

# Formal-Verification IPs

---

**DATE'08 OCP-IP Presentation  
with AerieLogic**



# AerieLogic



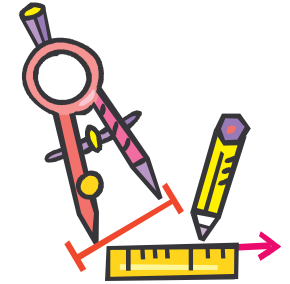
- A Formal VIP company
  - ◆ An experienced team in ABV solutions
    - Experts in building **libraries** of assertions
    - Specialists in **ABV** and **formal verification** usage
- Solutions deployed in the industry
  - ◆ Eases the use of ABV and Property Checking
  - ◆ Products have several **years of maturity**

# What is a Formal-VIP ?



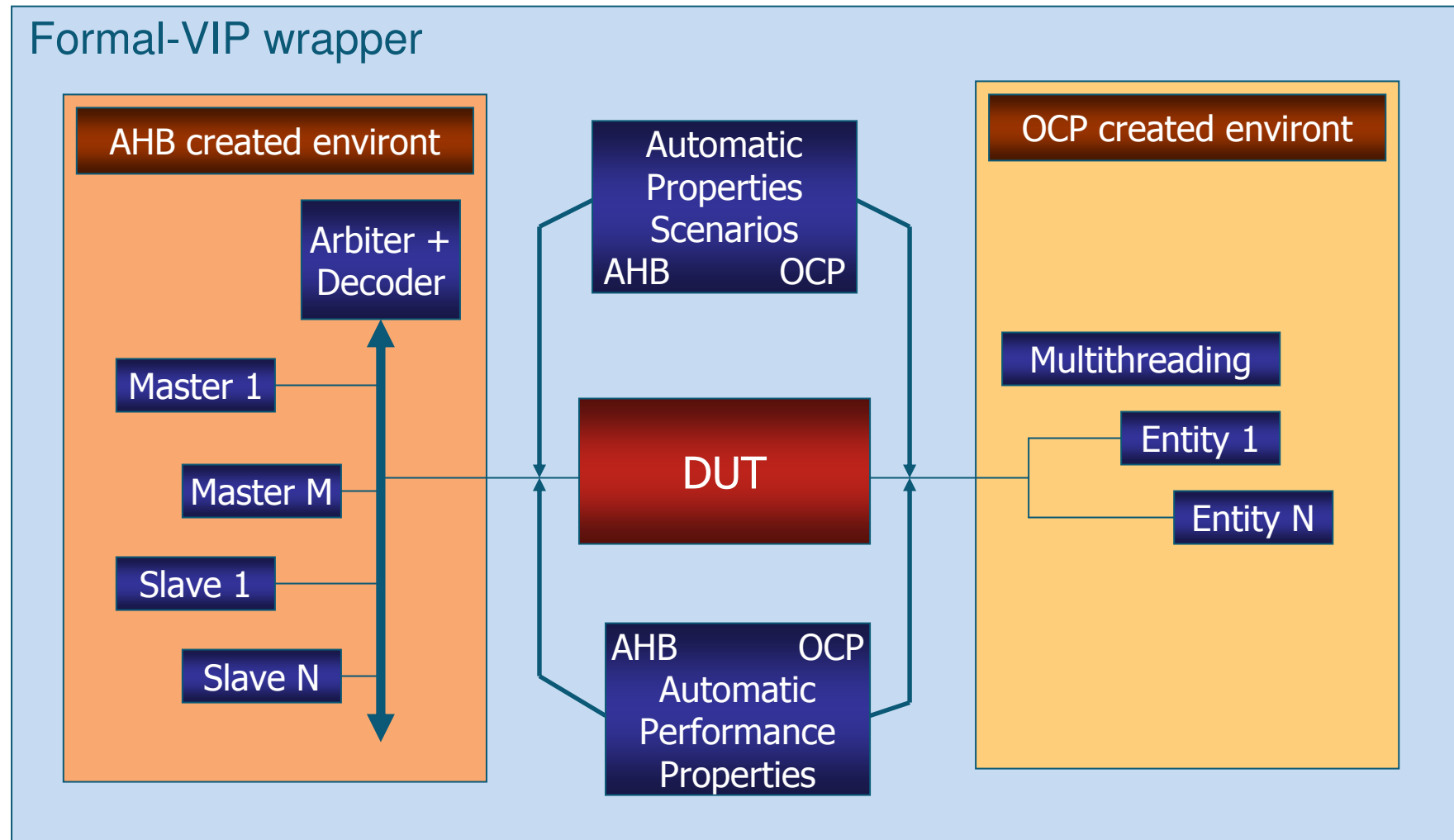
- It's a Verification IP working seamlessly with Property Checkers
  - ◆ Automatic usage – No knowledge in assertions or Property Checking needed
  - ◆ Complete documentation, scripts and report generation
  - ◆ Easy setting via GUI or CLI interface
- Provides automation to Property Checking
  - ◆ Builds the formal environment
  - ◆ Instanciates all assertions on the design
  - ◆ Creates regression scripts and collects data

# Exemple: Formal-VIP for OCP

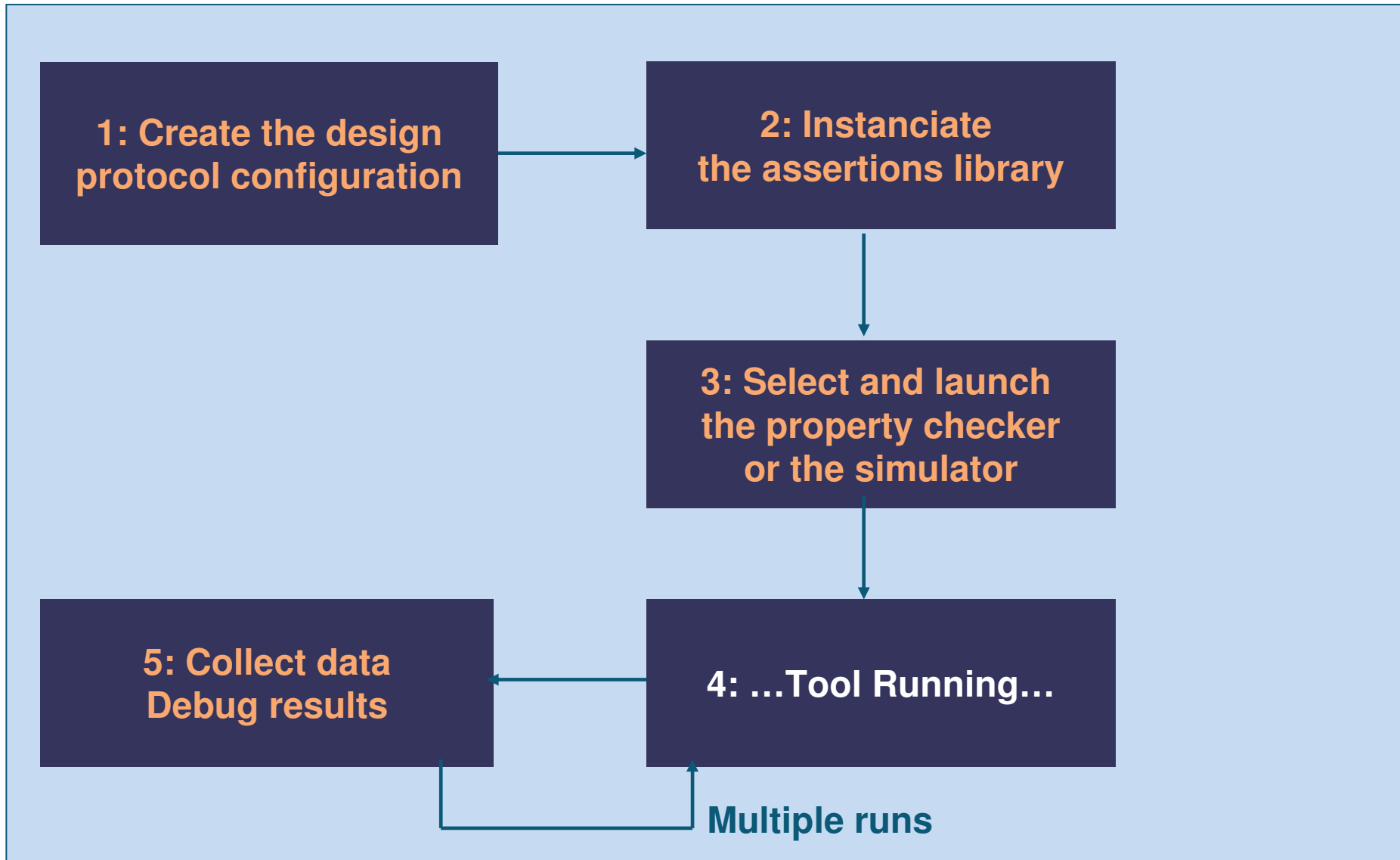


- For IP blocks based on the OCP 2.2/2.1/2.0/Sonics2.4/1.0 protocols
  - ♦ Automatic **instanciation** of assertions and formal environment
  - ♦ Automatic **formal verification** of properties and coverage scenarios
  - ♦ Formal **Proof**, or automatic VCD and testbench in case of **violation**
- Provides **100% compliance** against the OCP protocol specification
  - ♦ Fully aligned on the OCP-IP Compliance document
  - ♦ including multi-threading, tagging, datahandshake, ...
  - ♦ More than 110 properties and hundreds of coverage scenarios
- Provides **100% functional performance** analysis
  - ♦ For detection of dead-locks, loss of data, back-pressure, ...
  - ♦ *Ex: A thread will be busy at most N consecutive cycles*
  - ♦ More than 15 properties dedicated to performance analysis

# Example: AHB-OCP design



# Formal-VIP flow chart



# 1: Create the protocol configuration

Reload/Save configurations  
Create <rtl.conf> for OCP

syntax and semantic checks

Multi-interfaces support  
Straightforward to use

Bubbles for inline help

HPKWizard v.1.5.10  
File Configuration

HPK-AHB Master 1 (1)

not supported SUPPORT\_BURST\_INCR=NO and ALWAYS\_RESTART\_BURST\_WITH\_INCR=YES

OCP\_2.1 interface  
AHB\_interface

**Master Parameters**

REAL DESIGN

DATA BUS WIDTH 32 SUPPORT SINGLE TRANSFER

Support Burst

INCR  INCR4  INCR8  INCR16   
WRAP4  WRAP8  WRAP16

Support Size

SIZE  SIZE16  SIZE32  SIZE64   
SIZE128  SIZE256  SIZE512  SIZE1024

Others

ALWAYS RESTART BURST WITH INCR  ALWAYS RESTART BURST WITH SINGLE   
ALWAYS CANCEL BURST ON ERROR  NEVER CANCEL BURST ON ERROR   
CAN CANCEL  Indicates if the master always cancels its burst transfers after receiving an error response

IS DEFAULT MASTER  MASTER MAX BUSY CYCLES 3

MASTER MAX LOCKED CYCLES 7 BURST FOUR MAX CYCLES 12

BURST EIGHT MAX CYCLES 16 BURST SIXTEEN MAX CYCLES 26

Add Rename Remove Save Close

# 2: Instanciate the assertions library

Instanciate multi-interfaces and multi-protocols

Select which assertions to instanciate

Bubbles provide Information on each property

Log results

**Configuration**

ocp  
s\_axi

Interface : ocp Port : ocp Protocol : OCP 2.2

Compliance properties

- burst\_hold\_MBurstLength\_precise\_0
- burst\_hold\_MBurstSeq\_0
- burst\_hold\_MCmd\_0
- burst\_hold\_MReqInfo\_0

Performance properties

- fvip\_performance\_max\_cycles\_MCmd\_IDLE\_2\_0
- fvip\_performance\_max\_cycles\_burst\_2\_6\_0
- fvip\_performance\_pipeline\_response\_master\_10\_0

**8.2.10 burst\_hold\_MCmd**

Assertion name  
\* %(P)burst\_hold\_MCmd\_\$(i)  
with 0 = i = threads - 1  
For a given thread, the command (MCmd) must remain constant across all transfers of a burst issued within that thread.

References  
\* (OCP 2.1 p47, section "Constant Fields in Bursts")  
\* (OCP 2.1 p199, section "Burst checks", bullet 2)

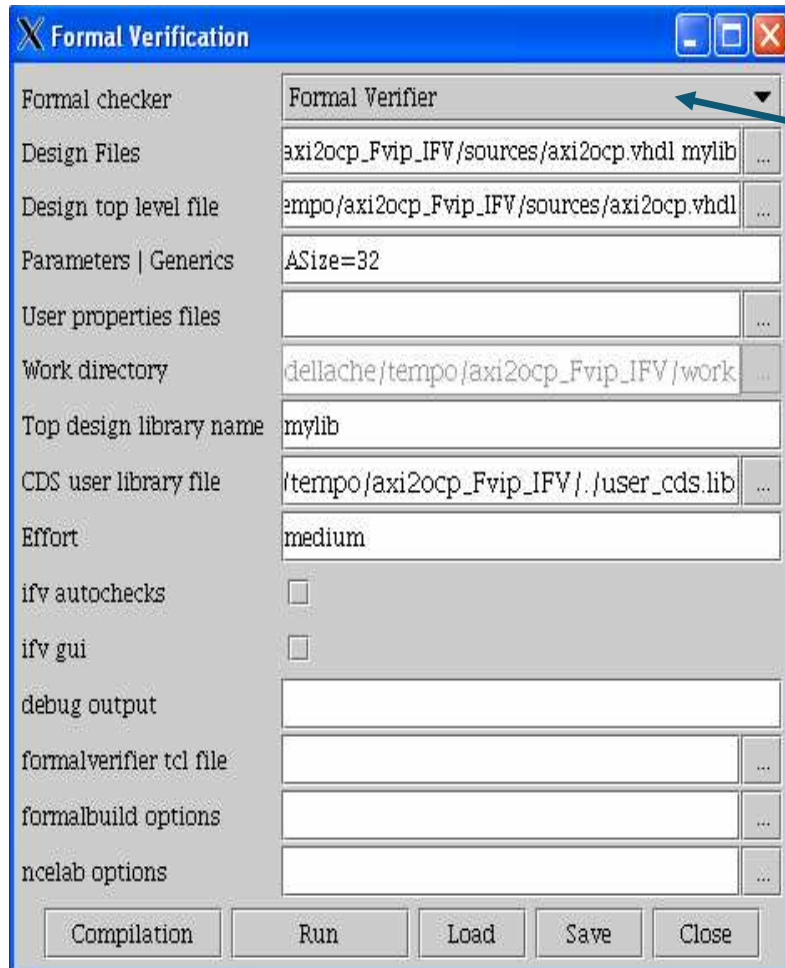
Filter (regexp) :  
Coverage :  
References :  
Filter (regexp) :

Clear Log    Formal Verification    Simulation    Close

**Log**

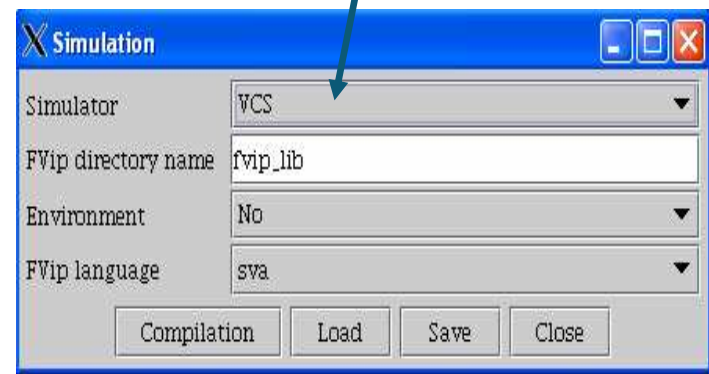
```
OCP 2.2> WARNING: pec_reset_length is less than 16 clock cycles
OCP 2.2> WARNING: The MReqLast signal is present in your design whereas the parameter reqlast is deactivated
OCP 2.2> BUILDING LIBRARY FOR AN ENVIRONMENT DUT OCP CORE SLAVE INTERFACE --- OCP 2.2 FVip v1.7.1 ...
OCP 2.2> BUILDING LIBRARY FOR THE DUT OCP CORE MASTER INTERFACE --- OCP 2.2 FVip v1.7.1 ...
```

# 3: Select the Tool



Select Property Checker

Select Simulation



# 4: Run the selected Tool

5: Verification results  
Are provided by the  
Selected Tool

The user can debug,  
modify the design and  
launch the verification  
at will

A batch mode is also  
available for steps 1 to 5

Log Window

The screenshot shows the 'FVip-Wizard Export' application. The 'Configuration' tab is active, showing 'Interface : s\_axi', 'Port : s\_axi', and 'Protocol : AXI 1.0'. A 'Formal Verification' dialog box is open, with 'Formal checker' set to 'Solidify'. The 'Log' window at the bottom shows the following text:

```
77 | properties.psl | 706| ocpfvip_coverage_trivial_MCmd_0 | axitooocp_wrapper|
78 | properties.psl | 709| ocpfvip_performance_cycles_burst_2_6_0 | axitooocp_wrapper|
79 | properties.psl | 712| ocpfvip_coverage_trivial_Clk | axitooocp_wrapper|
-----
autoverify command executed successfully.
Quitting Solidify...
-----
Verification Done
-----
-> This verification log is available in file /users/retd/dellache/tempo/axi2ocp_Fvip_Solidify_reduced/work/trace.solidif
```

# Formal-VIP Success-Story



- Background
  - ◆ 38 on-the-shelf protocol-based IPs
    - Had been heavily simulated with simulation VIPs
  - ◆ 2 people part-time during 3 months
  - ◆ World-wide semi-conductor company
- Results of F-VIP with Formal Verification
  - ◆ 9 functional bugs automatically found
  - ◆ Tens of wrong protocol documentations
  - ◆ 2 hours of work per IP on average

# Formal VIPs availability

- AHB 2.0 and APB 2.0
- AXI 1.0 and APB 3.0
- I<sup>2</sup>C 2.1 and 3.0
- DDR/DDR2/DDR3
- Bridge specialized Formal-VIP
- OCP 2.2/2.1/2.0
- OCP 1.0 and Sonics2.4

Other protocols upon demand

# Conclusion

- **AerieLogic provides solutions to structure the SoC functional verification flow**
  - ♦ Built on protocols such as AXI, AHB and OCP
    - Back-bone of SoC verification
  - ♦ No need for expert knowledge
    - Property Checker and assertions handled automatically
  - ♦ Easily **integrated** in any verification flow
    - Formal-VIPs work automatically with several formal verification tools and simulators of the market