AMIQ

DVT IDE

From GVIM/EMACS to Modern IDEs: A Verification Engineer's Journey with DVT

Netanel Miller

AI & Verification Innovation Lead, Texas Instruments

About ME

My Journey

2012 Electrical Engineering at Bar Ilan University

2016 Joined TI as Verification Engineer

2019 Technical Leader & Founder for next-gen SOC

2020 Verification Team Leadership role

2025 Leading AI initiatives for Hardware Development

Passion for innovation

- Continuous exploration of emerging EDA tools and technologies
- Advocate for adopting transformative development methods

"Always at the intersection of hardware engineering and cutting-edge technology"

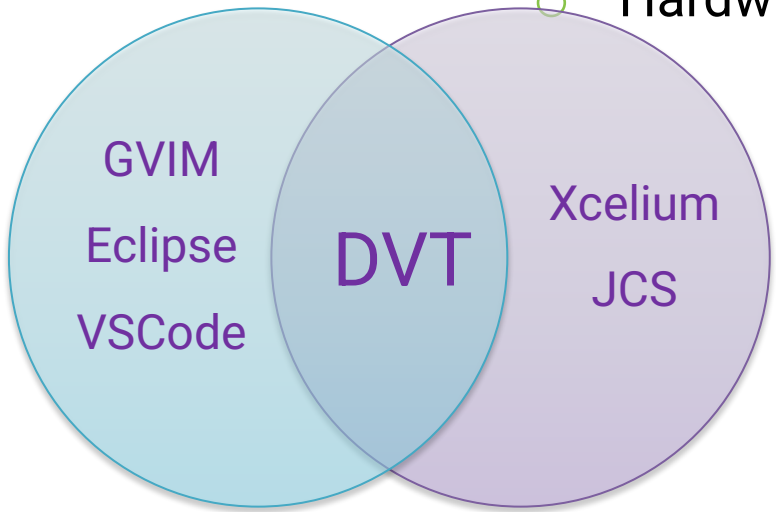
Editor vs. IDE: Understanding the Difference

Text Editor / Base IDE

- GVIM, VS Code, Eclipse (without extensions)
- Text manipulation and syntax highlighting
- Generic project organization

Hardware-Aware IDE (DVT)

- Domain-specific intelligence
- Design hierarchy understanding
- Hardware language semantics



“Editors know text; IDEs understand your design”
"The difference between typing code and engineering systems"

Introduction

PAST → PRESENT → FUTURE

The Starting Point

- Traditional text editors (GVIM)
- Manual navigation and lookups
- Joined TI 10 years ago to this reality

The "Aha" Moment

- Discovery of DVT's comprehensive solution
- Design and Verification in one environment
- Intelligence-driven development

AI Integration

- DVT + VS Code + AI Copilots
- Hardware-aware artificial intelligence
- Transforming how we design systems

"A journey from text editing to intelligent design assistance"

The DVT Advantage

Features Deep Dive

Not a full demo - just my everyday essentials

DVT's Universal Compiler Technology

"Design with confidence - your code, any tool, any standard"

Comprehensive Language Support

IEEE HW description language standards:

- SystemVerilog (IEEE 1800-2012/2017/2023)
- VHDL (IEEE 1076-1987 through 2008)
- Specman e language (IEEE 1647-2011)
- Mixed-language environments

Power Format Standards

- UPF (IEEE 1801-2013/2015/2018)
- CPF compatibility
- Multi-power domain visualization

Code navigation

- **Hyperlink** to declaration of anything:
  - class, module, struct, field, method, signal, macro, `included file, …

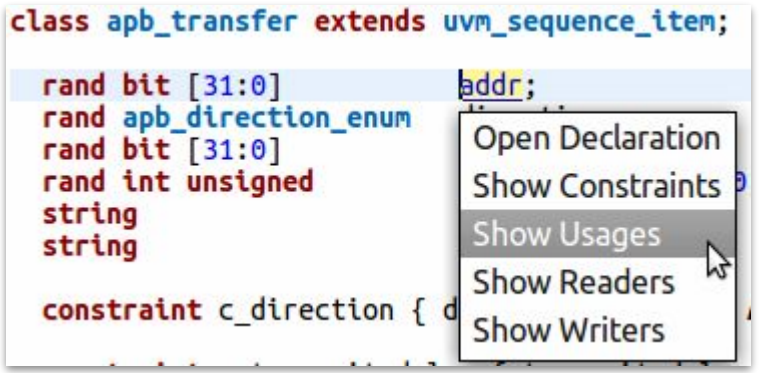
```
 99      virtual_sequencer = apb_subsystem_virtual_sequencer::type_id::create("virtual_sequencer",this);
100      ahb0              = ahb_pkg::ahb_env::type_id::create("ahb0",this);
101                        = apb_pkg::apb_env::type_id::create("apb0",this);
102                        = uart_pkg::uart_env::type_id::create("uart0",this);
103                        = uart_pkg::uart_env::type_id::create("uart1",this);
104                        = spi_pkg::spi_env::type_id::create("spi0",this);
105                        = gpio_pkg::gpio_env::type_id::create("gpio0",this);
```

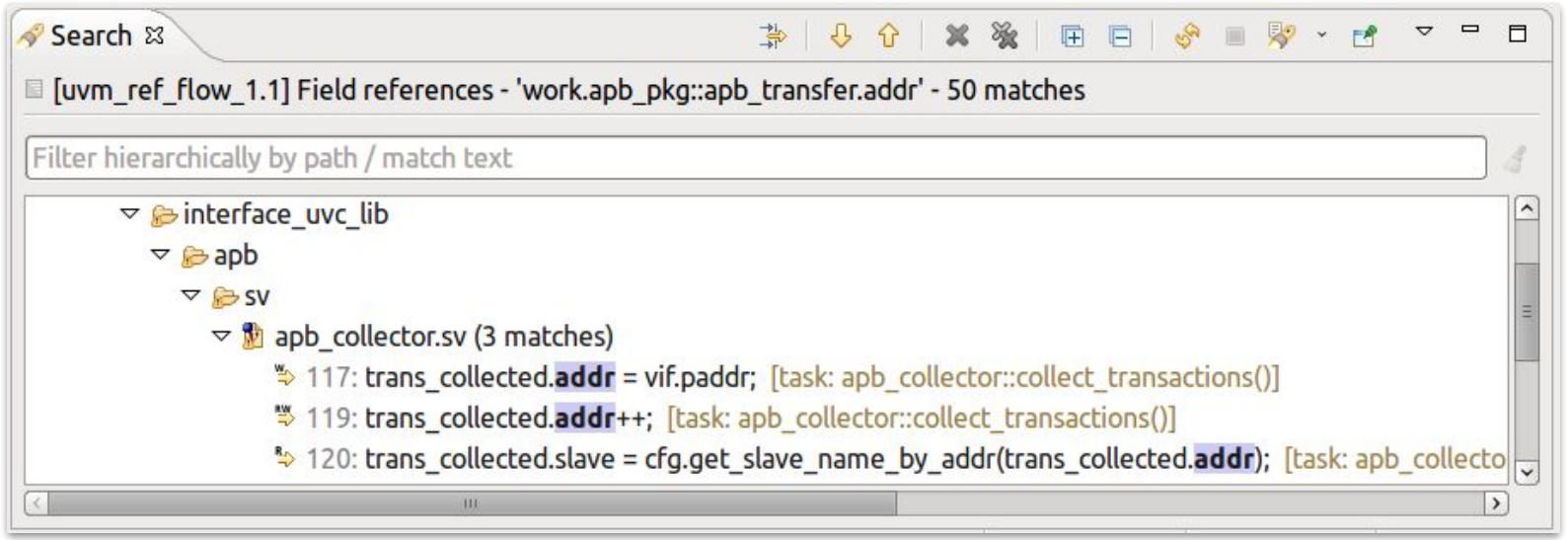
Open Declaration
Open Field Type
Show Usages

```
32 class apb_subsystem_tb extends uvm_env;
33
34   apb_subsystem_virtual_sequencer virtual_sequencer;  // multi-channel sequencer
35   ahb_pkg::ahb_env ahb0;                              // AHB UVC
36   apb_pkg::apb_env apb0;                              // APB UVC
```

Code navigation

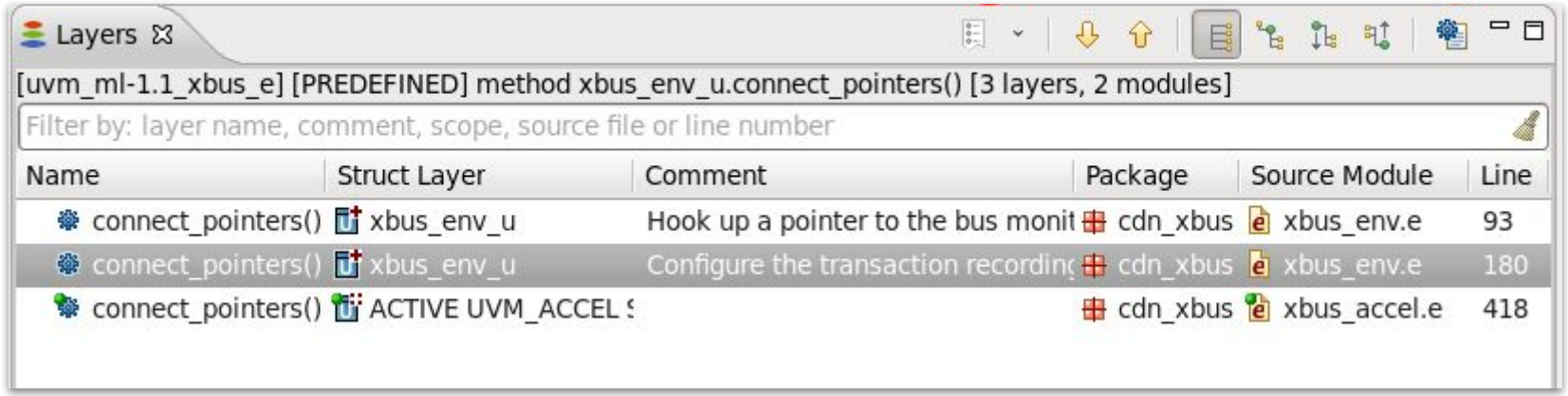
- **Hyperlink** to declaration of anything:
  - class, module, method, signal, macro, `included file, …
- Show **usages** of anything:
  - **Readers / Writers** of any variable / signal
  - **Constraints** of a rand class variable





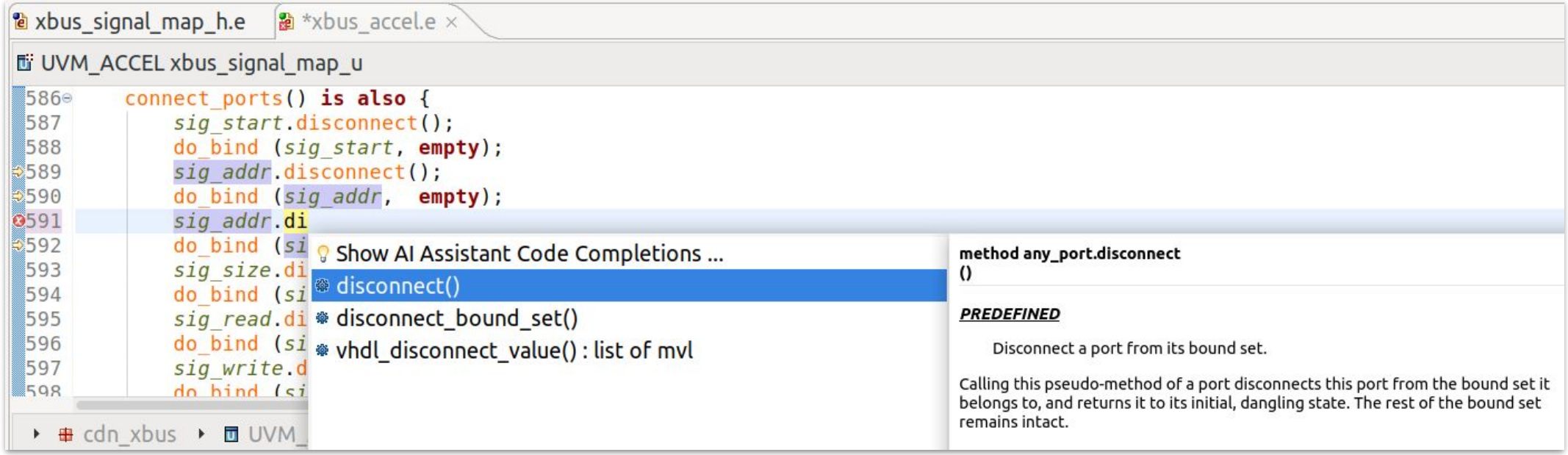
Code navigation

- **Hyperlink** to declaration of anything:
  - class, module, method, signal, macro, `included file, …
- Show **usages** of anything:
  - **Readers / Writers** of any variable / signal
  - **Constraints** of a rand class variable
- "**Layers**" of types, structs/units, methods, events, covergroups, … (**e** Language ;)



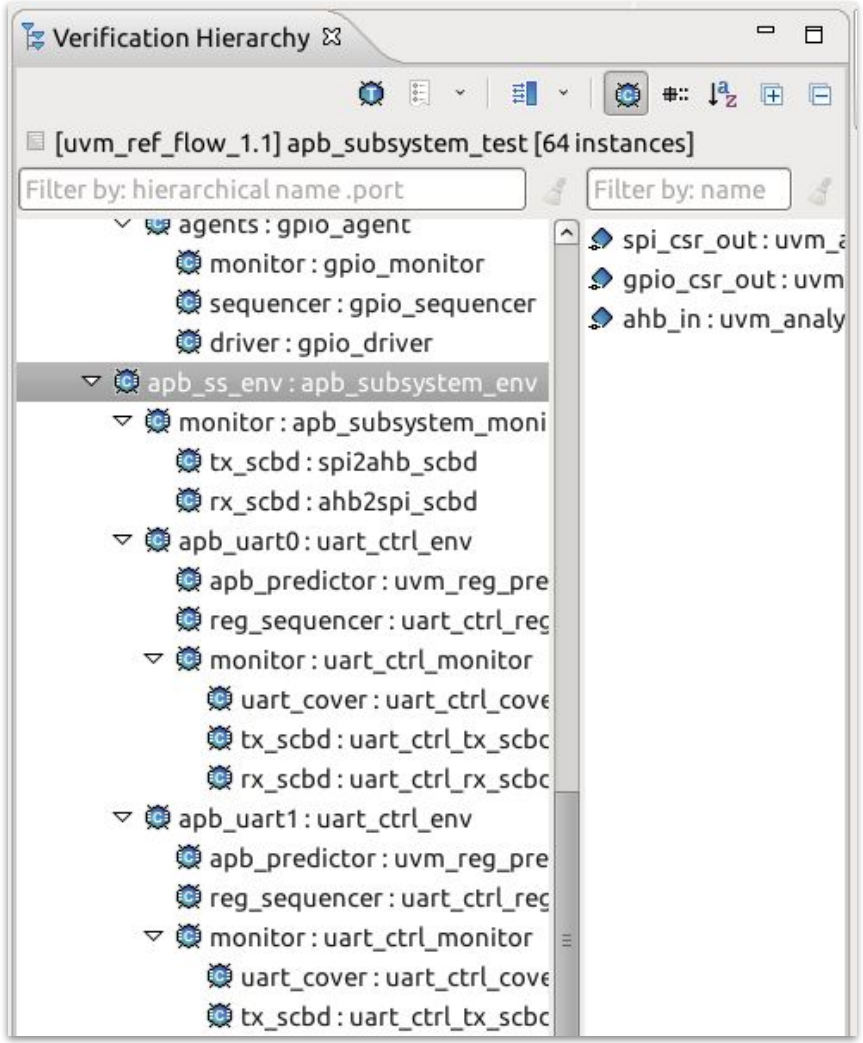
Auto-complete

- **Context-sensitive:** NOT textual proposals, only valid completions
  - `class_variable .` <shows fields, methods, constraints, … in the class>
  - `e_port .` <shows hdl_path(), put_mvl(), agent(), …>
  - `enum_variable ==` <shows only enum values>



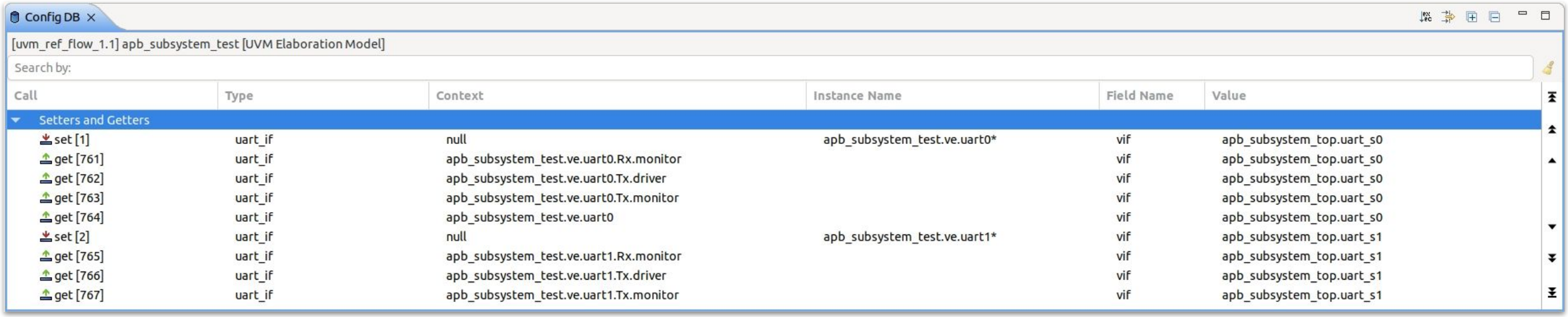
UVM Support

- Perform **UVM Runtime Elaboration**
- **Verification Hierarchy**
  - Browse & search accurate test topology



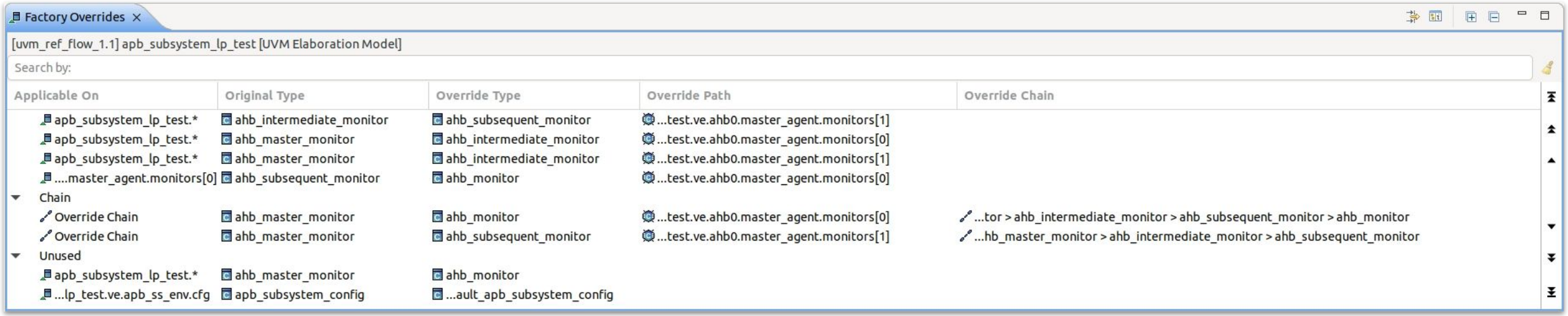
UVM Support

- Perform **UVM Runtime Elaboration**
- **Verification Hierarchy**
- **Config DB**
  - Browse config_db set/get calls
  - Inspect DB values



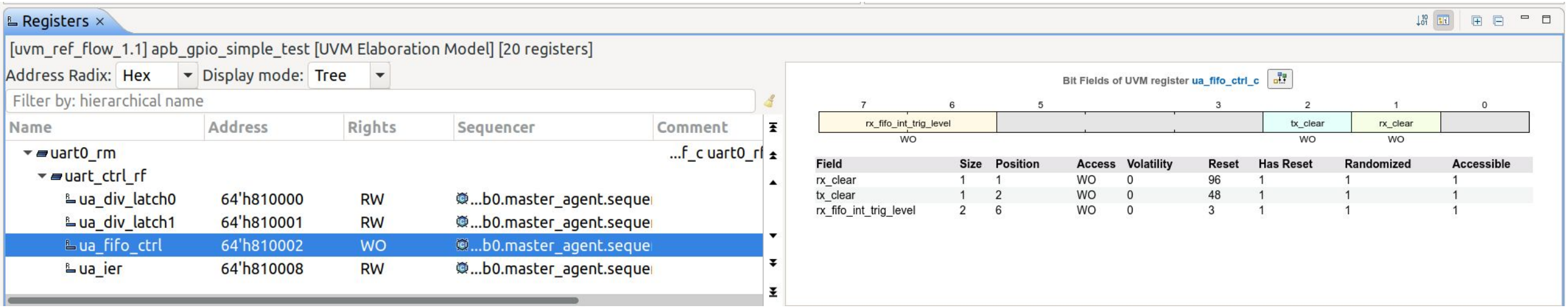
UVM Support

- Perform **UVM Runtime Elaboration**
- **Verification Hierarchy**
- **Config DB**
  - Browse config_db set/get calls
  - Inspect DB values
- **Factory Overrides**
  - Untangle overrides applied by the UVM factory



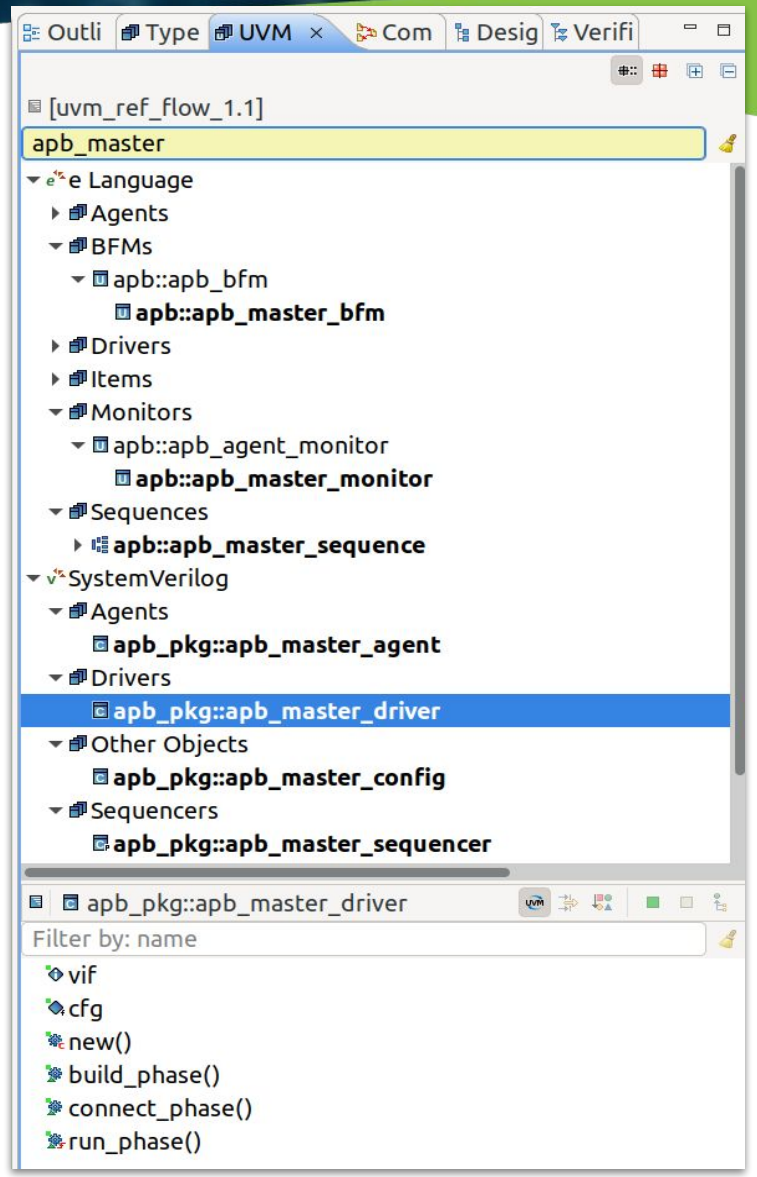
UVM Support

- Perform **UVM Runtime Elaboration**
- **Verification Hierarchy**
- **Config DB**
  - Browse config_db set/get calls
  - Inspect DB values
- **Factory Overrides**
  - Untangle overrides applied by the UVM factory
- **Registers**
  - Browse reg blocks
  - Visualize reg bitfields



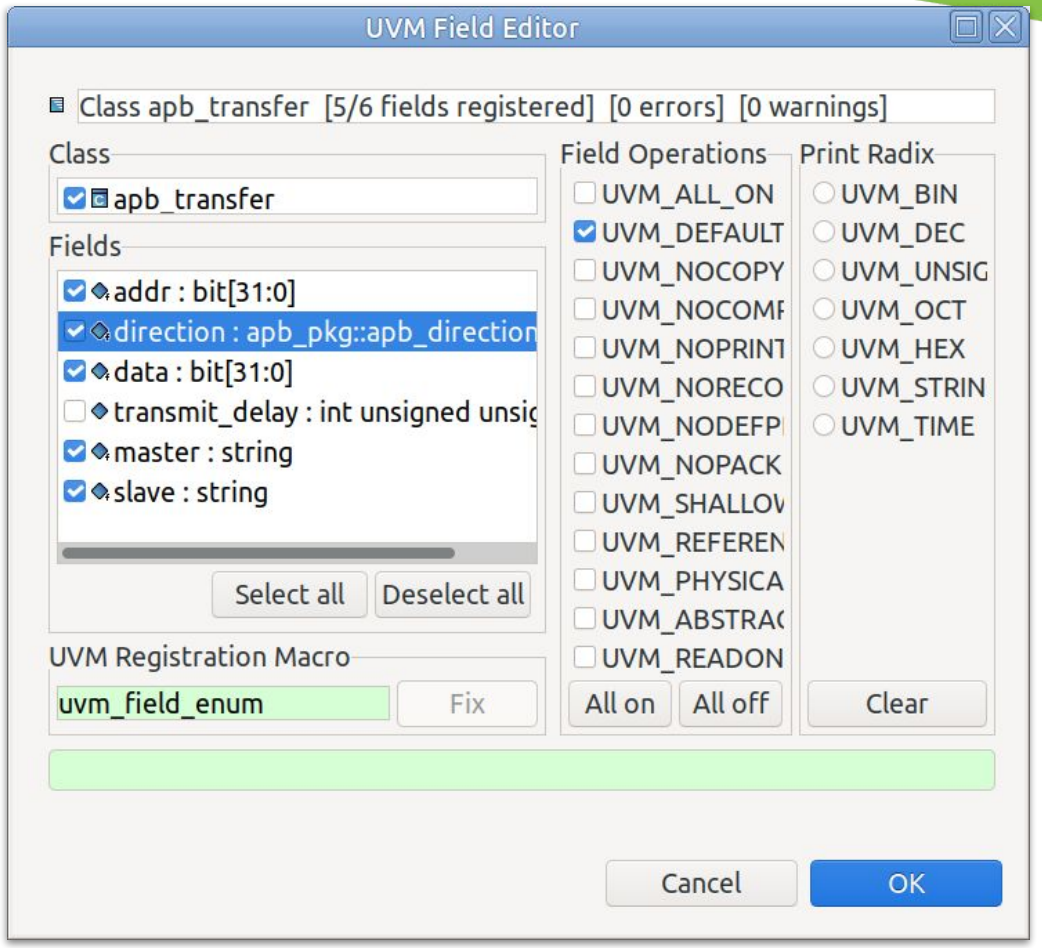
UVM Support

- **UVM Browser**
  - UVM based classes grouped by category
  - Mixed-language SV & e Language
  - UVM flow specific API
    - Overridden phases
    - Factory registered fields
    - TLM ports



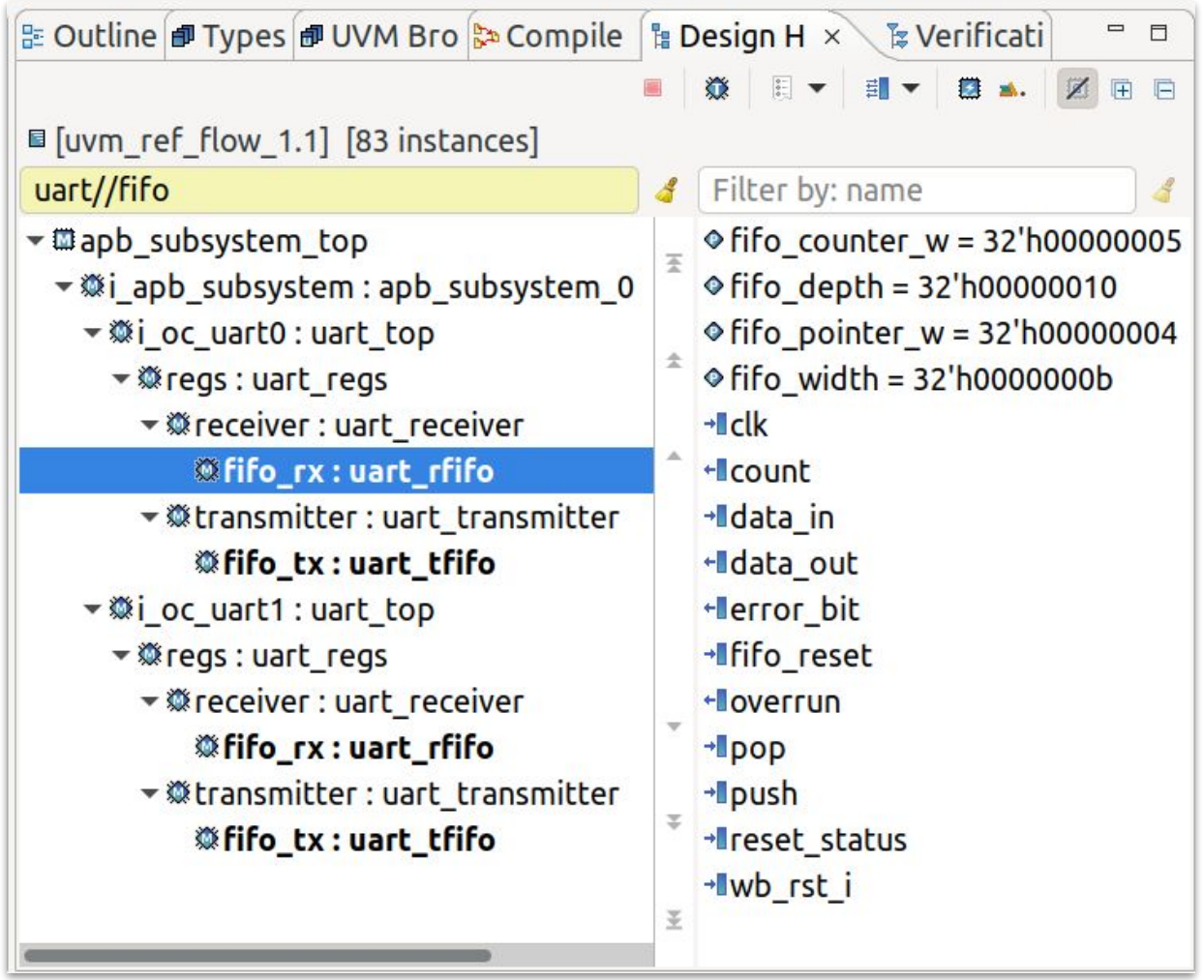
UVM Support

- **UVM Browser**
  - UVM based classes grouped by category
  - Mixed-language SV & e Language
  - UVM flow specific API
    - Overridden phases
    - Factory registered fields
    - TLM ports
- **UVM Field Editor**
  - Inspect and edit UVM field registrations
  - Auto-detects correct macro for each field



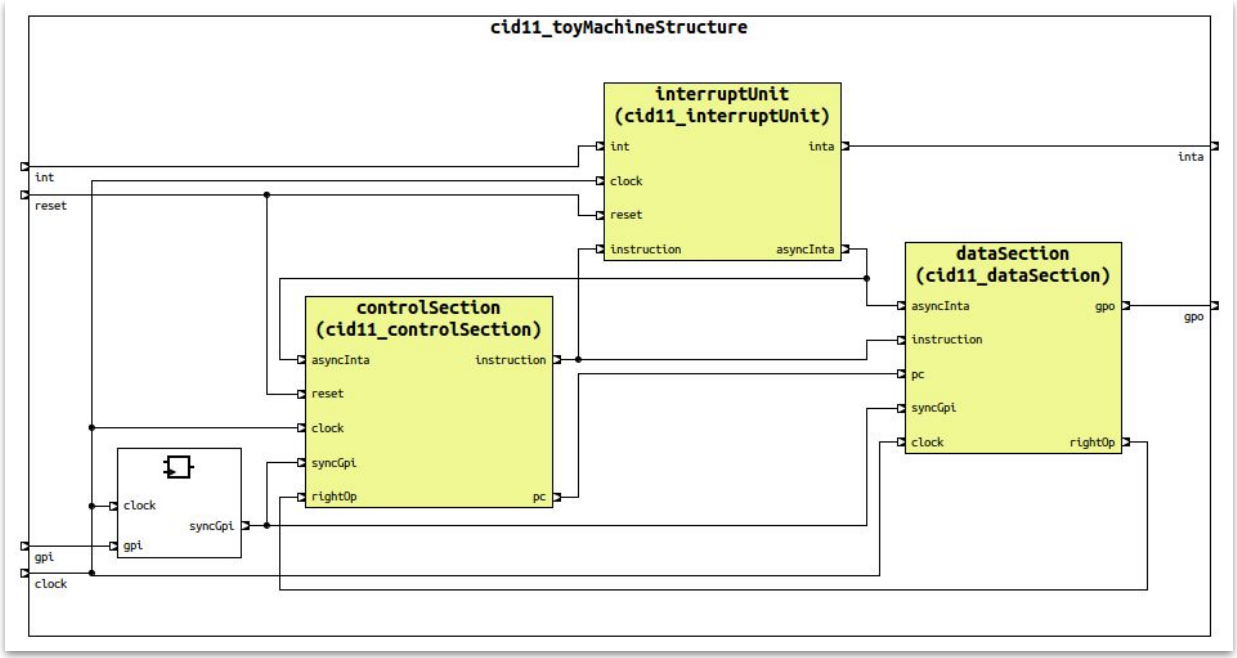
Visualization & exploration

- **Design Hierarchy View**
  - No need to open the simulation
  - Quick search (by instance name, by hierarchy, by port name)
  - Track elaborated param values



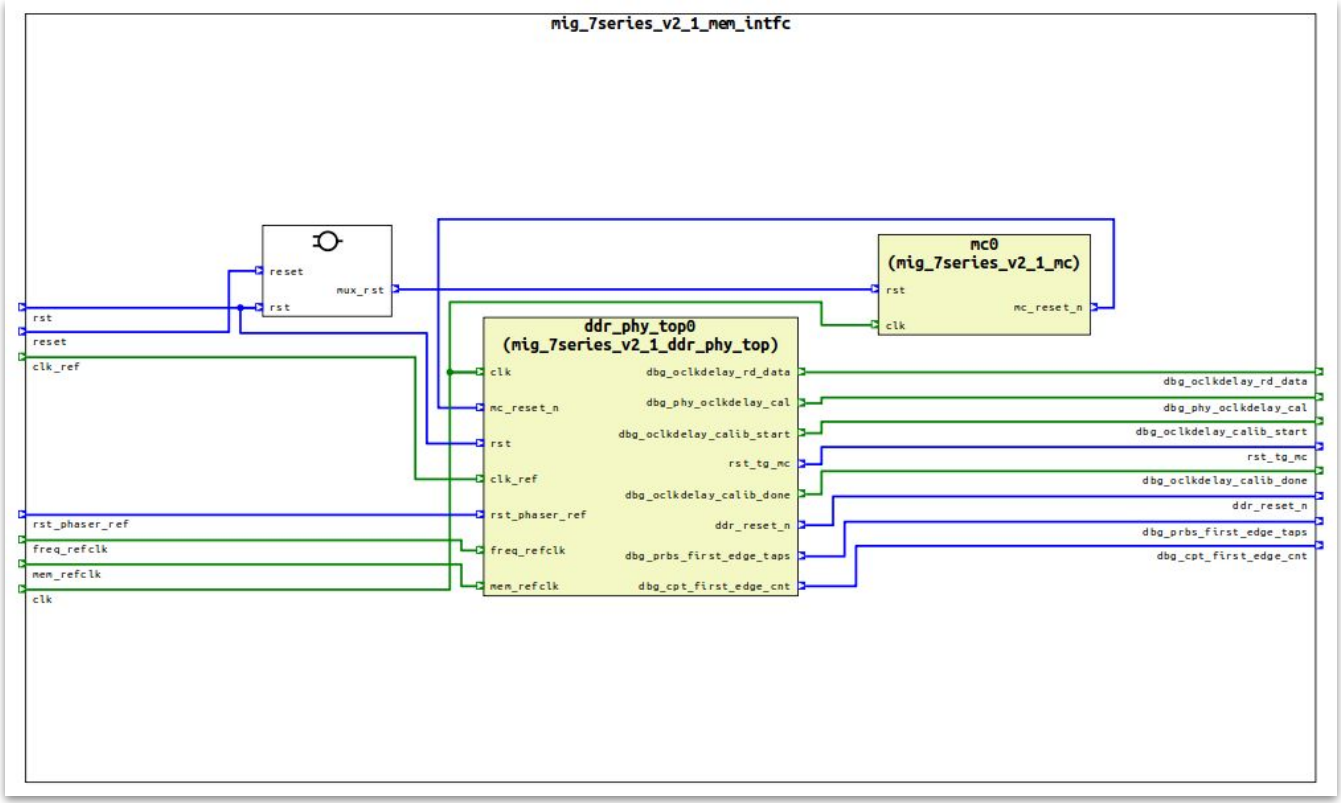
Visualization & exploration

- **Design Hierarchy View**
  - No need to open the simulation
  - Quick search (by instance name, by hierarchy, by port name)
  - Track elaborated param values
- **Schematic Diagrams**
  - **Show ports, sub-instances, logic blocks and connections**
  - Customizable (filters, colors)
  - Allows step by step tracing



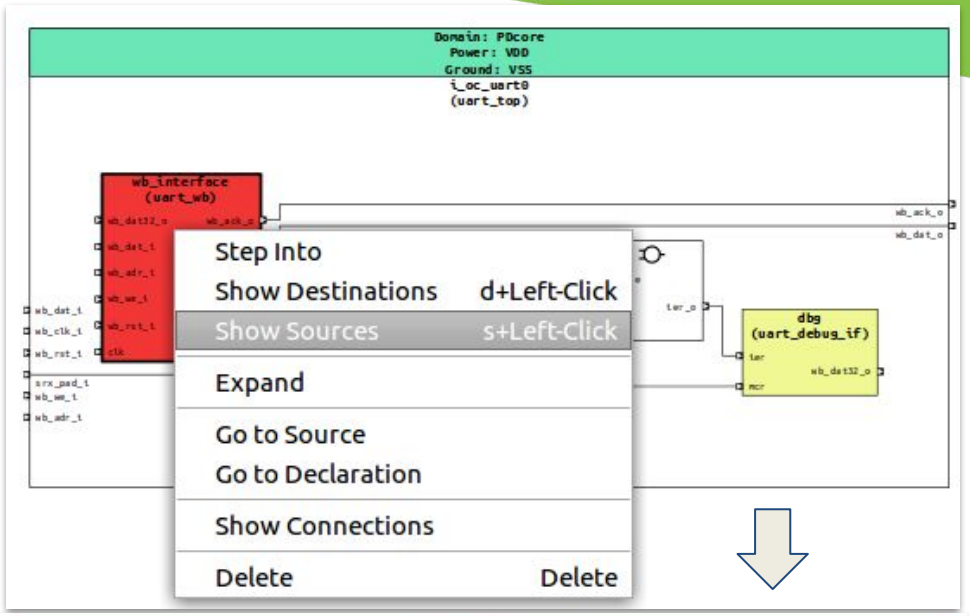
Visualization & exploration

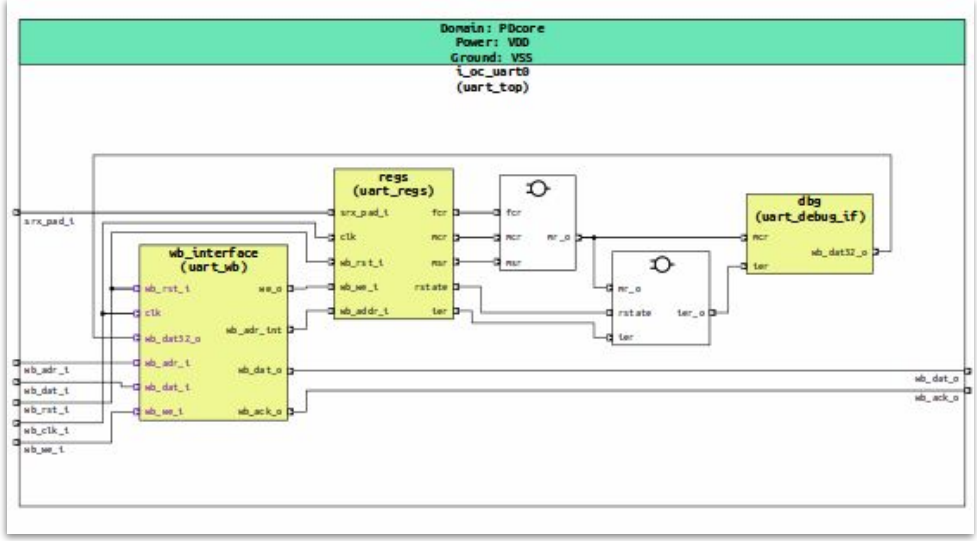
- **Design Hierarchy View**
  - No need to open the simulation
  - Quick search (by instance name, by hierarchy, by port name)
  - Track elaborated param values
- **Schematic Diagrams**
  - Show ports, sub-instances, logic blocks and connections
  - **Customizable (filters, colors)**
  - Allows step by step tracing



Visualization & exploration

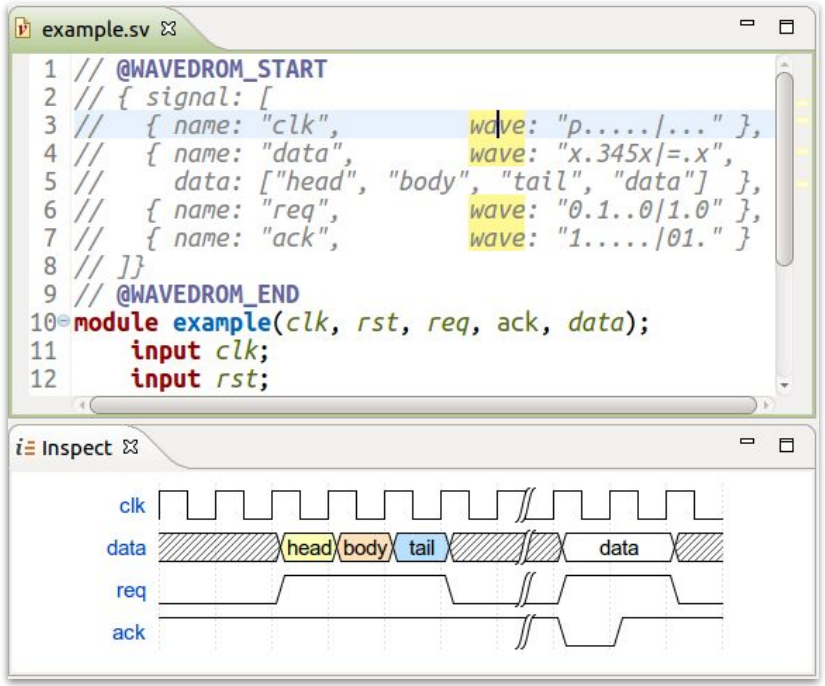
- **Design Hierarchy View**
  - No need to open the simulation
  - Quick search (by instance name, by hierarchy, by port name)
  - Track elaborated param values
- **Schematic Diagrams**
  - Show ports, sub-instances, logic blocks and connections
  - Customizable (filters, colors)
  - **Allows step by step tracing**





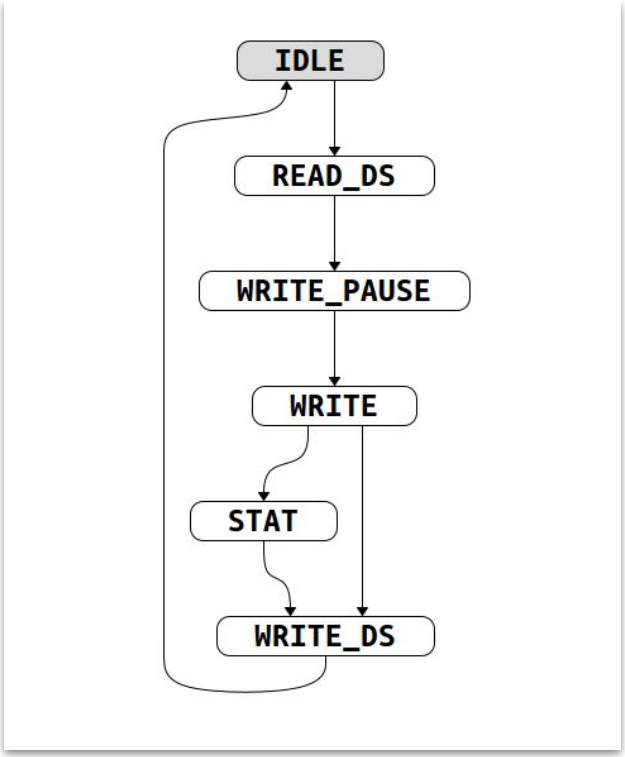
Code Visualization

- Many other types of diagrams:
  - **Wavedrom Timing Diagrams**
  - FSM Diagrams
  - UVM Component Diagrams
  - Bitfield Diagrams



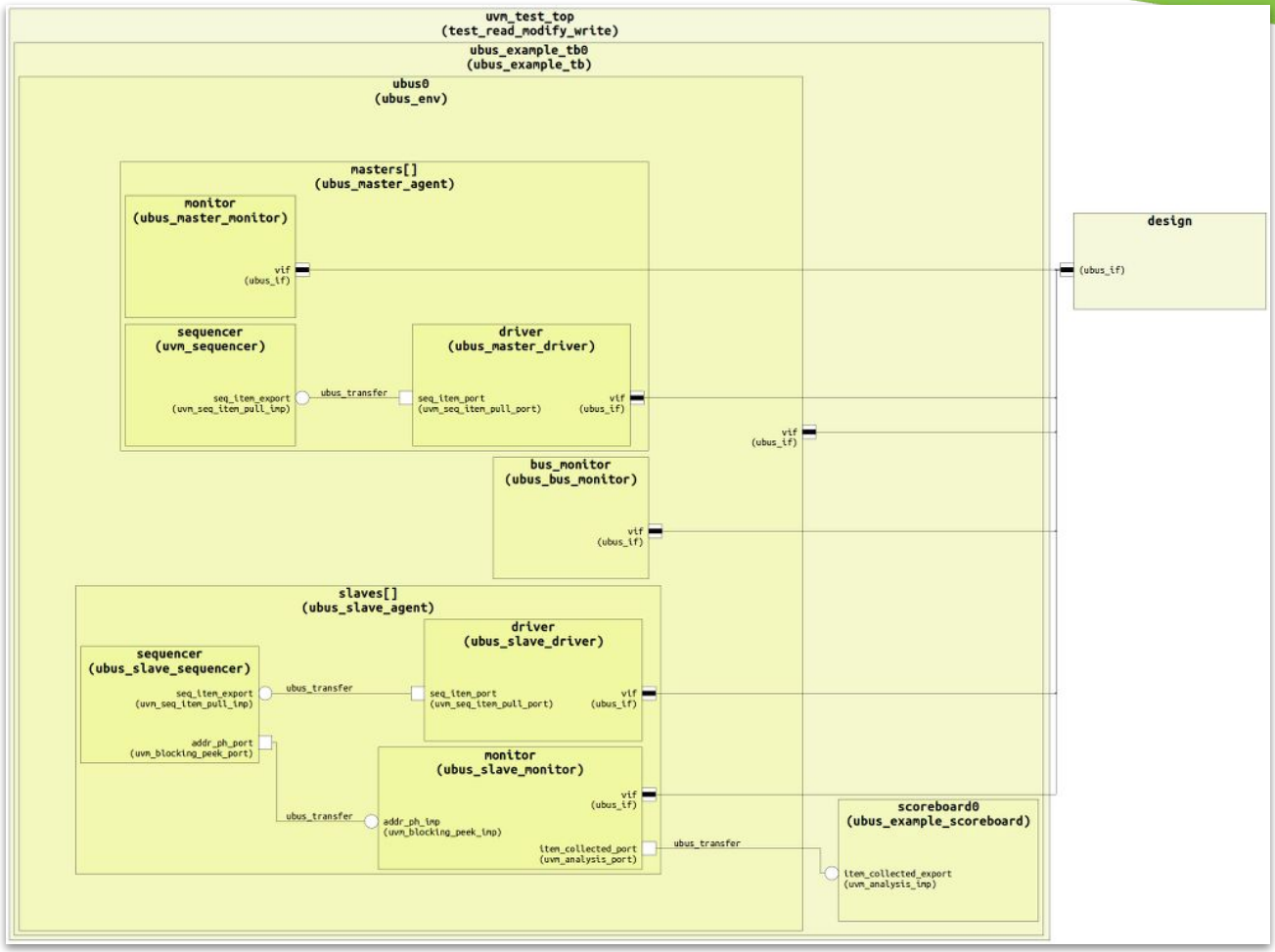
Code Visualization

- Many other types of diagrams:
  - Wavedrom Timing Diagrams
  - **FSM Diagrams**
  - UVM Component Diagrams
  - Bitfield Diagrams



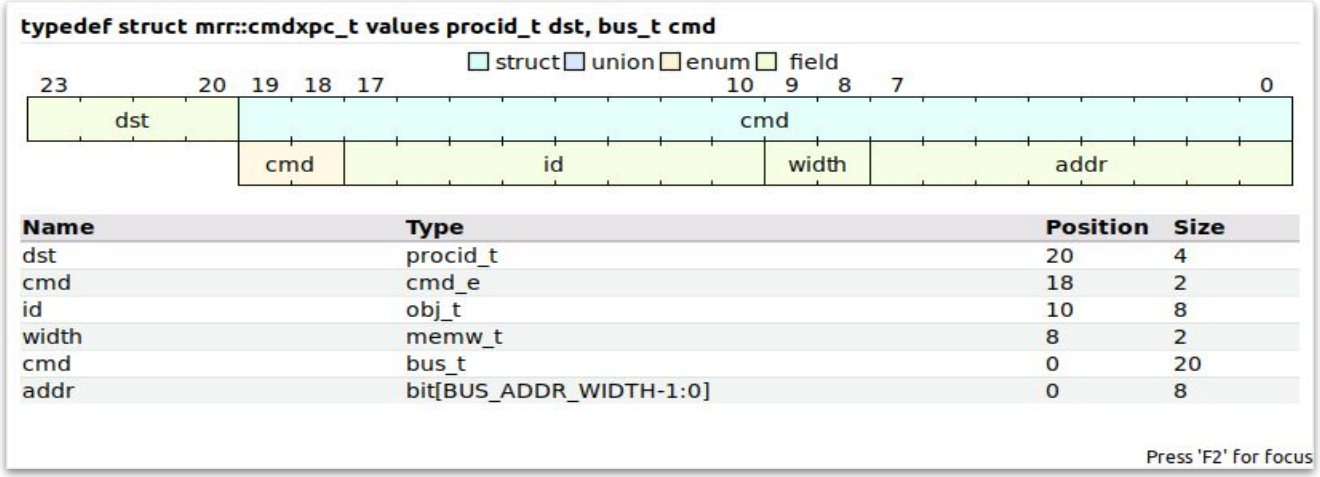
Visualization & exploration

- Many other types of diagrams:
  - Wavedrom Timing Diagrams
  - FSM Diagrams
  - **UVM Component Diagrams**
  - Bitfield Diagrams



Code Visualization

- Many other types of diagrams:
  - Wavedrom Timing Diagrams
  - FSM Diagrams
  - UVM Component Diagrams
  - **Bitfield Diagrams**



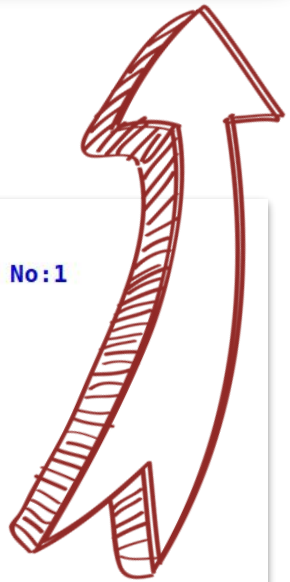
Name	Type	Position	Size
dst	procid_t	20	4
cmd	cmd_e	18	2
id	obj_t	10	8
width	memw_t	8	2
cmd	bus_t	0	20
addr	bit[BUS_ADDR_WIDTH-1:0]	0	8

Press 'F2' for focus

Debugging

- **Smart Log**
  - Jump from sim log to code
  - Messages color-coded by UVM component
  - Errors collected as part of the *Problems view*

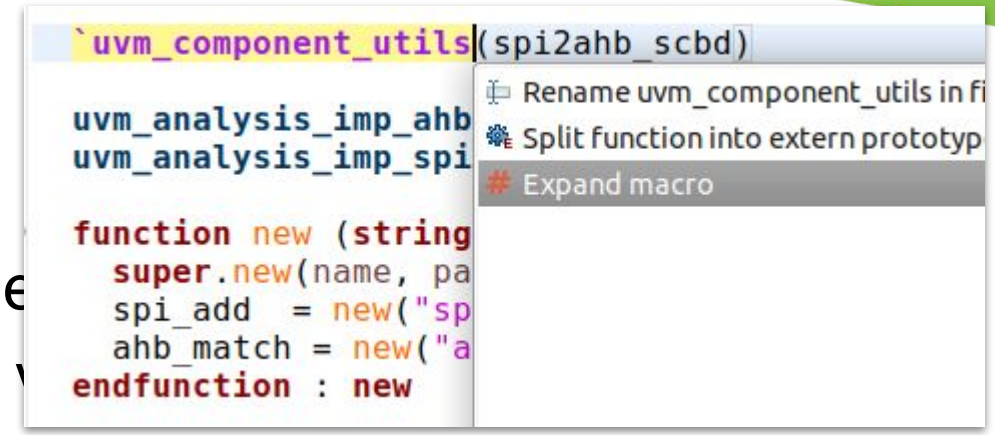
```
192     if ((temp1 & mask) == frame.payload)
193       `uvm_info("SCRBD", $psprintf("####### PASS : %s RECEIVED CORRECT DATA
194     else
195     begin
196       `uvm_error("SCRBD", $psprintf("####### FAIL : %s RECEIVED WRONG DATA"
197       `uvm_info("SCRBD", $psprintf("expected = 'h%0h, received = 'h%0h", ter
198     end
199   endfunction : write_uart
```



```
UVM_INFO interface_uvc_lib/uart/sv/uart_monitor.sv(227) @ 35151000: uvm_test_top.ve.uart0.Rx.monitor [uart_rx_monitor] Received a Stop bit: 'b1
UVM_INFO interface_uvc_lib/uart/sv/uart_monitor.sv(211) @ 35151000: uvm_test_top.ve.uart0.Tx.monitor [uart_tx_monitor] Received a Frame data bit:'b0
UVM_INFO interface_uvc_lib/uart/sv/uart_monitor.sv(236) @ 35950000: uvm_test_top.ve.uart0.Rx.monitor [uart_rx_monitor] Collected the following Frame No:1
----------------------------------------------------------------------------------------------------------------------------------
Name                                Type                          Size  Value
----------------------------------------------------------------------------------------------------------------------------------
uart_frame                          uart_frame                    -     @14998
  start_bit                         integral                      1     'h0
  payload                           integral                      8     'hcf
  parity                            integral                      1     'h1
  stop_bits                         integral                      2     'h1
  error_bits                        integral                      4     'h0
  parity_type                       parity_e                      1     GOOD_PARITY
  transmit_delay                    integral                      32    'd22
  begin_time                        time                          64    18750000
----------------------------------------------------------------------------------------------------------------------------------

UVM_ERROR uart_ctrl/sv/uart_ctrl_scoreboard.sv(196) @ 35950000: uvm_test_top.ve.apb_ss_env.apb_uart0.monitor.rx_scbd [SCRBD] ####### FAIL : uart0 RECEIVED WRONG DATA
UVM_INFO uart_ctrl/sv/uart_ctrl_scoreboard.sv(197) @ 35950000: uvm_test_top.ve.apb_ss_env.apb_uart0.monitor.rx_scbd [SCRBD] expected = 'hcf, received = 'hcf
UVM_INFO interface_uvc_lib/uart/sv/uart_monitor.sv(175) @ 35950000: uvm_test_top.ve.uart0.Rx.monitor [uart_rx_monitor] trying to detect SOP
UVM_INFO interface_uvc_lib/uart/sv/uart_monitor.sv(227) @ 35951000: uvm_test_top.ve.uart1.Rx.monitor [uart_rx_monitor] Received a Stop bit: 'b1
UVM_INFO interface_uvc_lib/uart/sv/uart_monitor.sv(236) @ 36750000: uvm_test_top.ve.uart1.Rx.monitor [uart_rx_monitor] Collected the following Frame No:1
```

Debugging

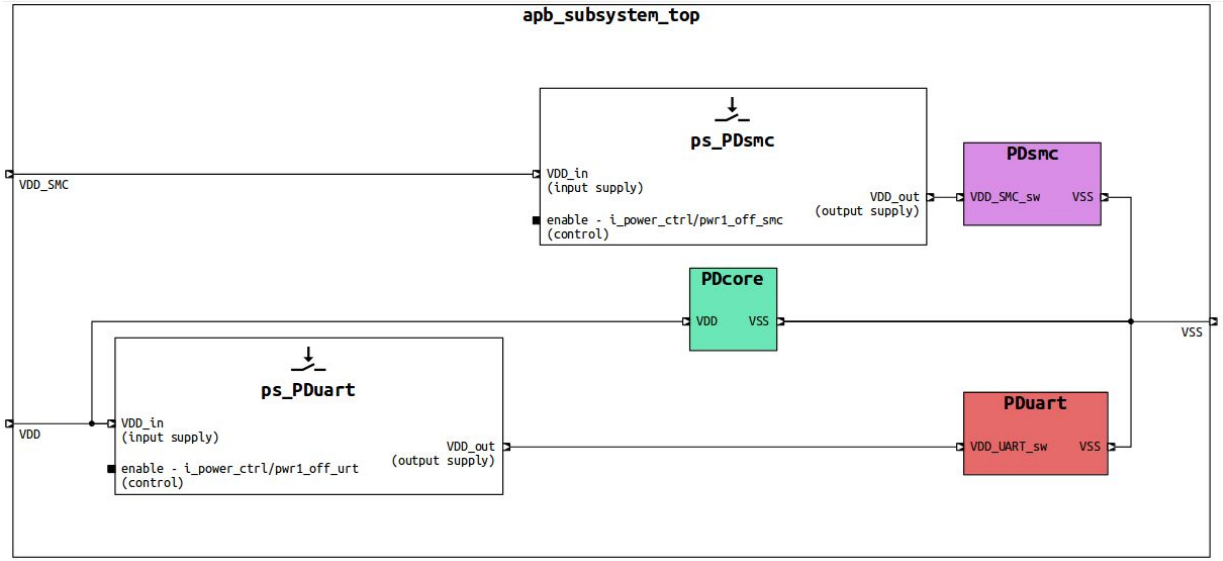
- **Smart Log**
  - Jump from sim log to code
  - Messages color-coded by UVM compone
  - Errors collected as part of the Problems v
- **Macro expansion**
  - Inline expansion & quick collapse
  - Also visible in Inspect View
  - See values in tooltips




```
// @DVT_EXPAND_MACRO_INLINE_START
// `uvm_component_utils(spi2ahb_scbd)
// @DVT_EXPAND_MACRO_INLINE_ORIGINAL

   static bit m_registered_converter__ = m_uvm_resource_sprint_converter#(spi2ahb_scbd)::register();

   typedef uvm_component_registry #(spi2ahb_scbd,"spi2ahb_scbd") type_id;
   static function type_id get_type();
     return type_id::get();
   endfunction
   virtual function uvm_object_wrapper get_object_type();
     return type_id::get();
   endfunction

   const static string type_name = "spi2ahb_scbd";
   virtual function string get_type_name ();
     return type_name;
   endfunction

// @DVT_EXPAND_MACRO_INLINE_END
```

Low power format (UPF)

- **Power Domain** View shows
  - Design instances
  - Isolation strategies
  - Retention rules
- Power domains also shown in
  - Design Hierarchy View
  - Schematic Diagrams
  - Breadcrumb Navigation Bar
- **Supply Network Diagram**



The Evolution Continues

DVT + VS Code + AI

DVT in the VS Code Era

Modern Development Environment

- **The Power of Integration**
  - DVT now available as a ***VS Code plugin***
  - Industry-standard development environment
  - Complete hardware design & verification capabilities in a modern IDE

- **VS-Code One Environment, Many Tools**

git

[redacted]

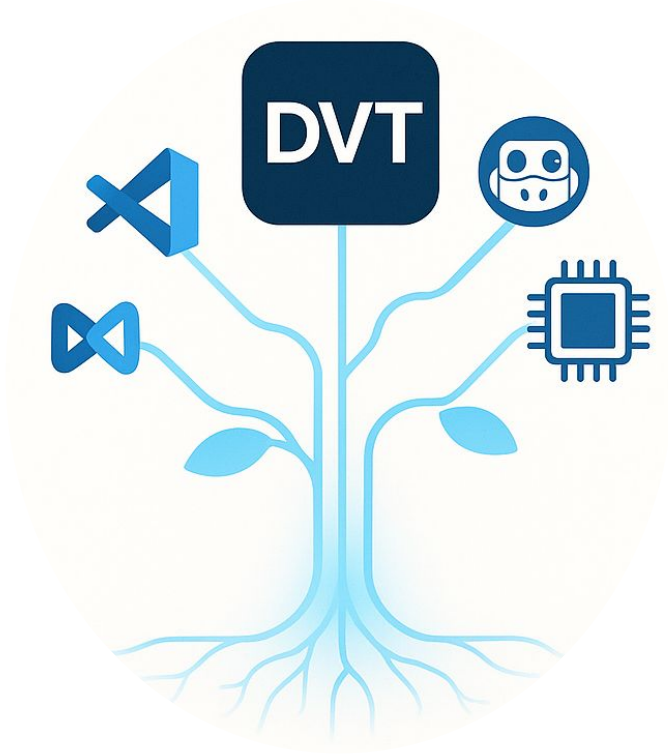
python™

ATLASSIAN

GitHub Copilot

AMIQ's Built-In AI Assistant

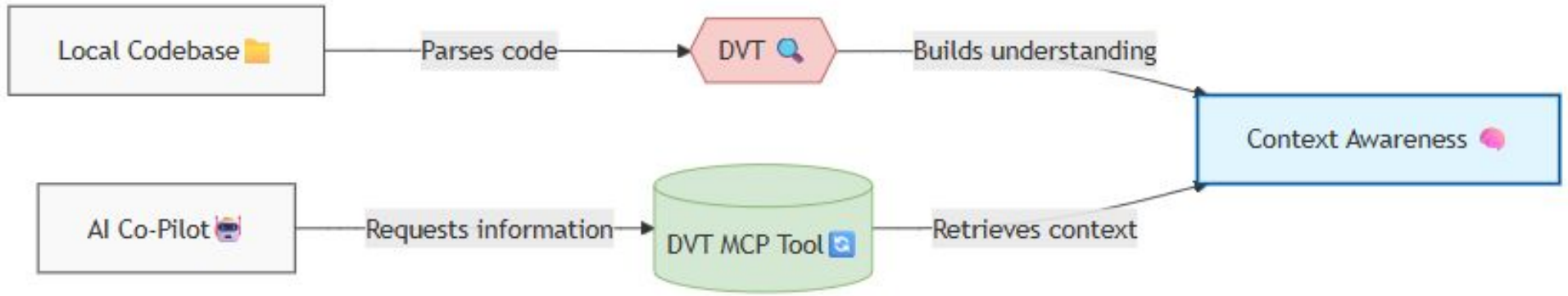
- **Smart Assistance for Hardware Design**
  - AMIQ's **built-in AI assistant** for DVT users
  - Hardware-aware code suggestions and completion
  - Specialized for HDL and verification languages
  - Connects to your **company's LLM** infrastructure



"AI assistance tailored specifically for hardware engineers"

MCP - Making AI Understand Your Code

- **Bringing AI Intelligence to Your local codebase**
  - Upcoming MCP tools connect your codebase with AI assistants
  - Your AI assistant becomes familiar with your specific codebase
  - Works alongside external AI Copilots for advanced assistance
  - Maintains privacy while enabling powerful completions



Bridges the gap between generic AI and hardware-specific knowledge

Thank you / Q&A

Contact me for further details
Email: miller2812@gmail.com
Phone: 054-6442812
LinkedIn: Netanel Miller