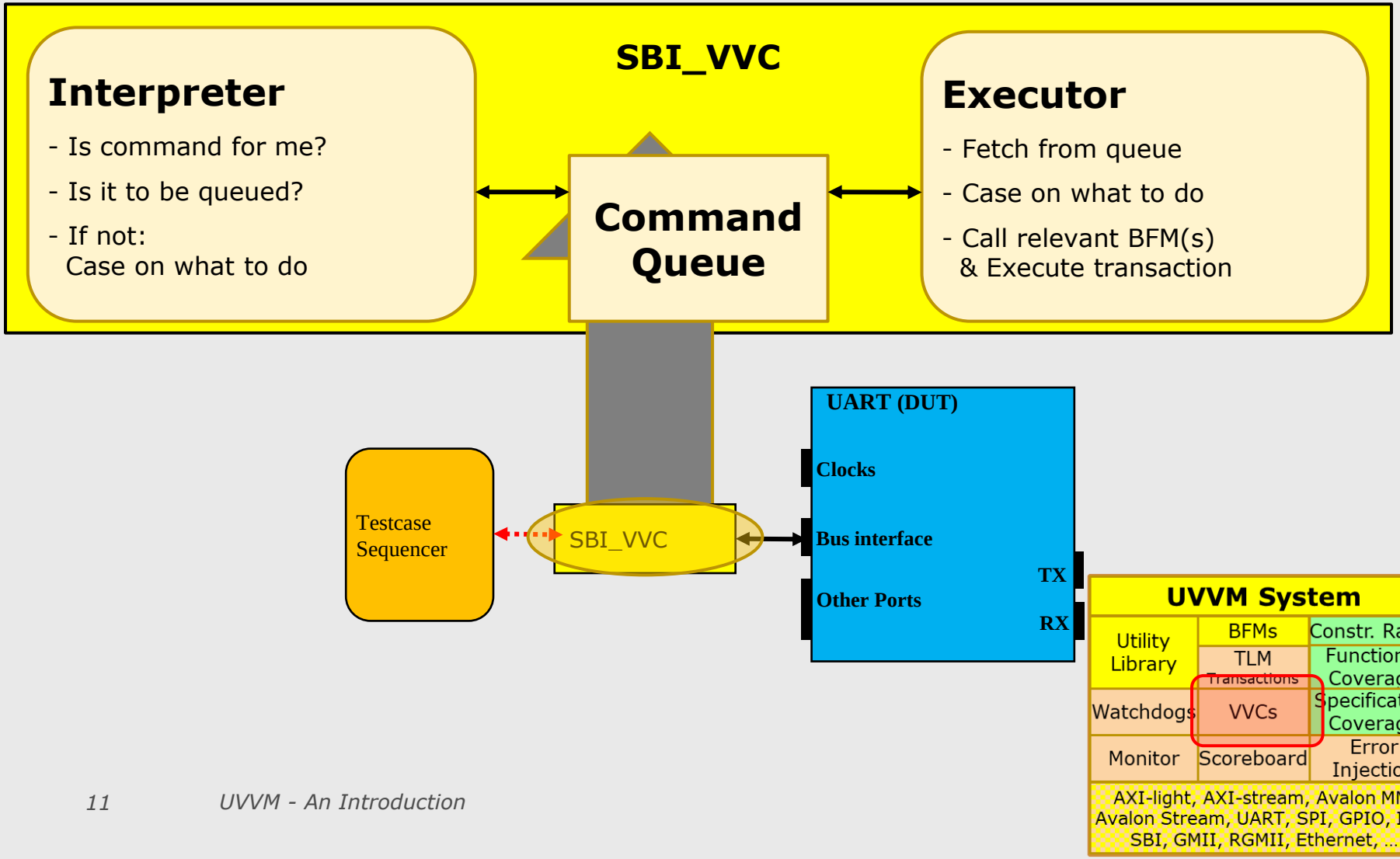
sbi_write(C_TX, x"B3")

uart_expect(x"B3")

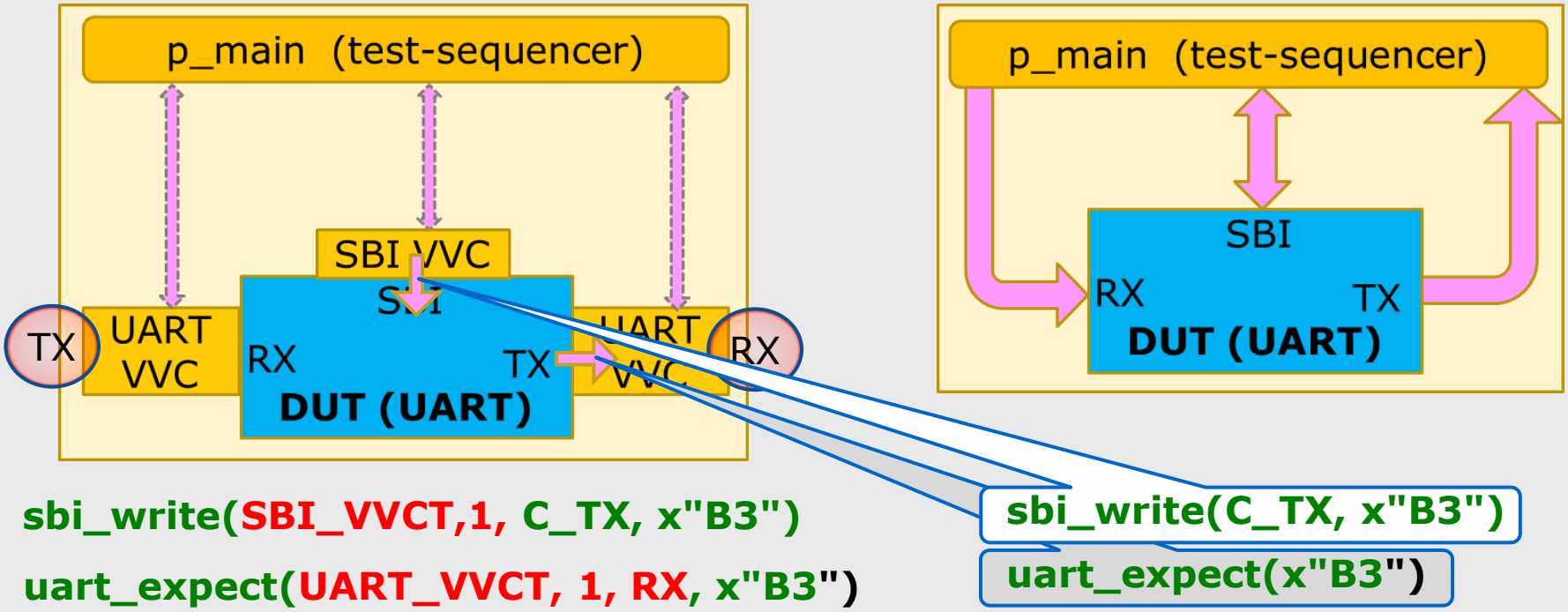
VVC: VHDL Verification Component (UVVM VC with extended functionality)

EmLogic

VVC: VHDL Verification Component



BFM to VVC: How?



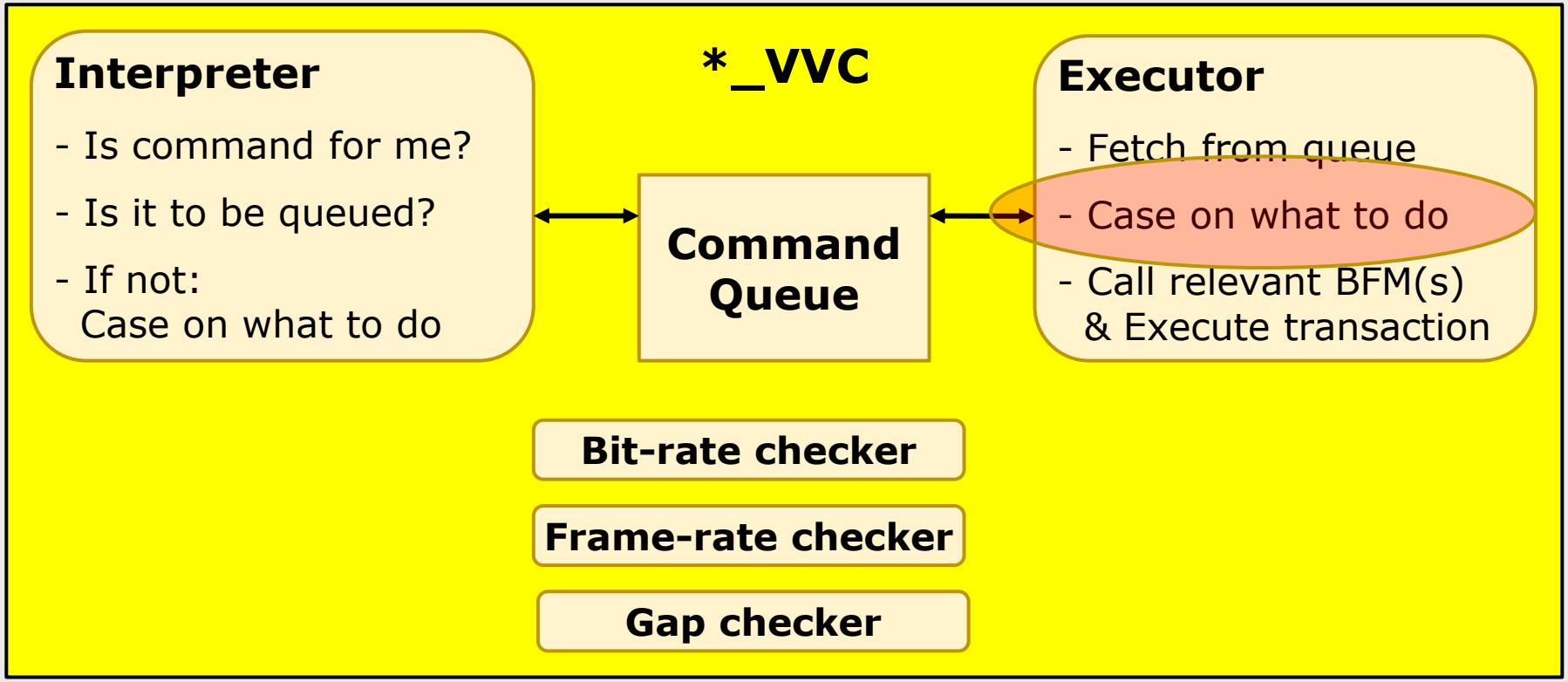
UVVM VVCs also include:
Delay-insertion, command queuing, completion detection, activity registration, multicast & broadcast, termination, set-up, data fetch, multi-channel support, interface checkers, scoreboards, transaction info, local sequencers, etc …

EmLogic

VVC: Easy to extend (1)

- Easy to add local sequencers

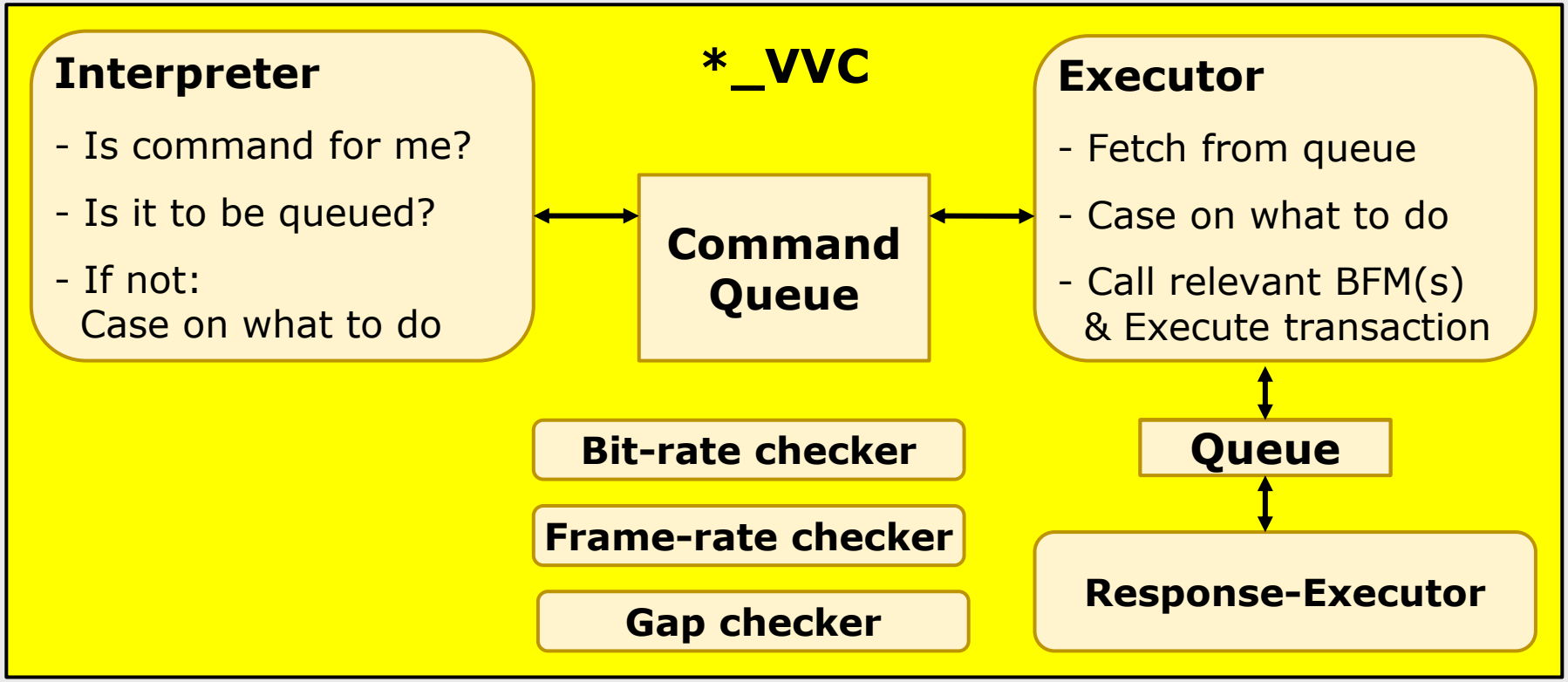
- Easy to add checkers/monitors/etc



EmLogic

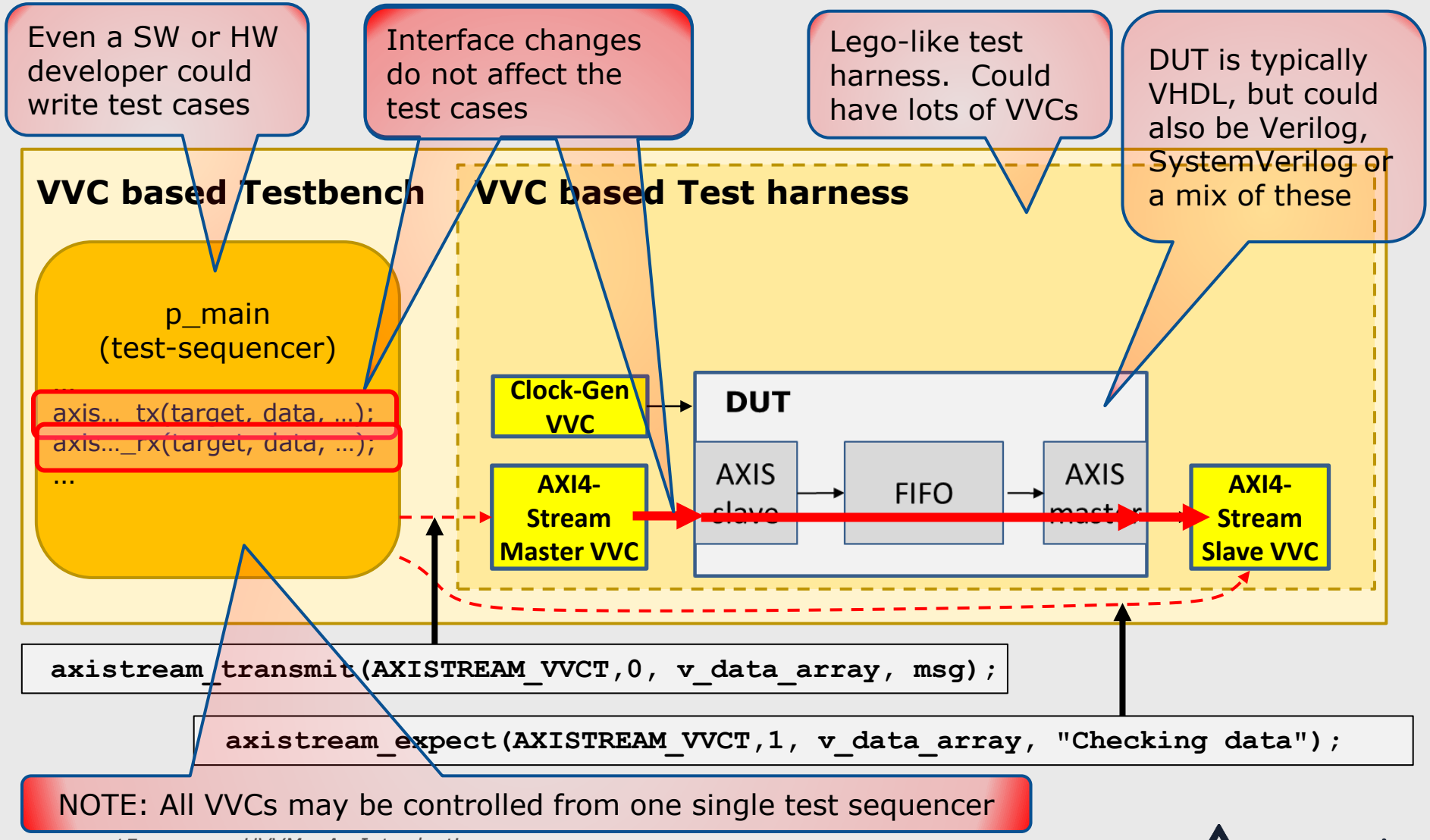
VVC: Easy to extend (2)

- **Easy to handle split transactions**
- **Easy to handle out of order execution**



EmLogic

VVC based TB



EmLogic

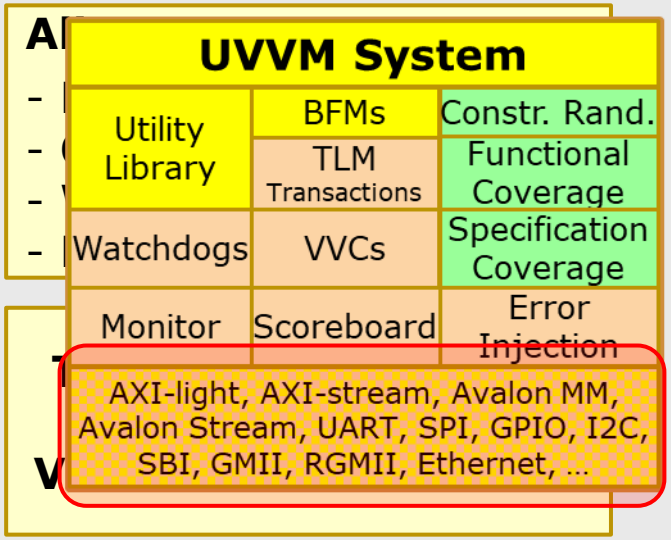
VVC Advantages

- Simultaneous activity on multiple interfaces
- Encapsulated → Reuse at all levels
- Queue → May initiate multiple high level commands
- Local Sequencers for predefined higher level commands
- **Unique for UVVM VVCs:**
  - Fully control **and** align all VVCs from a single sequencer!
  - May insert delay between commands – from sequencer
    → The only system to target cycle related corner cases
  - Simple handling of split transactions and out of order protocols
  - Common commands to control VVC behaviour
  - Simple synchronization of interface actions – from sequencer
  - May use Broadcast and Multicast

Better Overview, Maintenance, Extensibility and Reuse

Lot's of free UVVM BFMs and VVCs

- AXI4-lite
- AXI4 Full
- AXI-Stream Transmit and Receive
- UART Transmit and Receive
- SBI
- SPI Transmit and Receive
- I2C Transmit and Receive
- GPIO
- Avalon MM
- Avalon Stream Transmit and Receive
- RGMII Transmit and Receive
- GMII Transmit and Receive
- Ethernet Transmit and Receive
- Wishbone
- Clock Generator
- Error Injector



VVC: VHDL Verif. Comps.
- Includes the corresponding BFM

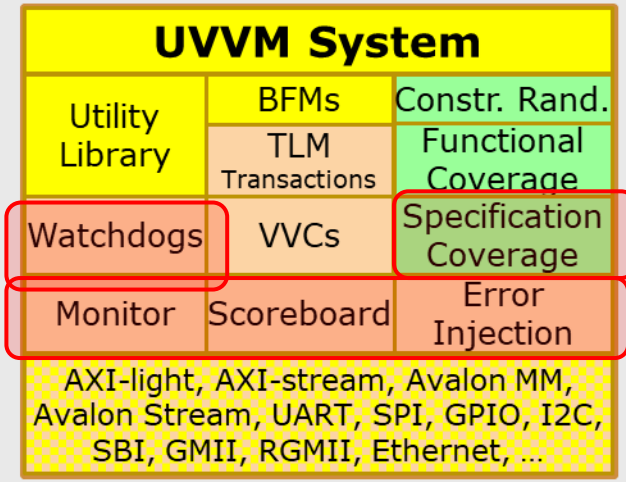
Allows:
- Simultaneous interface handling
- Synchronization of interfaces
- Skewing between interfaces
- Additional protocol checkers
- Local sequencers
- Activity detection
- Simple reuse between projects

EmLogic

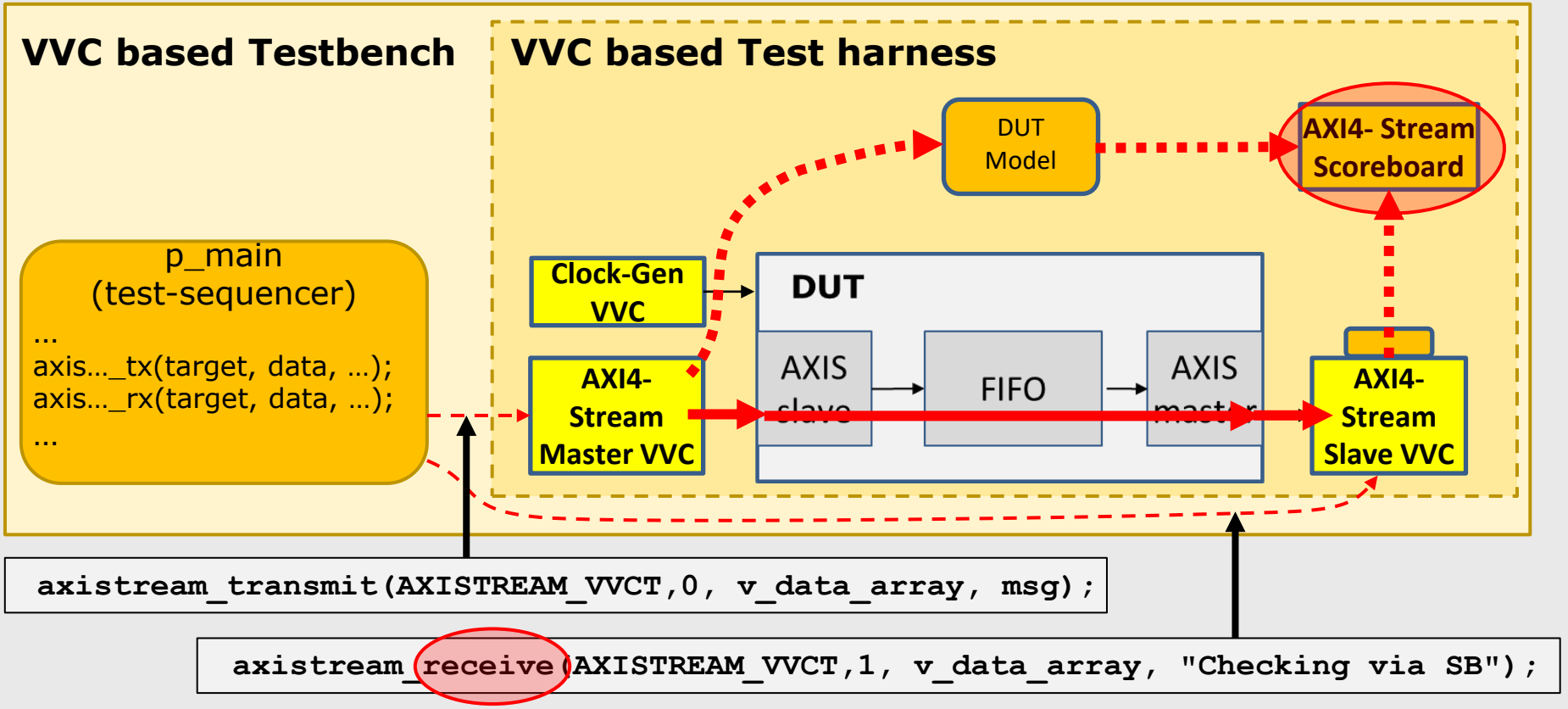
Added 2017-19 – in cooperation with ESA

- ESA Extensions in ESA-UVVM-1
  - **Scoreboards**
  - Monitors
  - Controlling randomisation and functional coverage
  - Error injection (Brute force and Protocol aware)
  - Local sequencers
  - Controlling property checkers
  - Transaction info
  - Watchdog (Simple and Activity based)
  - Hierarchical VVCs - And Scoreboards for these
  - **Specification Coverage** (Requirement/test coverage)

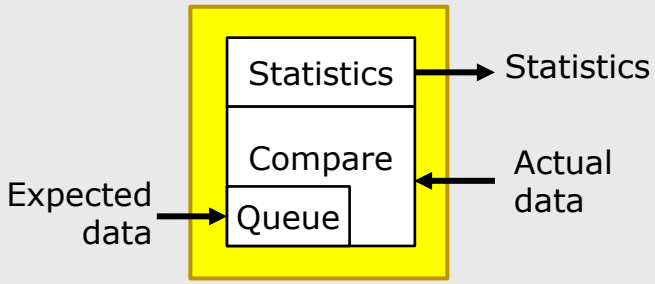
ESA is helping VHDL designers speed up FPGA and ASIC development and improve their product quality!

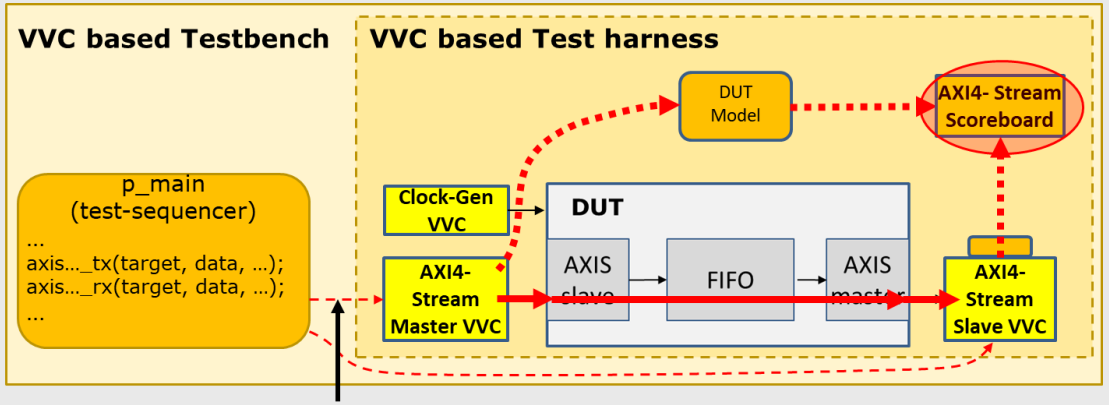


Advanced scoreboard-based TB



Generic Scoreboard





generic data type

- logging/reporting
- flushing queue
- clearing statistics

Configuration record:

- allow_lossy
- allow_out_of_order
- mismatch_alert_level

Counting:

- entered
- pending
- matched
- mismatched
- dropped
- deleted
- initial garbage

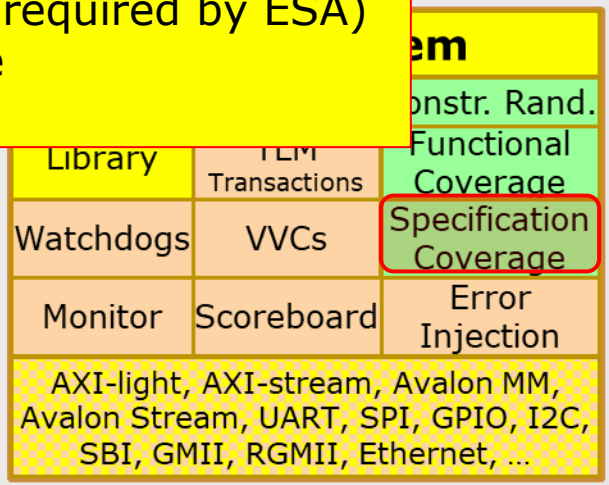
- insert, delete, fetch
- ignore_initial_mismatch
- indexed on either entry or position
- optional source element (in addition to expected + actual)

Specification Coverage

- Assure that all requirements have been verified
  1. Specify all requirements
  2. Report coverage from test sequencer(s) (or other TB parts)
  3. Generate summary report
     - Coverage per requirement
     - Test cases covering each requirement
     - Requirements covered by each Test case
     - Accumulate over multiple Test cases

Requirements Traceability Matrix

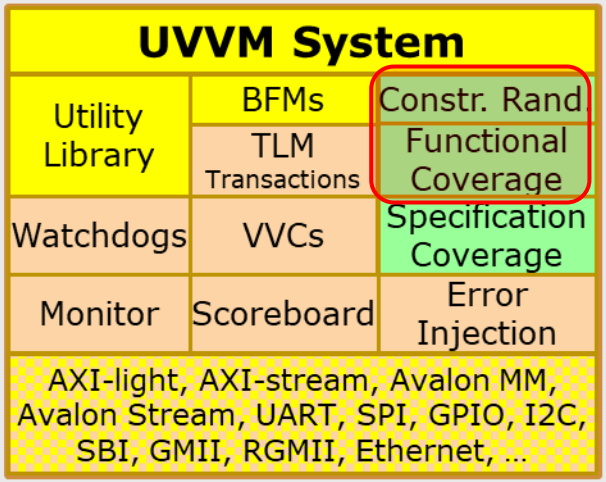
Mandatory for Safety and Mission Critical (Strictly required by ESA)
Strongly recommended for good quality assurance
Expensive tools exist…



The 2nd ESA project – 2020-22

- Enhanced Randomisation
  - Advanced randomisation in a simple way
- Optimised Randomisation
  - Randomisation without replacement
  - Weighted according to target distribution AND previous events

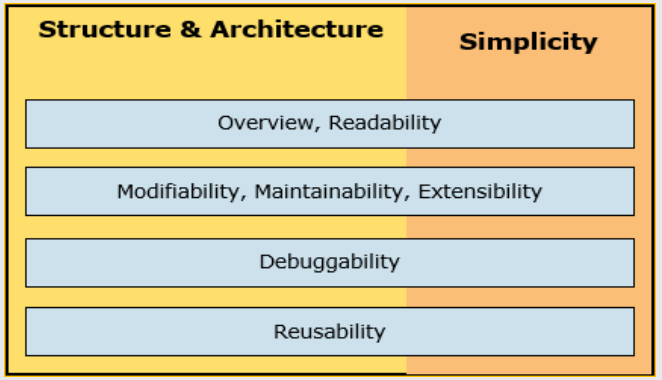
  → the lowest number of randomisations for a given target
- Functional Coverage
  - Checking that given scenarios have been verified



UVVM Enhanced Randomisation

- Well integrated with UVVM
  - Alert handling and logging in particular
- Strong focus on Overview & Readability
  - Adding keywords to ease understanding
- Easy to Maintain and Extend

Quality & Efficiency enablers



Typing code consumes is an insignificant part of the development time.

Reading and understanding code is repeated over and over again, and is thus a significant part of the development time

```
addr <= my_addr.rand(0, 18, ADD,(30,31), EXCL,(7));
```

```
addr <= my_addr.rand_range_weight((0,18,4),(19,31,1));
```

➔ **Investing in better code yields a huge return on investment**

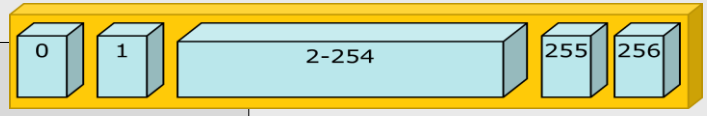
EmLogic

Functional Coverage (FC) – Typical Sequence

- Define a variable of type t_coverpoint

```
variable cp_payload_size : t_coverpoint;
```

- Add the bins




```
cp_payload_size.add_bins(bin(0));
cp_payload_size.add_bins(bin(1));
cp_payload_size.add_bins(bin_range(2,254,1));
cp_payload_size.add_bins(bin(255,256,2));
```

- Tick off bins as their corresponding payload size is used

```
cp_payload_size.sample_coverage(payload_size);
```

- Continue sending packets until coverage target is reached

```
while not cp_payload_size.coverage_completed(VOID);
```

UVVM also has transition coverage

Some FC reports – out of many

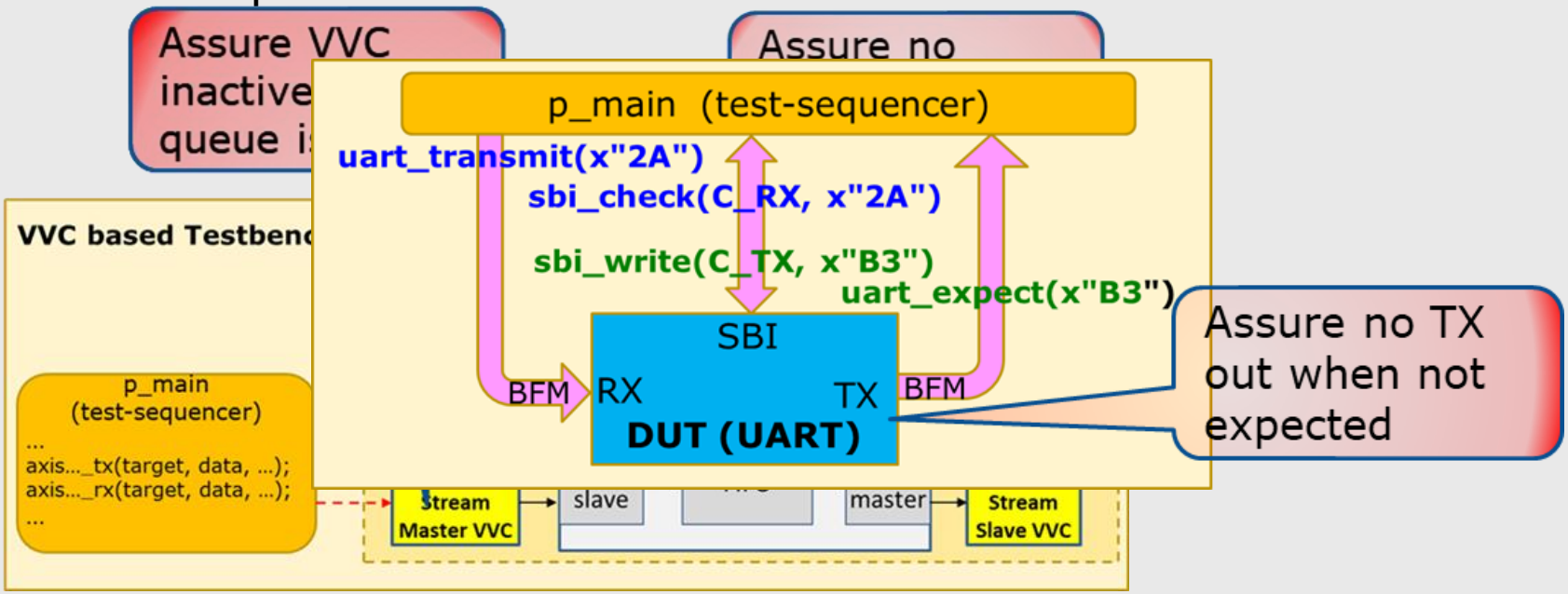
```
# UVVM:  ================================================================================================================
# UVVM:  0 ns *** COVERAGE SUMMARY REPORT (NON VERBOSE): TB seq. ***
# UVVM:  ================================================================================================================
# UVVM:  Coverpoint:                Covpt_1
# UVVM:  Coverage (for goal 100): Bins: 60.00%,   Hits: 76.47%
# UVVM:  ----------------------------------------------------------------------------------------------------------------
# UVVM:              BINS              HITS      MIN HITS     HIT COVERAGE          NAME            ILLEGAL/IGNORE
# UVVM:          (256 to 511)            1          N/A           N/A          illegal_addr           ILLEGAL
# UVVM:           (0 to 125)             6          8           75.00%         mem_addr_low             -
# UVVM:         (126, 127, 128)          3          1           100.00%        mem_addr_mid             -
# UVVM:          (129 to 255)            14         4           100.00%        mem_addr_high            -
# UVVM:          (0->1->2->3)            0          2           0.00%          transition_1             -
# UVVM:          transition_2            2          2           100.00%        transition_2             -
# UVVM:  ----------------------------------------------------------------------------------------------------------------
# UVVM:  transition_2: (0->15->127->248->249->250->251->252->253->254)
# UVVM:  ================================================================================================================
```

```
# UVVM:  ================================================================================================================
# UVVM:  0 ns *** OVERALL COVERAGE REPORT (VERBOSE): TB seq. ***
# UVVM:  Coverage (for goal 100): Covpts: 50.00%,   Bins: 73.68%,   Hits: 76.00%
# UVVM:  ================================================================================================================
# UVVM:     COVERPOINT    COVERAGE WEIGHT   COVERED BINS    COVERAGE(BINS|HITS)   GOAL(BINS|HITS)   % OF GOAL(BINS|HITS)
# UVVM:       Covpt_1           1              3 / 5         60.00% | 76.47%      50% | 100%        100.00% | 76.47%
# UVVM:       Covpt_2           1              3 / 3        100.00% | 100.00%     100% | 100%       100.00% | 100.00%
# UVVM:       Covpt_3           1              6 / 6        100.00% | 100.00%     100% | 100%       100.00% | 100.00%
# UVVM:       Covpt_4           1              0 / 4          0.00% | 0.00%       100% | 100%         0.00% | 0.00%
# UVVM:       Covpt_5           1              0 / 1          0.00% | 0.00%       100% | 100%         0.00% | 0.00%
# UVVM:       Covpt_6           1              4 / 4        100.00% | 100.00%     100% | 100%       100.00% | 100.00%
# UVVM:       Covpt_7           1              0 / 3          0.00% | 0.00%       100% | 100%         0.00% | 0.00%
# UVVM:       Covpt_8           1             12 / 12       100.00% | 100.00%     100% | 100%       100.00% | 100.00%
# UVVM:  ================================================================================================================
```

The 3rd ESA project – 2024

- Started March 2024

- First new features to be released in June
  - Completion detection
  - Detection of unwanted/unexpected interface activity
- More improvements to come



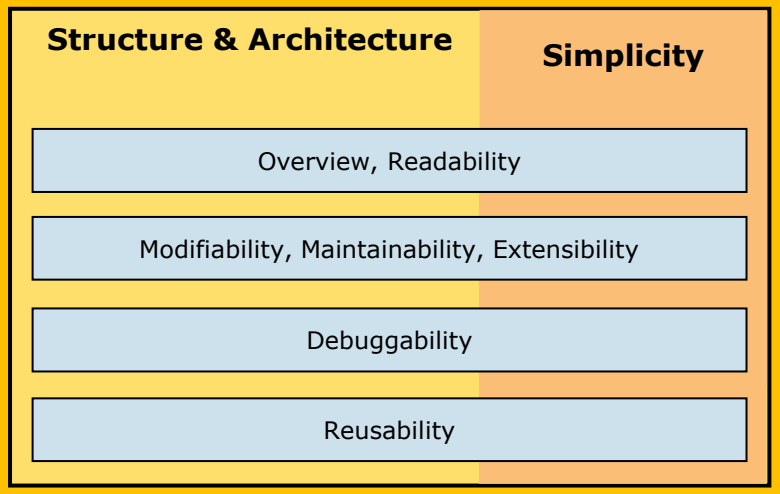
EmLogic

Sources of learning

- Very well documented
  - In transition from PDF to html
- Lots of free webinars available
  - From Siemens, Aldec, Trias, Verification Futures, etc
- New free introduction videos
  - Siemens' Xcelerated Academy
- EmLogic courses: https://emlogic.no/courses/
  - 'Advanced VHDL Verification - Made simple'
  - First planned live online course: November 2024
- Testbench examples
  - Note: Provided under the 'UVVM supplementary' repo
- New: Sharing training material with universities for free

UVVM in a nutshell

- Huge improvement potential for more structured FPGA verification



UVVM (incl. all) is Open Source

Game changer for efficiency & quality

UVVM has the largest collection of interface models (as BFMs and VVCs)

UVVM may save 200-2000 hours on a medium complex project

And at the same time improve TTM, MTBF & LCC

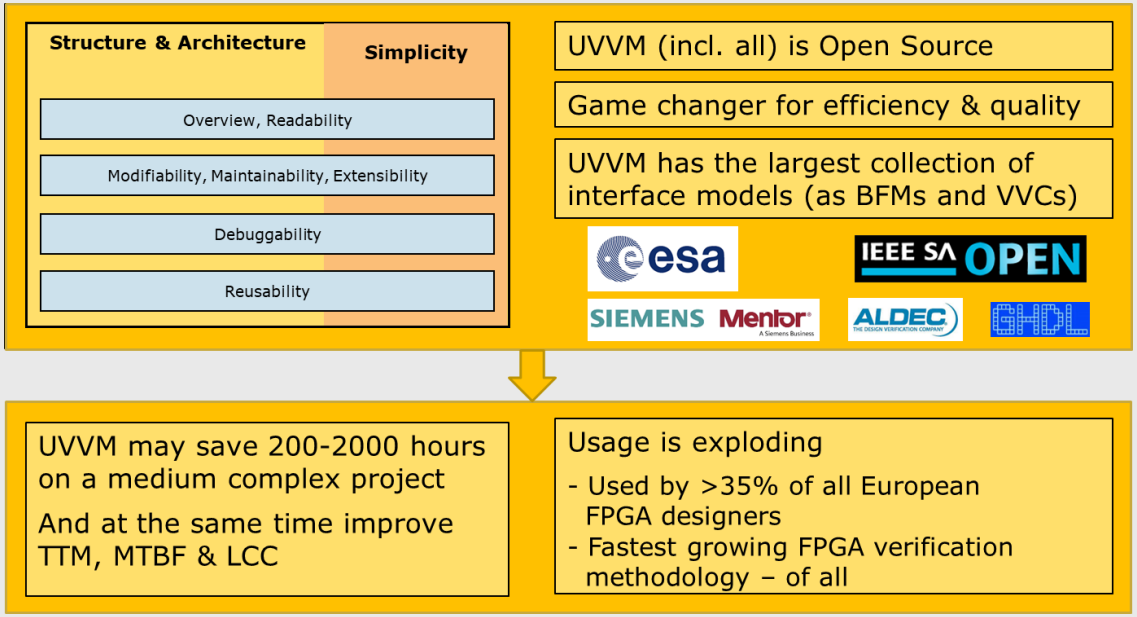
Usage is exploding

- Used by >35% of all European FPGA designers
- Fastest growing FPGA verification methodology – of all

EmLogic

Thank you for attending

UVVM
– An introduction to the world's fastest growing FPGA verification methodology



EmLogic

Feel free to connect on LinkedIn