2025
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
EUROPE
MUNICH, GERMANY
OCTOBER 14-15, 2025

AI Pair or Despair Programming
Using Aider to build a VIP with UVM-SV and PyUVM

André Winkelmann, Verilab

Damir Ahmetovic Ignjic, Verilab

verilab

accellera
SYSTEMS INITIATIVE

AI Reshaping Industries

- Coding assistants becoming part of software development
  - E.g. Github Copilot, Claude Code, Aider, Cursor, …
- Use cases
  - Simple code completions
  - Add new functionality
  - Refactor code
  - Complete new files
  - Entire projects from scratch
- What about DV?

Motivation

- LLMs are great at generating Python code
- UVM-SV is the defacto standard for DV

Investigated questions

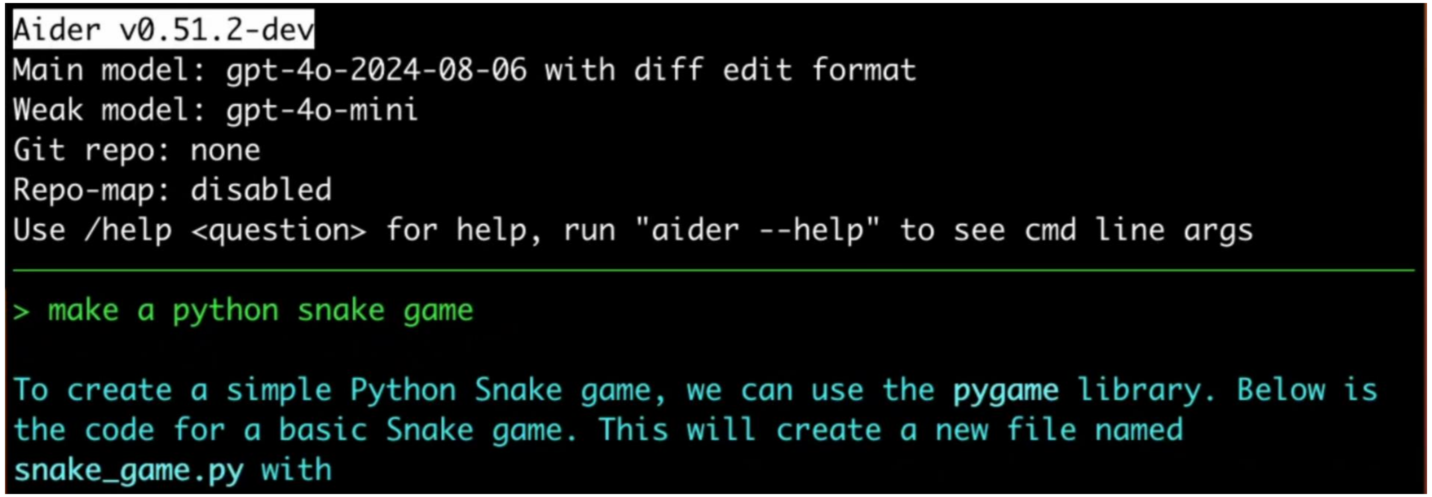
1. How effective is generative AI at producing cocotb+PyUVM code compared to UVM-SV?
2. Is generative AI mature enough to assist DV engineers in creating complex VIP, or is its utility limited to simpler code editing tasks?

Experimental Dimensions

- LLM Provider
  - Google: gemini-2.5-pro-preview-06-05
  - Anthropic: claude-opus-4-20250514
  - OpenAI: o3
- Verification Methodology
  - SystemVerilog using UVM
  - Python using cocotb + PyUVM
- Coding Conventions
  - Without additional coding conventions
  - With additional coding conventions

Aider

- *“AI pair programming in your terminal”*
- Selected Aider for the paper as it allows scripting its use
- Systematically test all model and configuration permutations



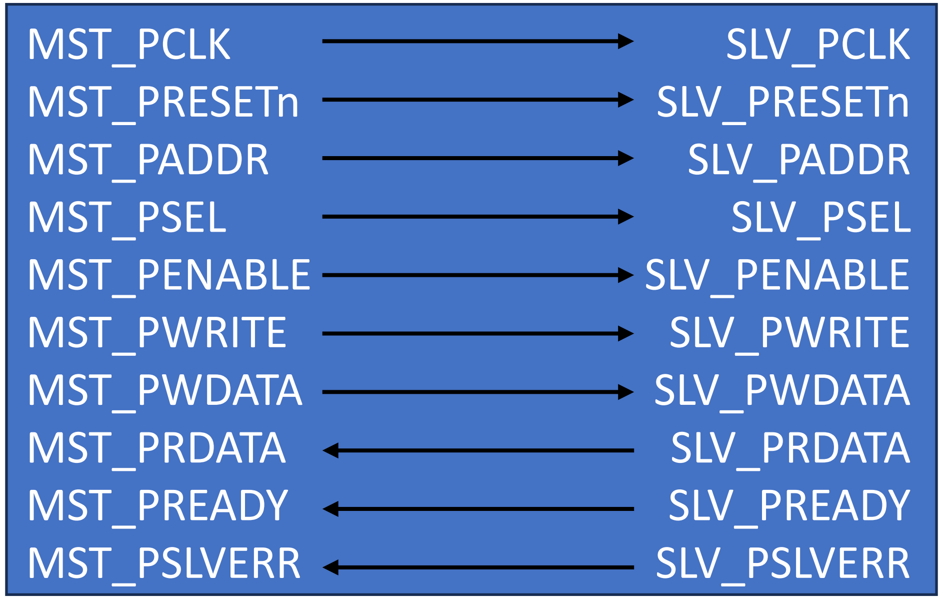
```
Aider v0.51.2-dev
Main model: gpt-4o-2024-08-06 with diff edit format
Weak model: gpt-4o-mini
Git repo: none
Repo-map: disabled
Use /help <question> for help, run "aider --help" to see cmd line args

> make a python snake game

To create a simple Python Snake game, we can use the pygame library. Below is
the code for a basic Snake game. This will create a new file named
snake_game.py with
```

VIP Generation Prompts

- **UVM-SV:** Create under vips/apb directory a production-quality UVM-1.2 SystemVerilog VIP for the AMBA APB3 protocol. The VIP should include all standard UVM components plus a comprehensive sequence library, a functional coverage model and protocol checks implementing the entire APB3 specification.
- **PyUVM:** Create under vips/apb directory a production quality PyUVM VIP for the AMBA APB3 protocol. The VIP should include all standard UVM components plus a comprehensive sequence library, a functional coverage model and protocol checks implementing the entire APB3 specification.

TB Generation

- Also asked to generate a TB using the APB VIP
- Simple APB passthrough DUT given (not generated)



MST_PCLK → SLV_PCLK

MST_PRESETn → SLV_PRESETn

MST_PADDR → SLV_PADDR

MST_PSEL → SLV_PSEL

MST_PENABLE → SLV_PENABLE

MST_PWRITE → SLV_PWRITE

MST_PWDATA → SLV_PWDATA

MST_PRDATA ← SLV_PRDATA

MST_PREADY ← SLV_PREADY

MST_PSLVERR ← SLV_PSLVERR

Result Evaluation

- Review of the generated VIP code
- Syntactic correctness & elaboration
- Iteration count VIP
- Iteration count TB+VIP
- Functional correctness
- LLM Costs

Review of the generated VIP code

- UVM-SV with and without conventions
  - Surprise in differences of generated code across LLMs
  - Conventions improve structural quality
  - Risk of loss of functionality with conventions
  - Protocol versions are a common pitfall (APB3 vs. APB4)
  - Coverage quality varies greatly
  - Reset and timing are weak points
  - Sequence libraries show promise

Code Convention Influence (1)

- Convention: *Use the covergroup sample() method to collect coverage*

Without convention	With convention
covergroup apb_cg;	**covergroup** apb_cg **with function** sample(apb_transaction trans);

Code Convention Influence (2)

- Convention: *Use prefix_ and _postfix to delineate name types*

Without convention

```
virtual apb_if vif;
apb_config cfg;
```

With convention

```
virtual apb_if m_vif;
apb_config m_config;
```

Code Convention Influence (3)

- Convention: *Use a begin-end pair to bracket conditional statements*

Without convention

```
if (!...::get(this, "", "cfg", cfg))
  `uvm_fatal("NOCFG", "…")
```

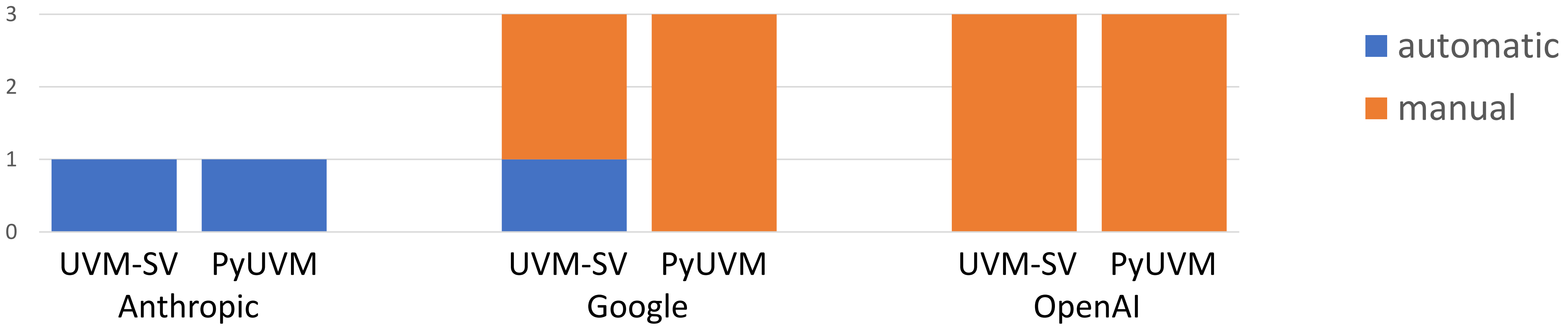
With convention

```
if (!...::get(this, "", "cfg", m_cfg)) begin
  `uvm_fatal("NOCFG", "…")
end
```

Syntactic correctness

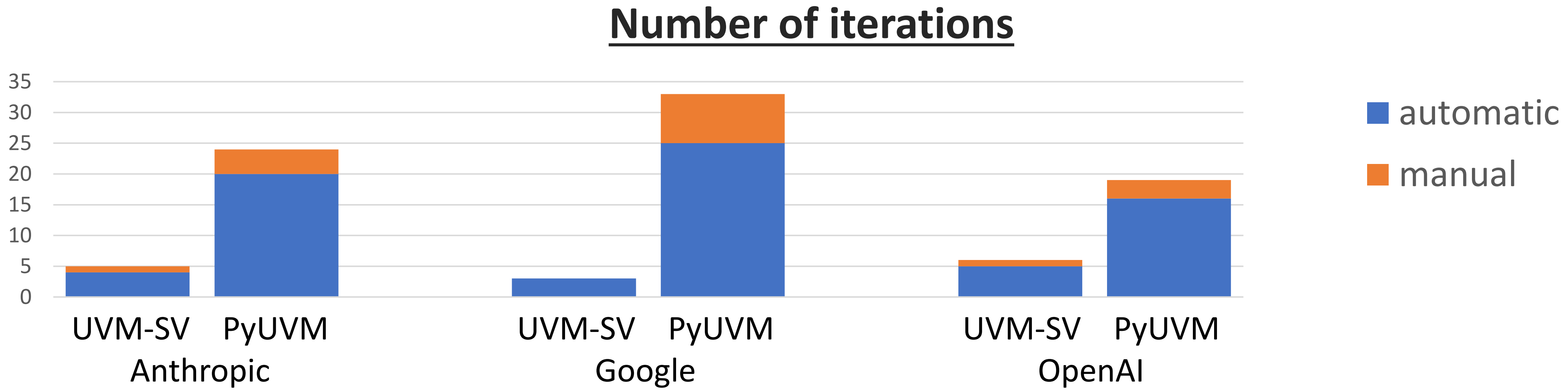
- Tested SV-UVM static compile, elaboration
- Mypy code analyzer for PyUVM
- No LLM first time right

Number of iterations



Iterations for first Simulation

- UVM-SV similar low effort for first simulation
- PyUVM needed lots and lots of iterations with mixed results
- Google run, Anthropic aborted, OpenAI very slow to respond



Observations Reaching Simulation Readiness

- Multiple runs of the same bug fix → different results
  - Like throwing a dice
- All LLMs work like trial and error
  - Thinking output: „Let’s try xyz and see if that fixes the issue.“
- Issues working with PyUVM
  - LLMs seem to guess what a Pythonic syntax might be compared to UVM-SV
  - Had a lot of trouble with factory and config database
  - Enum usage like UVM_ACTIVE
  - LLM attempts to use try … except blocks to fix coding issues

Waveform Analysis

	UVM-SV	PyUVM
Google	- PREADY, PSLVERR undriven - Not waiting on reset	+ No signals X or Z - Only read transactions
Anthropic	+ No signals X or Z + Written matches read data o Memory model	--- No waves analyzed --- Aborted compile/elab
OpenAI	+ No signals X or Z o Memory model - Not waiting on reset	--- All signals Z --- Missing DUT hookup code

Cost Analysis - VIP generation

- Cost to generate VIP only

	Anthropic	Google	OpenAI
UVM-SV no conv	0.88	**0.08**	0.10
UVM-SV with conv	0.84	0.13	**0.09**
PyUVM no conv	0.92	0.08	**0.06**
PyUVM with conv	0.85	**0.10**	0.11

[in USD]

Cost Analysis - Overall

- Cost to generate VIP, fix VIP, generate TB, fix TB and simulation

	Anthropic	Google	OpenAI
UVM-SV no conv	8.35	**0.62**	0.98
PyUVM no conv	29.94	6.76	**1.06**

[in USD]

Conclusion

- UVM-SV code generation more mature than PyUVM out of the box
- UVM-SV generation can be used for serious DV work
  - Given detailed prompts
  - Add code conventions to fit company rules
  - Great help for debug to get ideas
- Any LLM output needs experienced engineer to cross check results
  - Otherwise LLM might use try … except, comment out code or disable features
- PyUVM needs more research
  - Which context, rules, conventions need to be given

Questions