Pragmatic use cases of ChatGPT in Chip Verification

Authors

Hemamalini Sundaram (hema@verifworks.com)

Ajeetha Kumari Venkatesan (ajeethak@asfigo.com)

accellera SYSTEMS INITIATIVE™

2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Agenda

- Introduction
- Basic requirements to adopt GenAI
- Strategies – inhouse, vendor driven
- Case studies
- Conclusion

Why ChatGPT Verification?

- Knowledge Integration
  - Leverages vast data sources
  - Provides expert-level insights
- Adaptive Learning
  - Improves over time with usage
  - Adapts to specific project needs
- Enhanced Productivity
  - Automates repetitive tasks
  - Reduces manual effort
- Cost Efficiency
  - Reduces need for extensive manpower
  - Lower operational costs

Basic requirements to adopt GenAI

Infrastructure Readiness	• High-performance computing resources • Scalable storage solutions
Data Availability	• Access to large datasets • Quality and diversity of data
Technical Expertise	• Skilled personnel in AI and ML • Training programs for staff
Security Measures	• Data privacy and protection protocols • Robust cybersecurity infrastructure

Basic requirements to adopt GenAI

Regulatory Compliance

- Adherence to industry standards
- Understanding of legal and ethical guidelines

Integration Capabilities

- Compatibility with existing systems
- API and toolchain integration

Budget Allocation

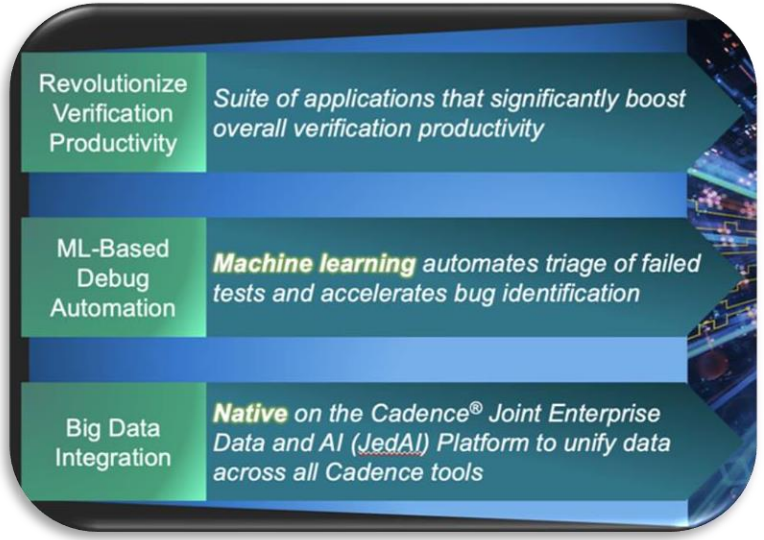
- Funding for initial setup and maintenance
- Investment in continuous improvement

Management Support

- Commitment from leadership
- Strategic alignment with business goals

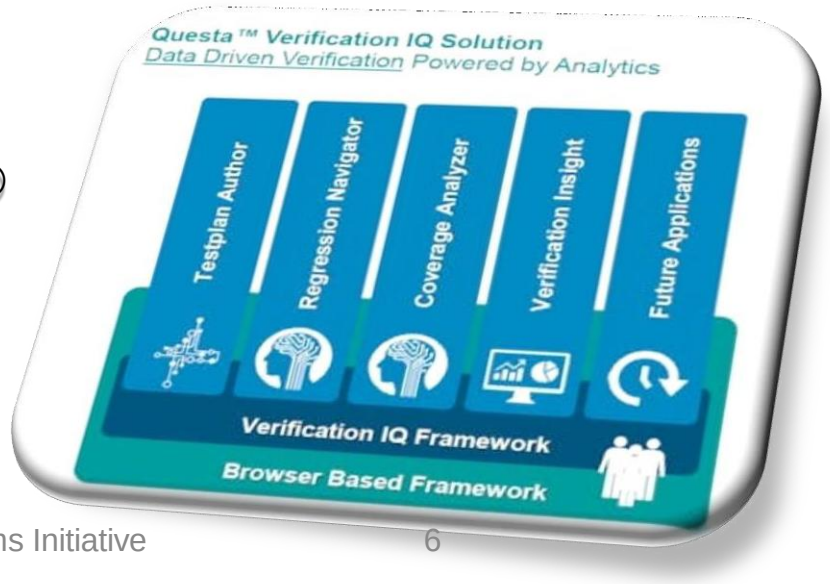
EDA Vendors emerging AI solutions

Cadence ©

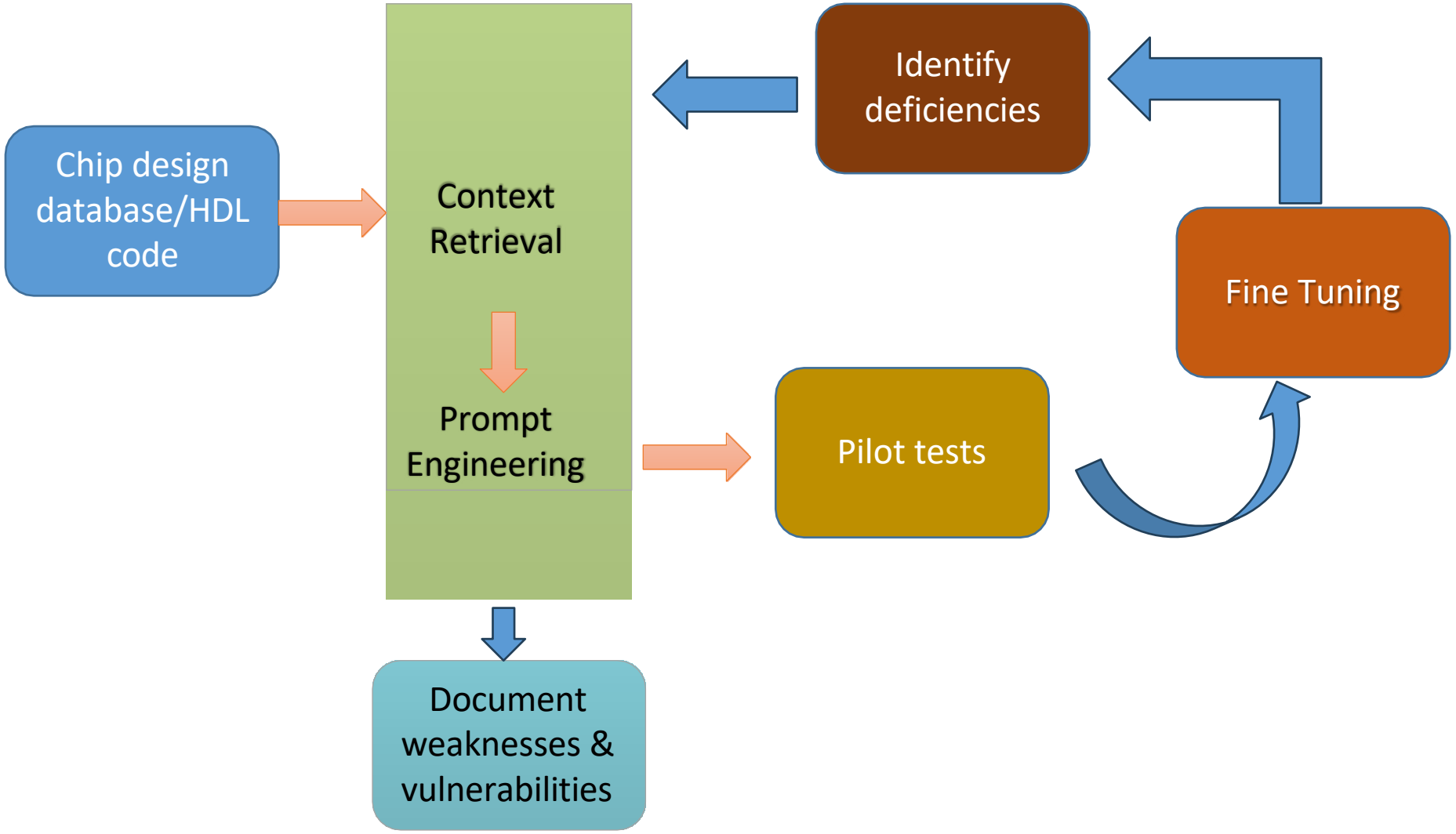


Synopsys ©

Siemens EDA ©



Inhouse LLM deployment

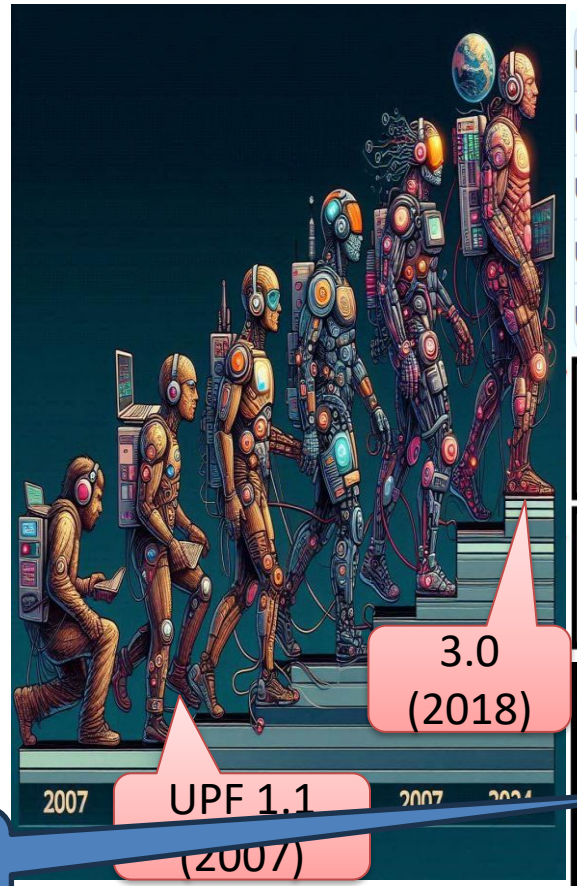


Inhouse LLM based infrastructure

- Work with LLM provider to have secure, private cloud
- Dedicate high-end hardware (GPU, CPU)
- Most likely your SW teams are already doing this
- Get quality consultants who have deployed/customized LLM flows for chip design verification
- Identify short-term goals to prototype, get success, motivate, repeat..

UPF porting with GenAI

- Large SoC with many UPF files (legacy)
- UPF tool support improved across vendors and new requirements on DV
- Power State Coverage – transitions, information model queries etc.
- Tasked with estimating it manually vs. GenAI as prototype solution
- 2 engineers, 3 months, not including validation
- Alternate: GenAI



UPF Version	Total Commands	Power Intent	Power Management	Query Commands
UPF 3.0	62	51	6	5
UPF 2.1	86	46	6	34
UPF 2.0	88	49	-	39
UPF 1.0	32	-	-	-

UPF 1.1

```
set_isolation ISO_STRAT \
  -domain SUB0 \
  -isolation_power_net VDD \
  -isolation_ground_net GND \
  -clamp_value 0
```

```
set_isolation_control ISO_STRAT \
  -domain SUB0 \
  -isolation_signal reg_out[25] \
  -isolation_sense high \
  -location parent
```

UPF 3.0 (2.1)

Merged to single command

```
create_pst PM_pst –supplies\
 { VDD    u0/VDD_SUB0_SW  VDD_SUB1   VDD_SUB2 }

add_pst_state pst0 –pst PM_pst –state \
 { VDD_N  SW_on           SUB1_H     SUB2_H}

add_pst_state pst1 –pst PM_pst –state \
 { VDD_N  SW_off          SUB1_L     SUB2_L}
```

Replaced with add_power_state* commands

UPF porting with GenAI

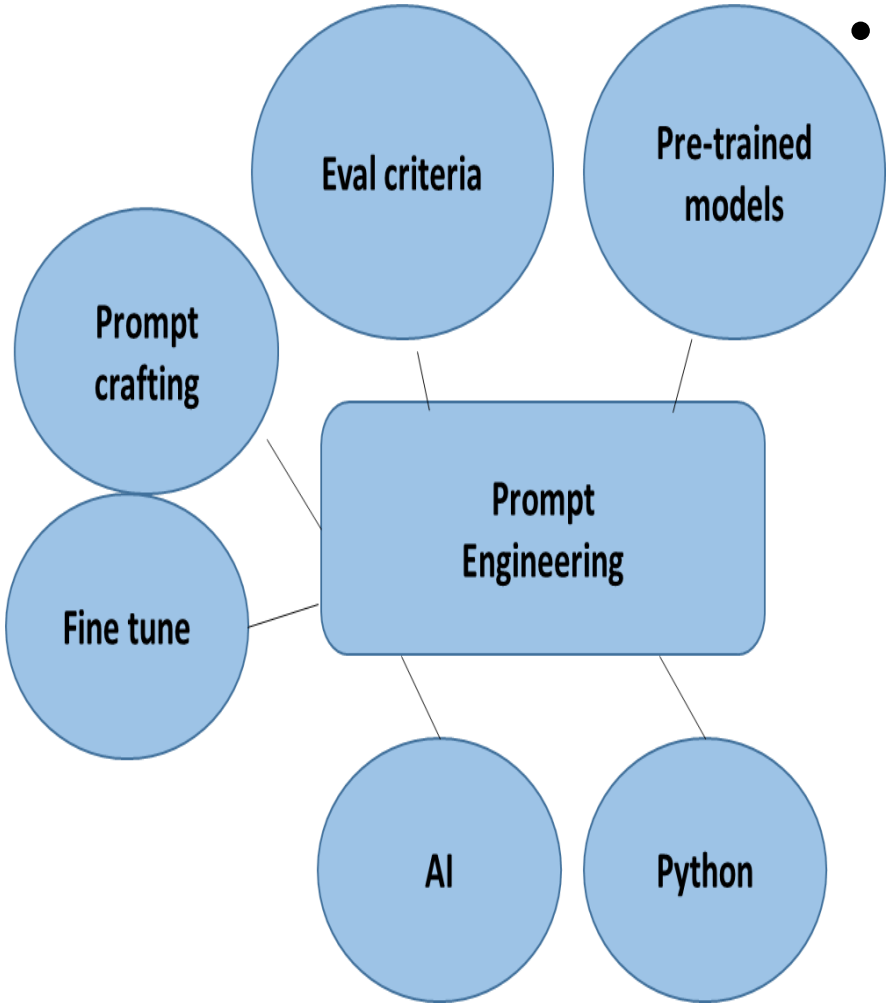
Phase-1

- Extract key UPF (3.0) features across design flow/tools
- Devise unit-testing framework to exercise various UPF elements (Simulation centric)
  - Abstract out DUT, use stubs
- Build basic tests for:
  - Isolation
  - Power State ON/OFF
  - Retention elements
  - Power State Coverage
- Train AI model to create these tests

Phase-2

- GenAI to migrate UPF
  - Start small, ISO commands first target
  - Level shifters, supply network next
  - Power State tables as last
    - Train AI to do the syntax migration
- Run unit tests
    - Abstract out DUT, use stubs
  - Pick relevant tests
    - Isolation
    - Power State ON/OFF
    - Retention elements
    - Power State Coverage
  - Train AI model to create these tests
  - Run new UPF code against UPF 1.1 (manual) vs. UPF 3.0 (AI Generated)

Case study – SVA unit tests



- EDA vendors are working on generating SVA from prompts

  Inhouse – develop unit testing flow for such SVAs using LLM

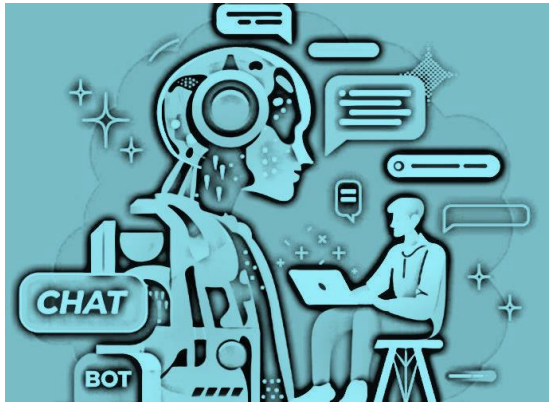
  – Helps validate EDA vendors' emerging tools for current project

  – Helps in picking right vendor solution for your projects

  – Enables workforce to adopt Prompt Engineering and be AI-ready!

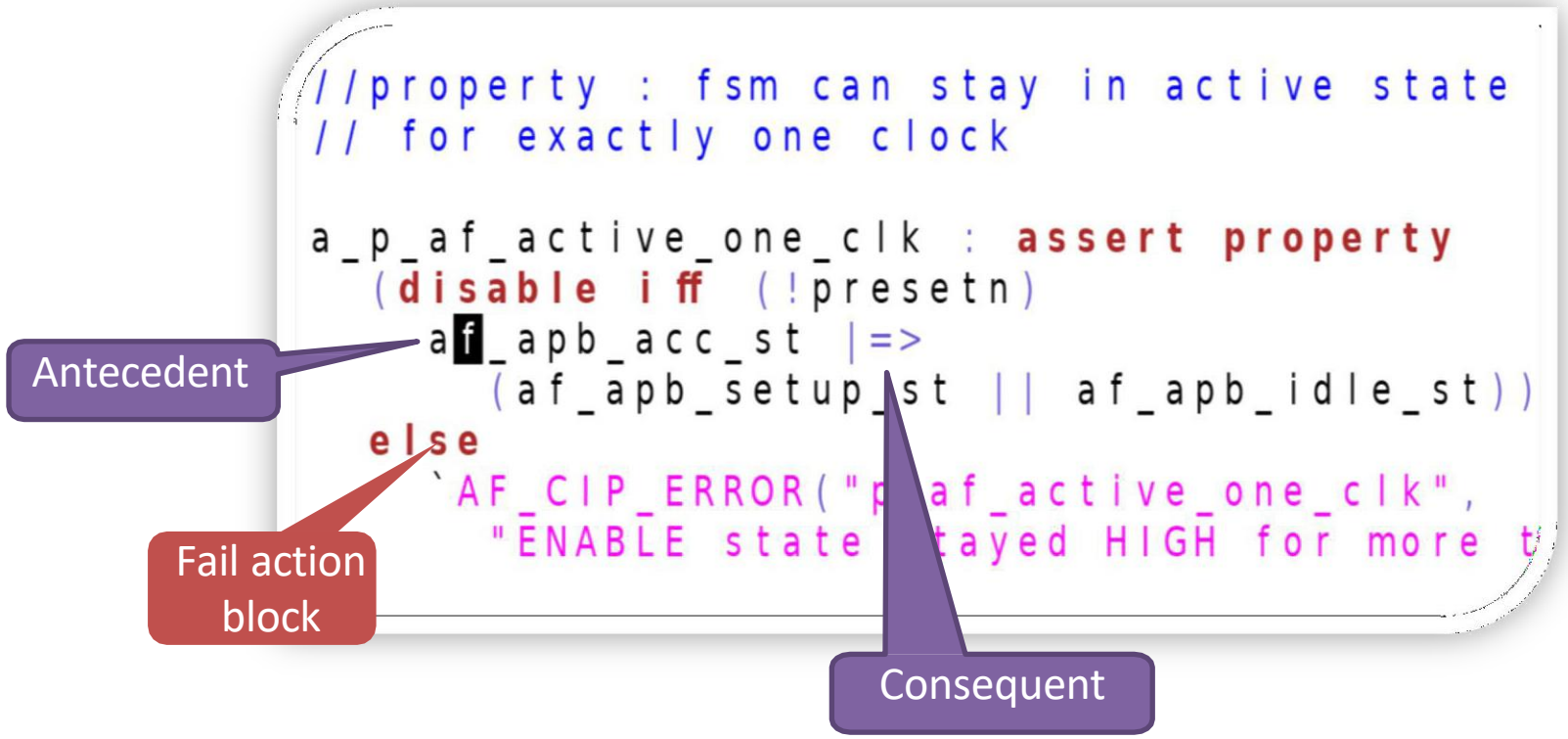
LLM assisted Unit test development

- Input SVA
- Custom crafted prompt
- Templates using GO2UVM library + SVUnit
- Be ready for few iterations to get it right
- Once done, LLM can do wonderful job!

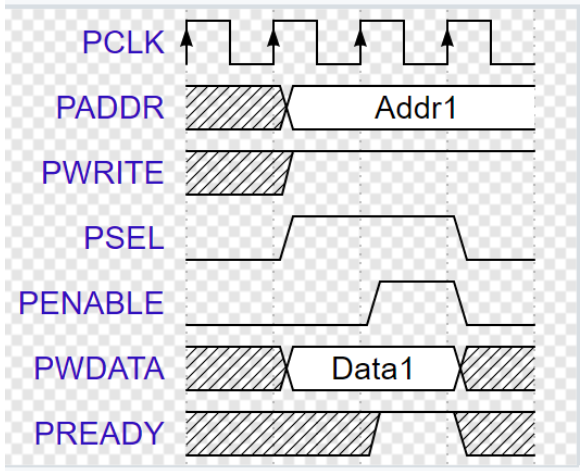


SVUnit
a new chance to lead

Sample APB assertion



Unit Tests for APB assertion



psel && !pen
|=> pen

```
//  Desc:
//    Unit test for CIP_ID: a_p_af_psel_pen
//    psel = 1, next clock penable = 0
//    Expected result: FAIL
//*****************************************
***
`SVTEST(fail_a_p_af_psel_pen)

   `g2u_display ("fail_a_p_af_psel_pen")
   idle ();
   `g2u_display ("Driving psel to 1")
   psel = 1'b1;
   wait_for_n_clks (1);
   `g2u_display ("Driving pen to 0")
   penable = 1'b0;
   wait_for_n_clks (1);
   penable = 1'b0;
   idle ();
   wait_for_n_clks (10);
   `FAIL_IF_LOG(1, "Expected FAIL a_p_af_pse
   `g2u_display ("End fail_a_p_af_psel_pen")

`SVTEST_END
```

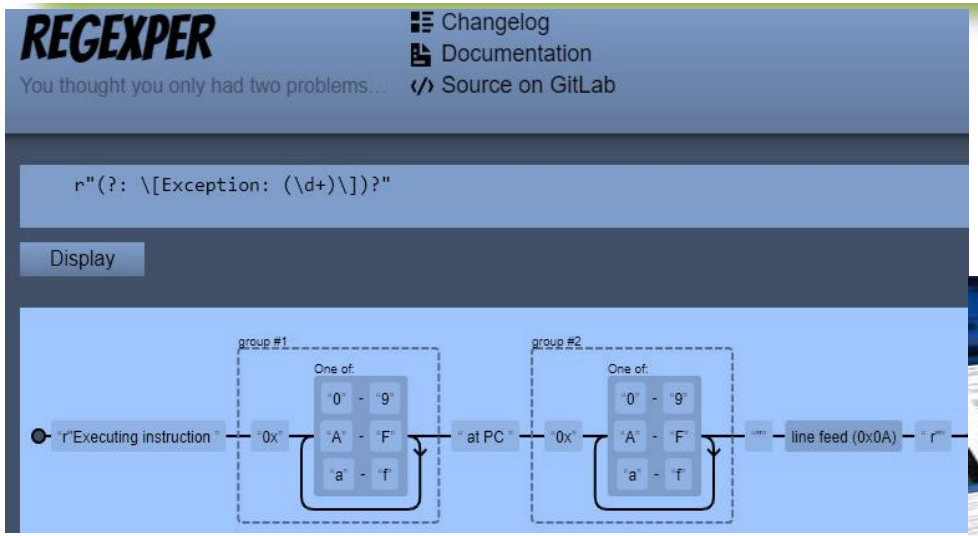
Unit tests – testsuite, PASS/FAIL

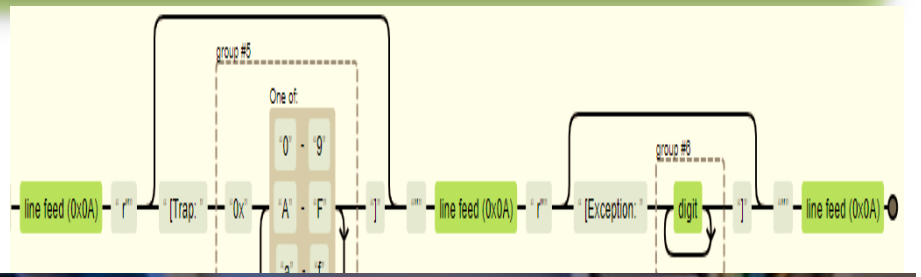
```
INFO:  [0.000 ns][testrunner]: Registering Test Suite __ts
INFO:  [370.000 ns][apb_slave_ut]: pass_a_p_af_psel_pen::PASSED
INFO:  [620.000 ns][apb_slave_ut]: fail_a_p_af_psel_pen::FAILED
INFO:  [720.000 ns][apb_slave_ut]: pass_a_p_af_active_one_clk::PASSED
INFO:  [850.000 ns][apb_slave_ut]: fail_a_p_af_active_one_clk::FAILED
INFO:  [1050.000 ns][apb_slave_ut]: pass_a_p_af_idle_paddr_stable::PASSED
INFO:  [1250.000 ns][apb_slave_ut]: fail_a_p_af_idle_paddr_stable::FAILED
INFO:  [1500.000 ns][apb_slave_ut]: pass_a_p_af_idle_pwrite_stable::PASSED
INFO:  [1700.000 ns][apb_slave_ut]: fail_a_p_af_idle_pwrite_stable::FAILED
INFO:  [1950.000 ns][apb_slave_ut]: pass_a_p_af_paddr_stable_bw_setup_and_en::PASSED
INFO:  [2200.000 ns][apb_slave_ut]: fail_a_p_af_paddr_stable_bw_setup_and_en::FAILED
INFO:  [2320.000 ns][apb_slave_ut]: pass_a_p_af_pwdata_stable_bw_setup_and_en::PASSED
INFO:  [2440.000 ns][apb_slave_ut]: fail_a_p_af_pwdata_stable_bw_setup_and_en::FAILED
INFO:  [2640.000 ns][apb_slave_ut]: pass_a_p_af_pwrite_stable_bw_setup_and_en::PASSED
INFO:  [2790.000 ns][apb_slave_ut]: fail_a_p_af_pwrite_stable_bw_setup_and_en::FAILED
INFO:  [3010.000 ns][apb_slave_ut]: pass_a_p_af_rst_paddr::PASSED
INFO:  [3250.000 ns][apb_slave_ut]: fail_a_p_af_rst_paddr::FAILED
INFO:  [3250.000 ns][apb_slave_ut]: FAILED (8 of 16 tests passing)
INFO:  [3250.000 ns][__ts]: FAILED (0 of 1 testcases passing)
INFO:  [3250.000 ns][testrunner]: FAILED (0 of 1 suites passing) [SVUnit Verilator Patch]
```


```
UVM_INFO ../src/APB_CIP_src//test_a_p_af_pwrite_stable_bw_setup_and_en.sv(53) @ 2750.000 ns [IVL_GO2UVM] Driving pwrite to
UVM_INFO ../src/APB_CIP_src//af_apb_cip.sv(45) @ 2755.000 ns [IVL_GO2UVM] psel: 1 pen: 1
UVM_INFO ../src/APB_CIP_src//test_a_p_af_pwrite_stable_bw_setup_and_en.sv(56) @ 2760.000 ns [IVL_GO2UVM] Driving pwrite to
UVM_ERROR ../src/APB_CIP_src//af_apb_cip.sv(81) @ 2765.000 ns [a_p_af_idle_pwrite_stable]  pwrite is not stable when APB is
UVM_ERROR ../src/APB_CIP_src//af_apb_cip.sv(155) @ 2765.000 ns [a_p_af_pwrite_stable_bw_setup_and_en] pwrite is not stable
ERROR: [2790.000 ns][apb_slave_ut]: fail_if: 1 [ Expected FAIL a_p_af_pwrite_stable_bw_setup_and_en ] (at ../src/APB_CIP_sr
and_en.sv line:61)
UVM_INFO ../src/APB_CIP_src//test_a_p_af_pwrite_stable_bw_setup_and_en.sv(62) @ 2790.000 ns [IVL_GO2UVM] End fail_a_p_af_pw
INFO:  [2790.000 ns][apb_slave_ut]: fail_a_p_af_pwrite_stable_bw_setup_and_en::FAILED
INFO:  [2790.000 ns][apb_slave_ut]: pass_a_p_af_rst_paddr::RUNNING
UVM_INFO ../src/APB_CIP_src//test_a_p_af_rst_paddr.sv(12) @ 2885.000 ns [IVL_GO2UVM] pass_a_p_af_rst_paddr
UVM_INFO ../src/APB_CIP_src//test_a_p_af_rst_paddr.sv(13) @ 2885.000 ns [IVL_GO2UVM] Driving rst_n to 0
UVM_INFO ../src/APB_CIP_src//test_a_p_af_rst_paddr.sv(15) @ 2885.000 ns [IVL_GO2UVM] Driving paddr to 0
UVM_INFO ../src/APB_CIP_src//test_a_p_af_rst_paddr.sv(20) @ 3010.000 ns [IVL_GO2UVM] End pass_a_p_af_rst_paddr
INFO:  [3010.000 ns][apb_slave_ut]: pass_a_p_af_rst_paddr::PASSED
INFO:  [3010.000 ns][apb_slave_ut]: fail_a_p_af_rst_paddr::RUNNING
UVM_INFO ../src/APB_CIP_src//test_a_p_af_rst_paddr.sv(35) @ 3105.000 ns [IVL_GO2UVM] fail_a_p_af_rst_paddr
UVM_INFO ../src/APB_CIP_src//test_a_p_af_rst_paddr.sv(37) @ 3120.000 ns [IVL_GO2UVM] Driving rst_n to 0
UVM_INFO ../src/APB_CIP_src//test_a_p_af_rst_paddr.sv(39) @ 3120.000 ns [IVL_GO2UVM] Driving paddr to 1
UVM_ERROR ../src/APB_CIP_src//af_apb_cip.sv(121) @ 3125.000 ns [p_af_output_at_reset] paddr is not low at reset
ERROR: [3250.000 ns][apb_slave_ut]: fail_if: 1 [ Expected FAIL a_p_af_rst_paddr ] (at ../src/APB_CIP_src//test_a_p_af_rst_p
UVM_INFO ../src/APB_CIP_src//test_a_p_af_rst_paddr.sv(45) @ 3250.000 ns [IVL_GO2UVM] End fail_a_p_af_rst_paddr
INFO:  [3250.000 ns][apb_slave_ut]: fail_a_p_af_rst_paddr::FAILED
INFO:  [3250.000 ns][apb_slave_ut]: FAILED (8 of 16 tests passing)

INFO:  [3250.000 ns][__ts]: FAILED (0 of 1 testcases passing)
```

Regexp & DV engineers

```
pattern = re.compile(
    r"Executing instruction (0x[0-9A-Fa-f]+) at PC (0x[0-9A-Fa-f]+)"
    r"(?: \[Cycle (\d+)\])?"
    r"(?: \[Interrupt: (\d+)\])?"
    r"(?: \[Trap: (0x[0-9A-Fa-f]+)\])?"
    r"(?: \[Exception: (\d+)\])?"
)
```





Use LLM to decrypt regexp

```
# Regular expression to match two IPv4 addresses
if ($log_entry =~ /source (\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}) to DEST (\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})/) {
    print "Matched source IP: $1\n";
    print "Matched destination IP: $2\n";
} else {
    print "No match found\n";
}
```

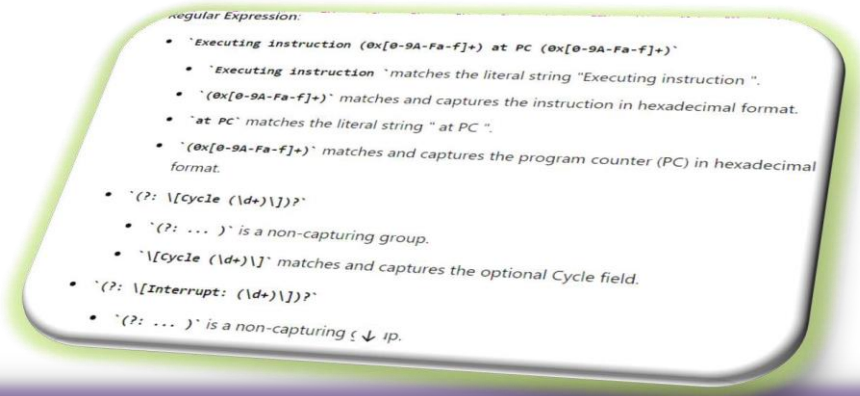
```
Sending new packet from source 126.128.9.110 to DEST 168.202.255.255

Packet from 192.168.1.1 to 10.0.0.1 dropped

Establishing connection from 172.16.0.1 to 172.16.0.2
```

Explanation

- Sample Log Entry: The provided log entry includes two IPv4 addresses.
- Regular Expression: `source (\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}) to DEST (\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})`
  - `source ` matches the literal string "source ".
  - `(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})` matches and captures the first IPv4 address (source IP).
    - `\d{1,3}` matches one to three digits.
    - `\.` matches the period separating the octets.
    - This pattern repeats for all four octets of the IP address.
  - `to DEST` matches the literal string " to DEST ".
  - `(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})` matches and captures the second IPv4 address (destination IP).

RISCV Trace debug & LLM

```
pattern = re.compile(
    r"Executing instruction (0x[0-9A-Fa-f]+) at PC (0x[0-9A-Fa-f]+)"
    r"(?: \[Cycle (\d+)\])?"
    r"(?: \[Interrupt: (\d+)\])?"
    r"(?: \[Trap: (0x[0-9A-Fa-f]+)\])?"
    r"(?: \[Exception: (\d+)\])?"
)
```




```
Executing instruction 0x20000013 at PC 0x80000000 [Cycle 17028] [Interrupt: 1]
[Trap: 0x0] [Exception: 2]
Executing instruction 0x00000297 at PC 0x80000004
Executing instruction 0x00028067 at PC 0x80000008 [Cycle 17029]
Executing instruction 0x12345678 at PC 0x80000010 [Cycle 17030] [Exception: 3]
```

Lessons learnt, next steps

- Use custom UVM library (e.g. GO2UVM), Unit test frameworks and tune LLM to that subset
- Pick the appropriate strategy for your GenerativeAI deployment
- Leverage on known-expertise in DV with LLM
- Managers/Leads can quickly help juniors to debug complex scripts
- Consult your EDA vendors for latest in their AI offerings

Questions

Thank you