



DVT IDE

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

Netanel Miller

AI & Verification Innovation Lead, Texas Instruments

My Journey



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

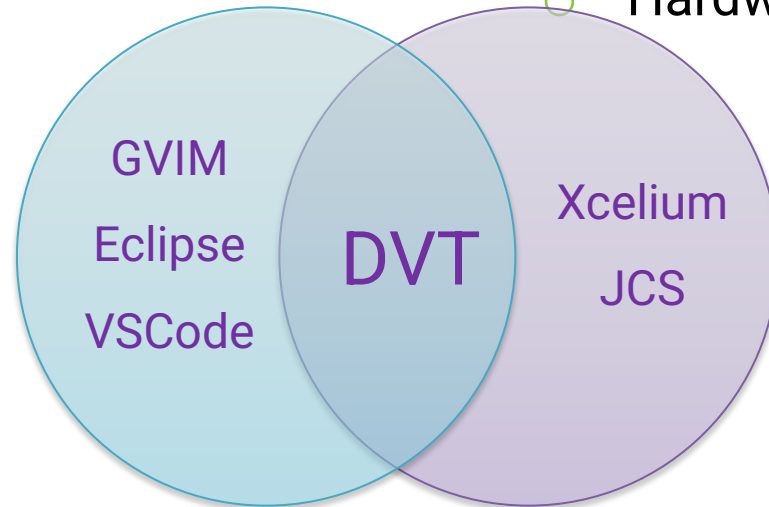


Text Editor / Base IDE

- 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"



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

"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);
```

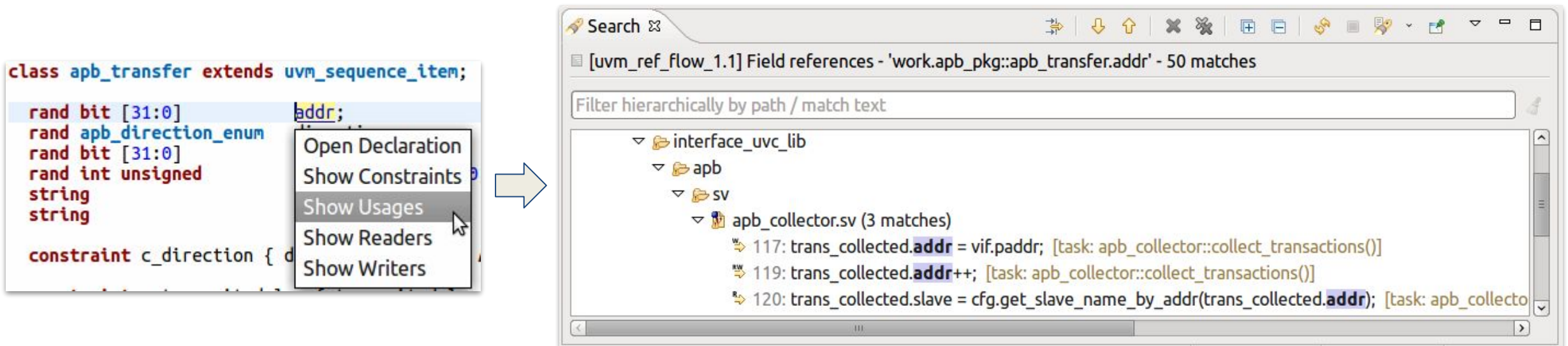
Open Declaration
Open Field Type
Show Usages



```
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



```
class apb_transfer extends uvm_sequence_item;
  rand bit [31:0] addr;
  rand apb_direction_enum
  rand bit [31:0]
  rand int unsigned
  string
  string

  constraint c_direction { d
```

Search [x]

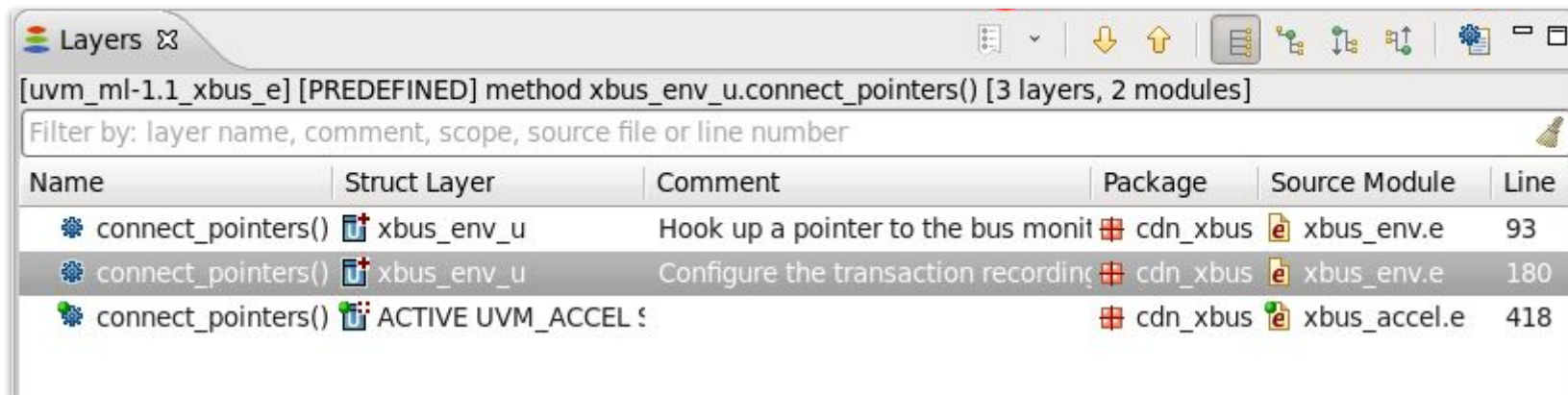
[uvm_ref_flow_1.1] Field references - 'work.apb_pkg::apb_transfer.addr' - 50 matches

Filter hierarchically by path / match text

- interface_uvc_lib
 - apb
 - sv
 - apb_collector.sv (3 matches)
 - 117: trans_collected.addr = vif.paddr; [task: apb_collector::collect_transactions()]
 - 119: trans_collected.addr++; [task: apb_collector::collect_transactions()]
 - 120: trans_collected.slave = cfg.get_slave_name_by_addr(trans_collected.addr); [task: apb_collector::collect_transactions()]

> 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 ;)



The screenshot shows a 'Layers' window with a search bar and a table of results. The search bar contains the text 'Filter by: layer name, comment, scope, source file or line number'. The table has columns for Name, Struct Layer, Comment, Package, Source Module, and Line. The results are for the method 'connect_pointers()' in the 'cdn_xbus' package, showing three instances in different source modules: 'xbus_env.e' (line 93), 'xbus_env.e' (line 180), and 'xbus_accel.e' (line 418).

Name	Struct Layer	Comment	Package	Source Module	Line
connect_pointers()	xbus_env_u	Hook up a pointer to the bus monit	cdn_xbus	xbus_env.e	93
connect_pointers()	xbus_env_u	Configure the transaction recording	cdn_xbus	xbus_env.e	180
connect_pointers()	ACTIVE UVM_ACCEL :		cdn_xbus	xbus_accel.e	418

> 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>

The screenshot shows a code editor with two tabs: `xbus_signal_map_h.e` and `*xbus_accel.e`. The active tab is `xbus_signal_map_h.e`, which contains the following code snippet:

```
586 connect_ports() is also {
587     sig_start.disconnect();
588     do_bind (sig_start, empty);
589     sig_addr.disconnect();
590     do_bind (sig_addr, empty);
591     sig_addr.di
592     do_bind (si
593     sig_size.di
594     do_bind (si
595     sig_read.di
596     do_bind (si
597     sig_write.d
598     do_bind (si
```

An auto-completion popup is visible over the code, showing the following options:

- Show AI Assistant Code Completions ...
- disconnect()**
- disconnect_bound_set()
- vhdl_disconnect_value() : list of mvl

On the right side of the editor, a tooltip for the `disconnect()` method is displayed:

method any_port.disconnect()

PREDEFINED

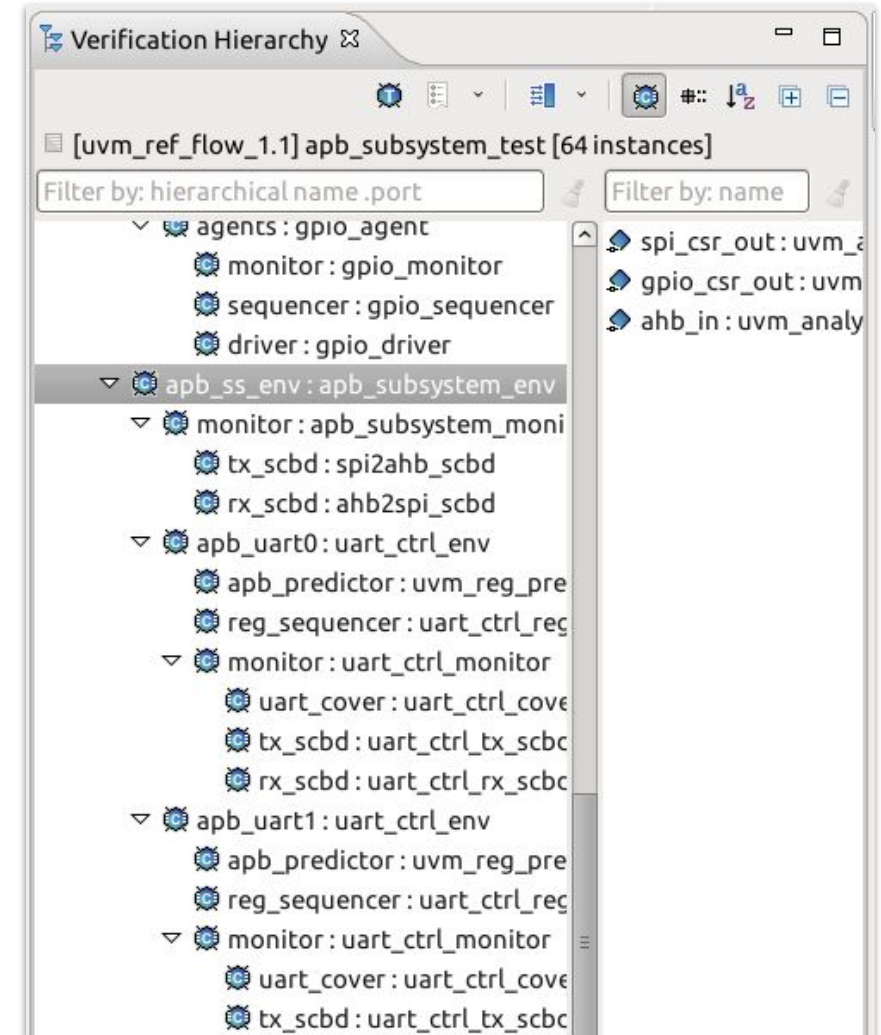
Disconnect a port from its bound set.

Calling this pseudo-method of a port disconnects this port from the bound set it belongs to, and returns it to its initial, dangling state. The rest of the bound set remains intact.

> 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

The screenshot shows a software window titled "Config DB" with a search bar and a table of configuration data. The table has columns for Call, Type, Context, Instance Name, Field Name, and Value. A section titled "Setters and Getters" is expanded, showing a list of calls. The first call is a "set" operation on the "uart_if" field of the "apb_subsystem_test.ve.uart0*" instance. Subsequent calls are "get" operations on the same instance, with values ranging from 761 to 767. The table also shows calls for the "uart1*" instance, with values ranging from 765 to 767.

Call	Type	Context	Instance Name	Field Name	Value
set [1]	uart_if	null	apb_subsystem_test.ve.uart0*	vif	apb_subsystem_top.uart_s0
get [761]	uart_if	apb_subsystem_test.ve.uart0.Rx.monitor		vif	apb_subsystem_top.uart_s0
get [762]	uart_if	apb_subsystem_test.ve.uart0.Tx.driver		vif	apb_subsystem_top.uart_s0
get [763]	uart_if	apb_subsystem_test.ve.uart0.Tx.monitor		vif	apb_subsystem_top.uart_s0
get [764]	uart_if	apb_subsystem_test.ve.uart0		vif	apb_subsystem_top.uart_s0
set [2]	uart_if	null	apb_subsystem_test.ve.uart1*	vif	apb_subsystem_top.uart_s1
get [765]	uart_if	apb_subsystem_test.ve.uart1.Rx.monitor		vif	apb_subsystem_top.uart_s1
get [766]	uart_if	apb_subsystem_test.ve.uart1.Tx.driver		vif	apb_subsystem_top.uart_s1
get [767]	uart_if	apb_subsystem_test.ve.uart1.Tx.monitor		vif	apb_subsystem_top.uart_s1

> 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

Applicable On	Original Type	Override Type	Override Path	Override Chain
apb_subsystem_lp_test.*	ahb_intermediate_monitor	ahb_subsequent_monitor	...test.ve.ahb0.master_agent.monitors[1]	
apb_subsystem_lp_test.*	ahb_master_monitor	ahb_intermediate_monitor	...test.ve.ahb0.master_agent.monitors[0]	
apb_subsystem_lp_test.*	ahb_master_monitor	ahb_intermediate_monitor	...test.ve.ahb0.master_agent.monitors[1]	
...master_agent.monitors[0]	ahb_subsequent_monitor	ahb_monitor	...test.ve.ahb0.master_agent.monitors[0]	
Chain				
Override Chain	ahb_master_monitor	ahb_monitor	...test.ve.ahb0.master_agent.monitors[0]	...tor > ahb_intermediate_monitor > ahb_subsequent_monitor > ahb_monitor
Override Chain	ahb_master_monitor	ahb_subsequent_monitor	...test.ve.ahb0.master_agent.monitors[1]	...hb_master_monitor > ahb_intermediate_monitor > ahb_subsequent_monitor
Unused				
apb_subsystem_lp_test.*	ahb_master_monitor	ahb_monitor		
...lp_test.ve.apb_ss_env.cfg	apb_subsystem_config	...ault_apb_subsystem_config		

> 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

The screenshot displays the 'Registers' tool window. The left pane shows a hierarchical tree of registers under the path '[uvm_ref_flow_1.1] apb_gpio_simple_test [UVM Elaboration Model] [20 registers]'. The 'ua_fifo_ctrl' register is selected. The right pane shows the 'Bit Fields of UVM register ua_fifo_ctrl_c' with a bitfield diagram and a table of field details.

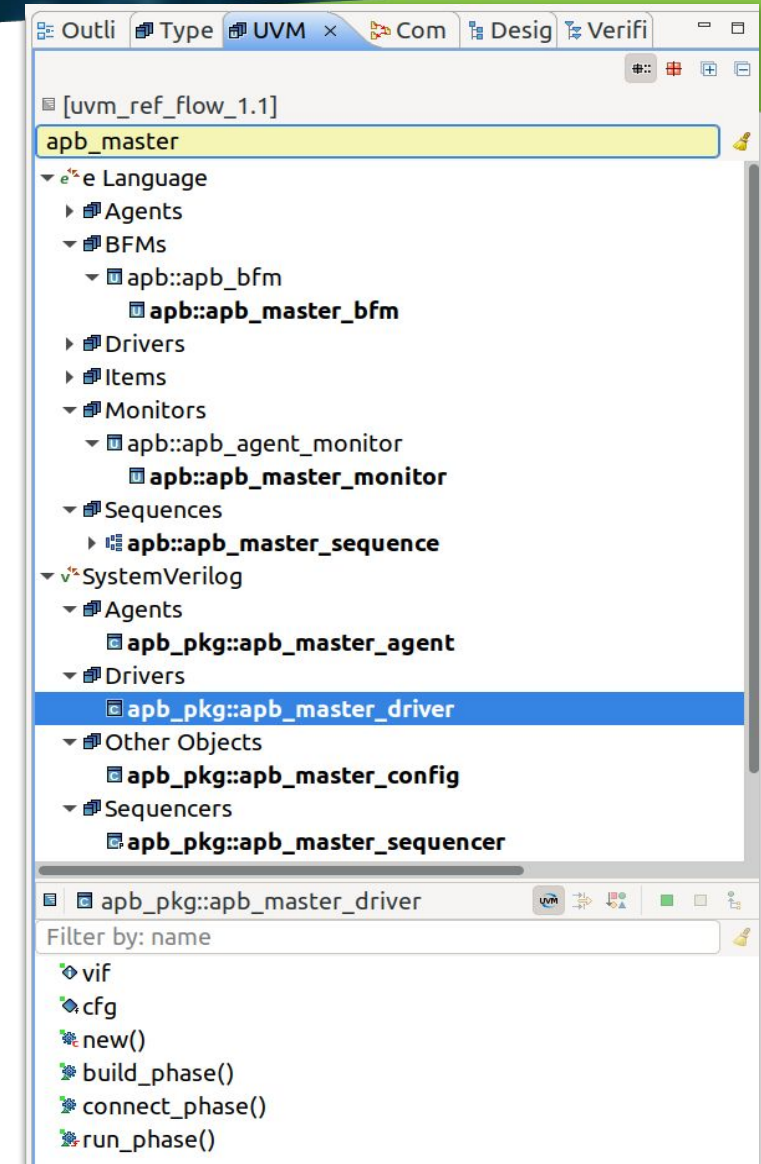
Field	Size	Position	Access	Volatility	Reset	Has Reset	Randomized	Accessible
rx_clear	1	1	WO	0	96	1	1	1
tx_clear	1	2	WO	0	48	1	1	1
rx_fifo_int_trig_level	2	6	WO	0	3	1	1	1

> 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

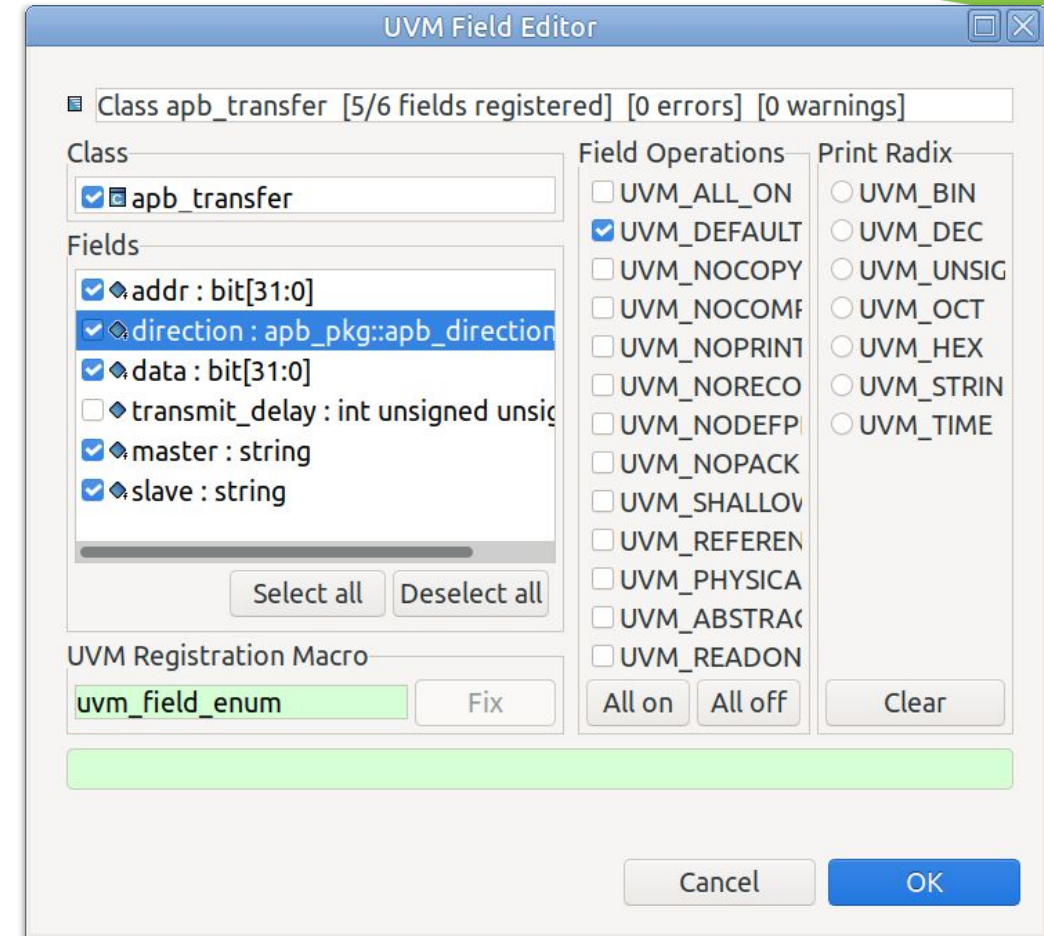


- **UVM Browser**

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

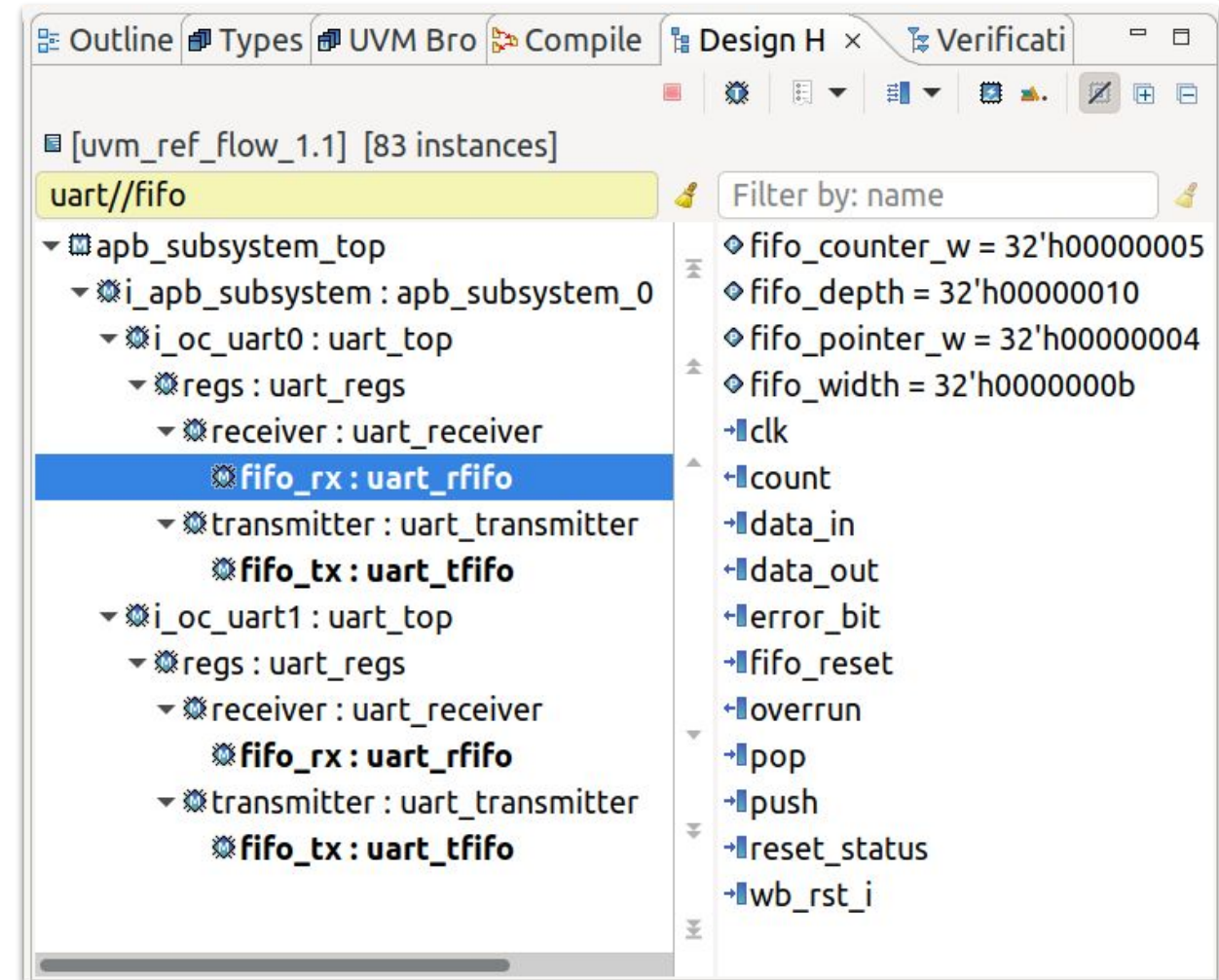
- **UVM Field Editor**

- 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

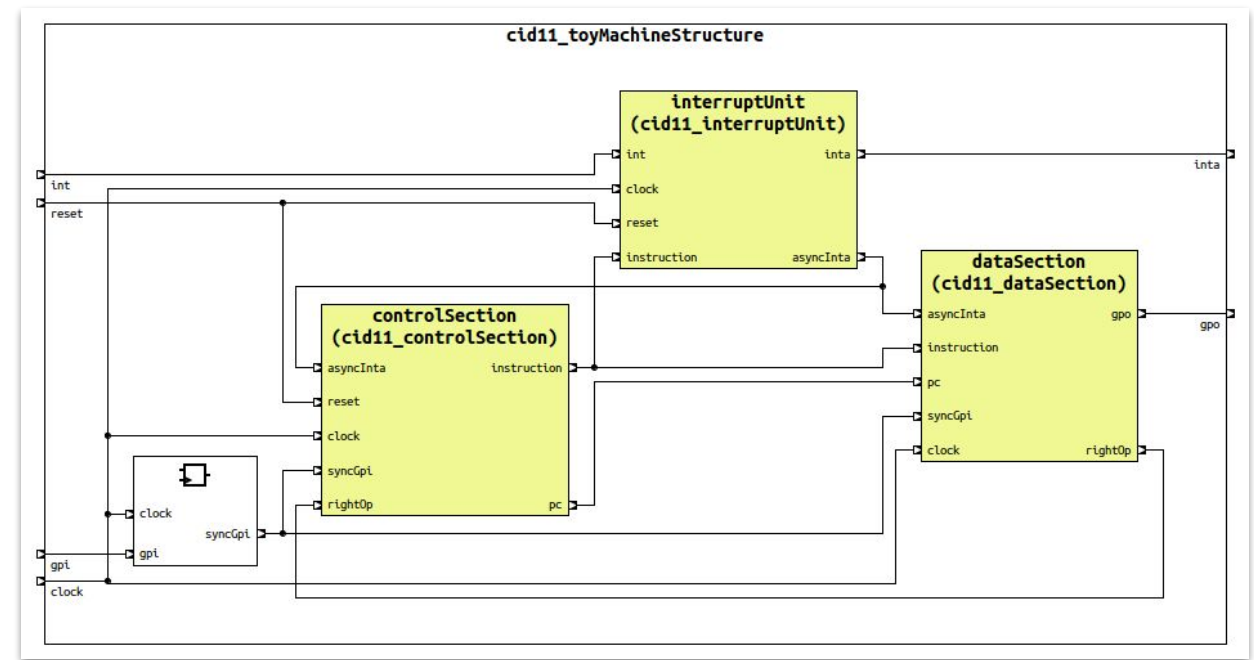


- **Schematic Diagrams**

- 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**

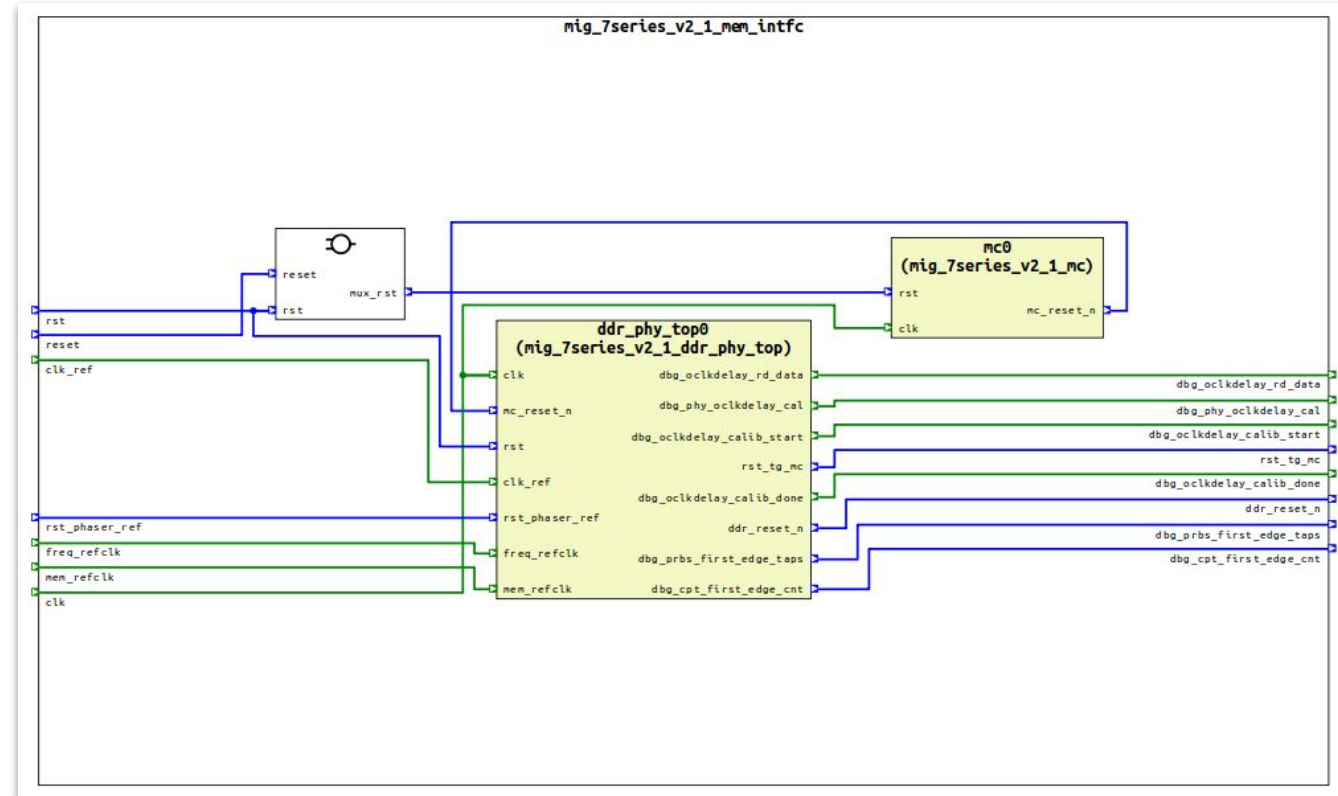
- Customizable (filters, colors)
- Allows step by step tracing



> 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



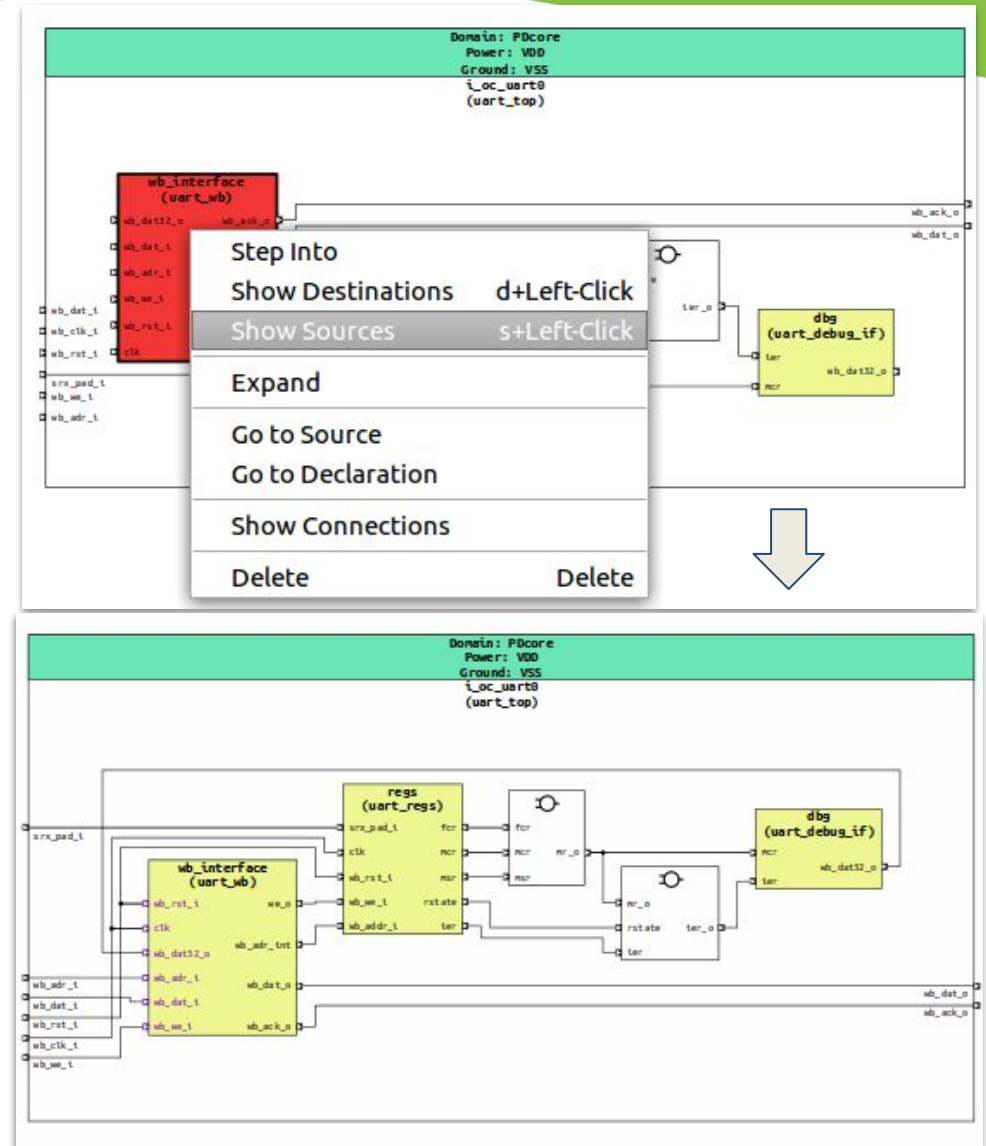
> Visualization & exploration

- **Design Hierarchy View**

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

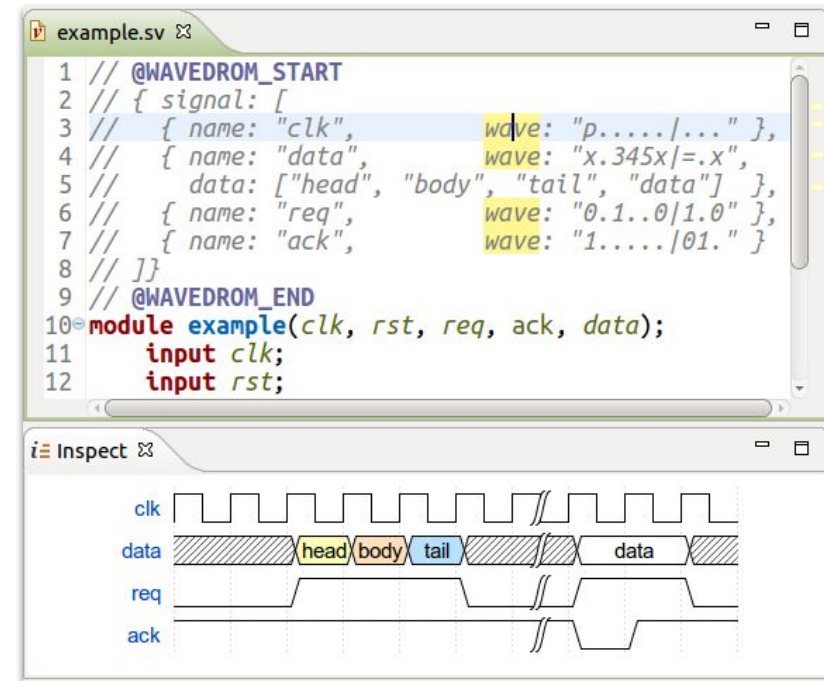
- **Schematic Diagrams**

- Show ports, sub-instances, logic blocks and connections
- Customizable (filters, colors)
- **Allows step by step tracing**



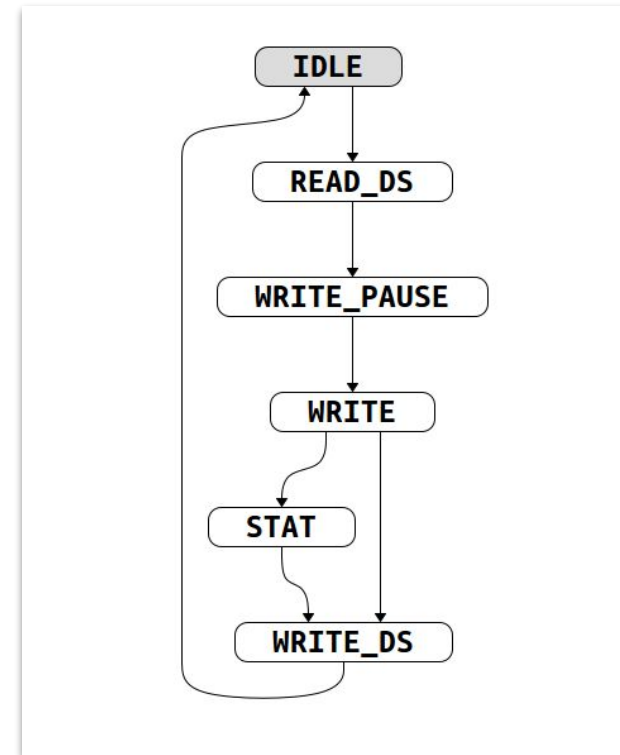
> Code Visualization

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



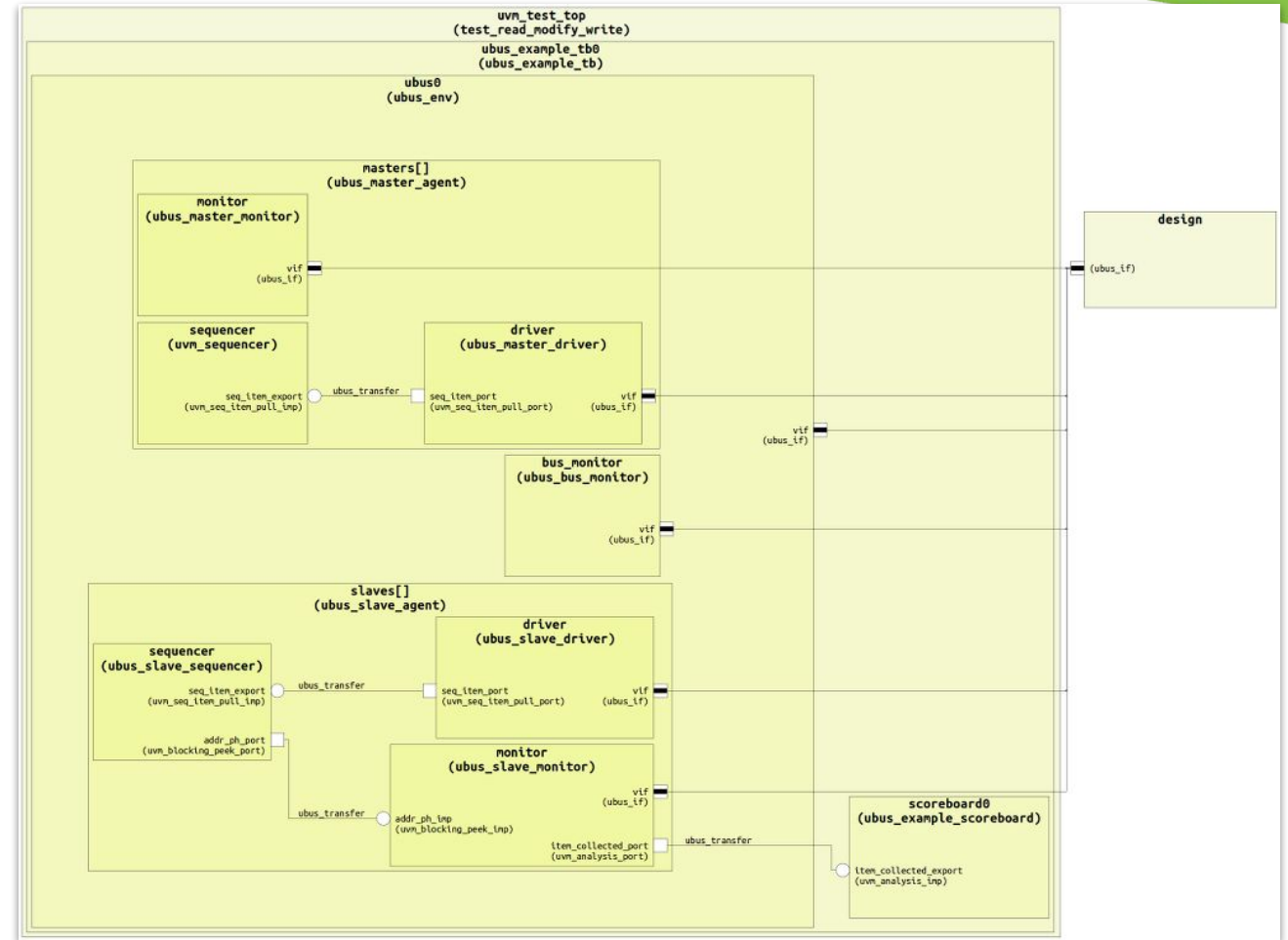
> Code Visualization

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



➤ Visualization & exploration

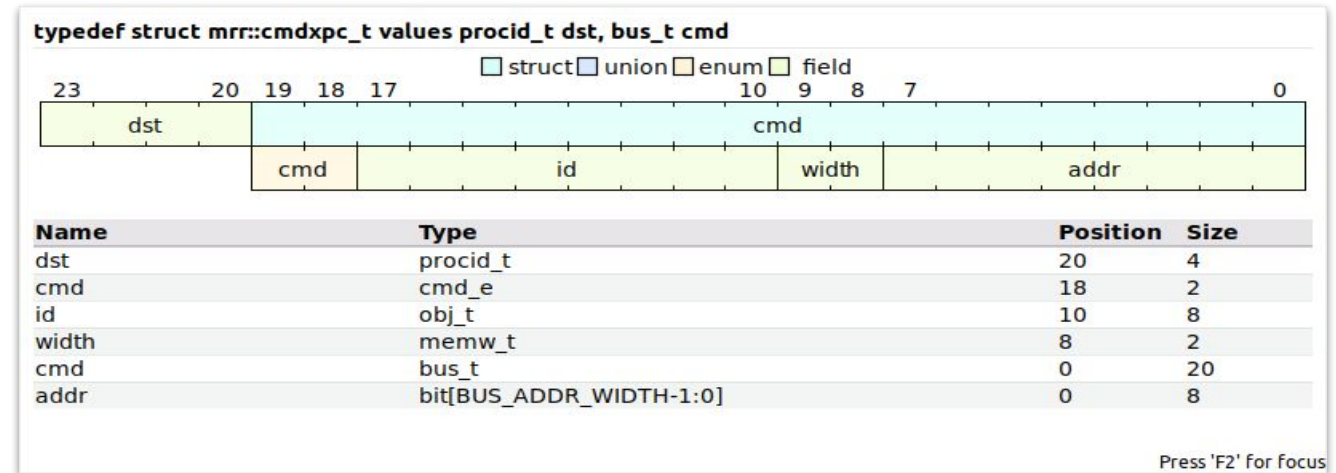
- Many other types of diagrams:
 - Wavedrom Timing Diagrams
 - FSM Diagrams
 - **UVM Component Diagrams**
 - Bitfield Diagrams



> Code Visualization



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



> Debugging



- **Smart Log**

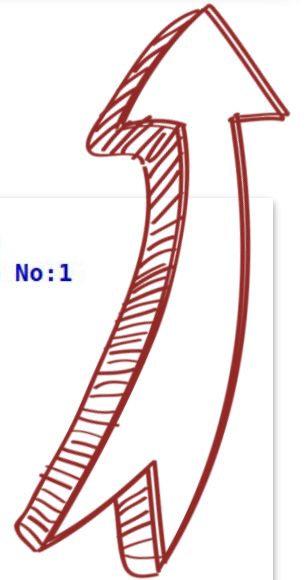
- Jump from sim log to code
- Messages color-coded by UVM component
- Errors collected as part of the *Problems view*

```
192     if ((temp1 & mask) == frame.payload)
193         `uvm_info("SCRBD", $sprintf("##### PASS : %s RECEIVED CORRECT DATA", temp1), UVM_INFO)
194     else
195     begin
196         `uvm_error("SCRBD", $sprintf("##### FAIL : %s RECEIVED WRONG DATA", temp1), UVM_ERROR)
197         `uvm_info("SCRBD", $sprintf("expected = 'h%0h, received = 'h%0h", temp1, temp2), UVM_INFO)
198     end
199 endfunction : write_uart
```

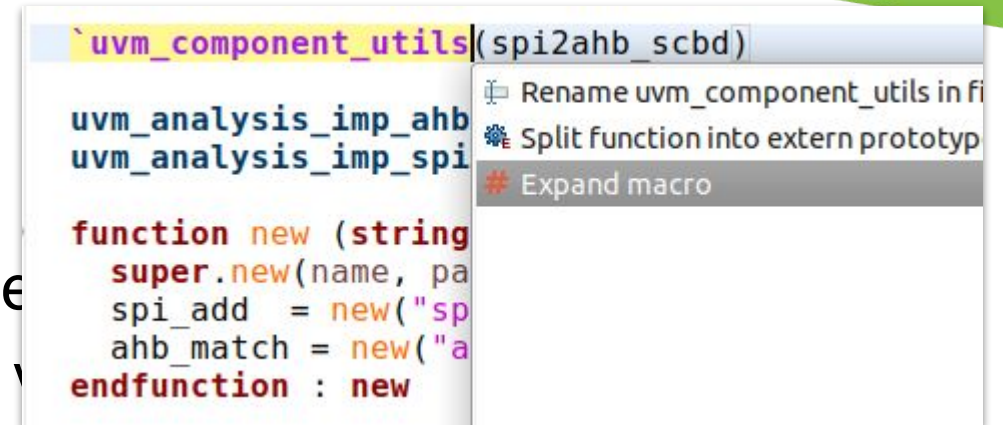
```
UVM_INFO interface_uvc_lib/uart/sv/uart_monitor.sv(227) @ 35151000: uvm_test_top.ve.uart0.Rx.monitor [uart_rx_monitor] Received a Stop bit: 'b1
UVM_INFO interface_uvc_lib/uart/sv/uart_monitor.sv(211) @ 35151000: uvm_test_top.ve.uart0.Tx.monitor [uart_tx_monitor] Received a Frame data bit:'b0
UVM_INFO 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
```



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



```
`uvm_component_utils(spi2ahb_scbd)

uvm_analysis_imp_ahb
uvm_analysis_imp_spi

function new (string
    super.new(name, pa
    spi_add = new("sp
    ahb_match = new("a
endfunction : new
```



```
// @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();

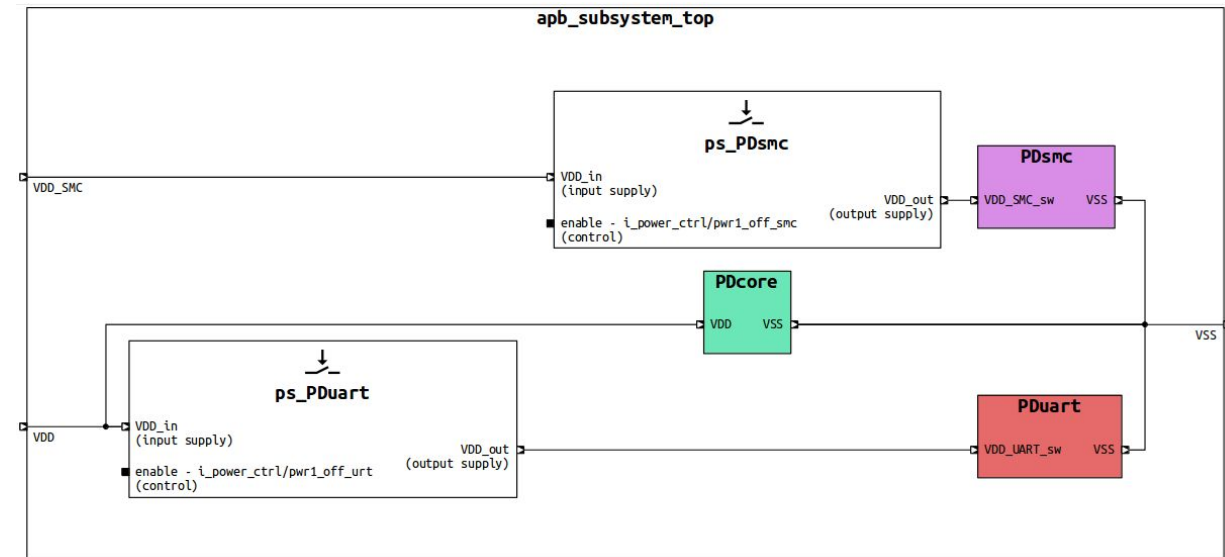
typedef uvm_component_registry #(spi2ahb_scbd, "spi2ahb_scbd") type_id;
static function type_id get_type();
    return type_id::get();
endfunction
virtual function uvm_object_wrapper get_object_type();
    return type_id::get();
endfunction

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**





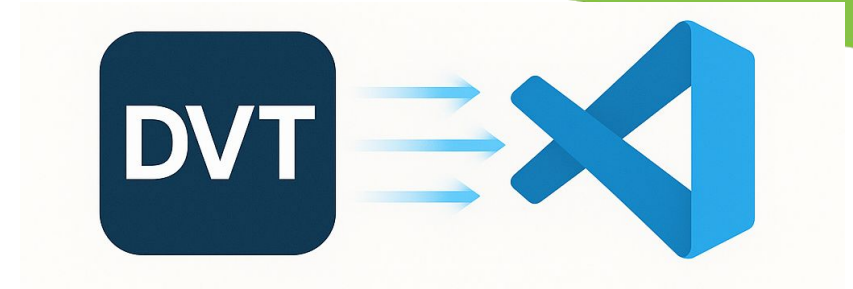
The Evolution Continues

DVT + VS Code + AI

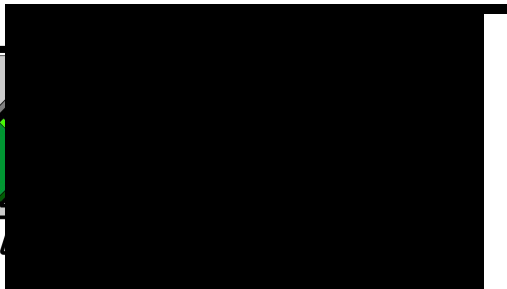
> DVT in the VS Code Era



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**

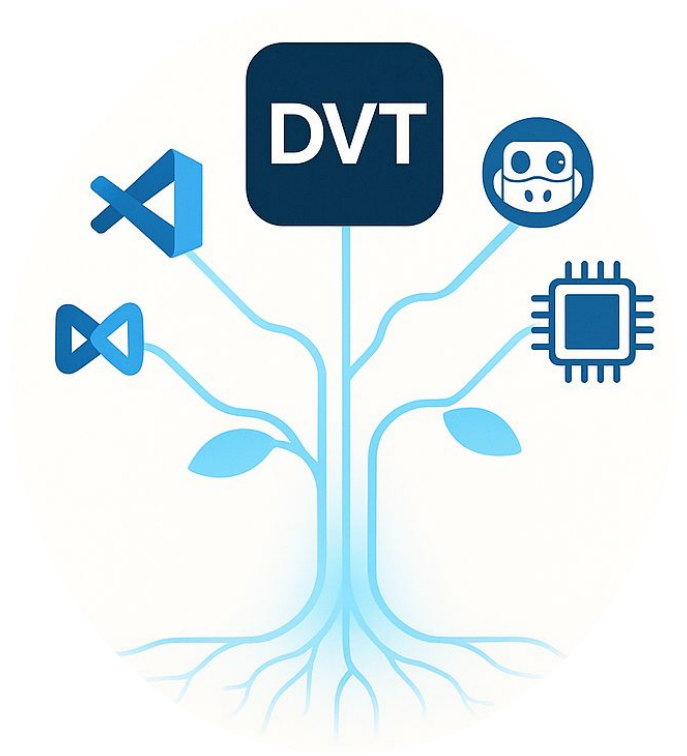


GitHub
Copilot

> 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

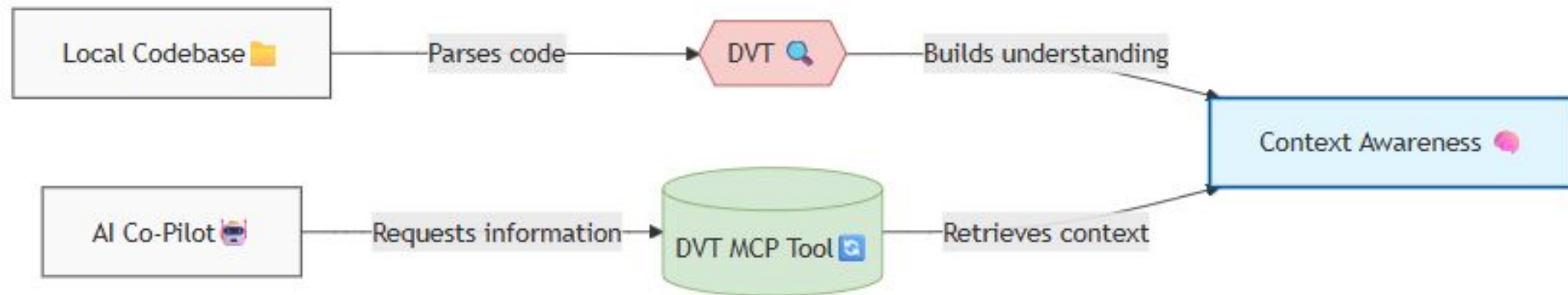


"AI assistance tailored specifically for hardware engineers"

> 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!

Contact me for further details

Email: miller2812@gmail.com

Phone: 054-6442812

LinkedIn: [Netanel Miller](#)