A novel approach to generate random/constrained Non-Volatile-Memory content in a UVM environment

Davide Sanalitro
Digital Verification Senior Team Leader
STMicroelectronics

SPONSORED BY CEDA IEEE Council on Electronic Design Automation sigda acm

Challenge and needs

A verification engineer needs an automated flow that can:

- generate random or constrained NVM content
- create tests to verify all possible values
- allow the user to easily modify the NVM content based on the specific test being executed.

Typical approach

- Manually writing the content of the NVM bit by bit (text file) or generating it using custom scripts before launching the simulation.

Process is time-consuming

Loss of dynamism

No coverage collection

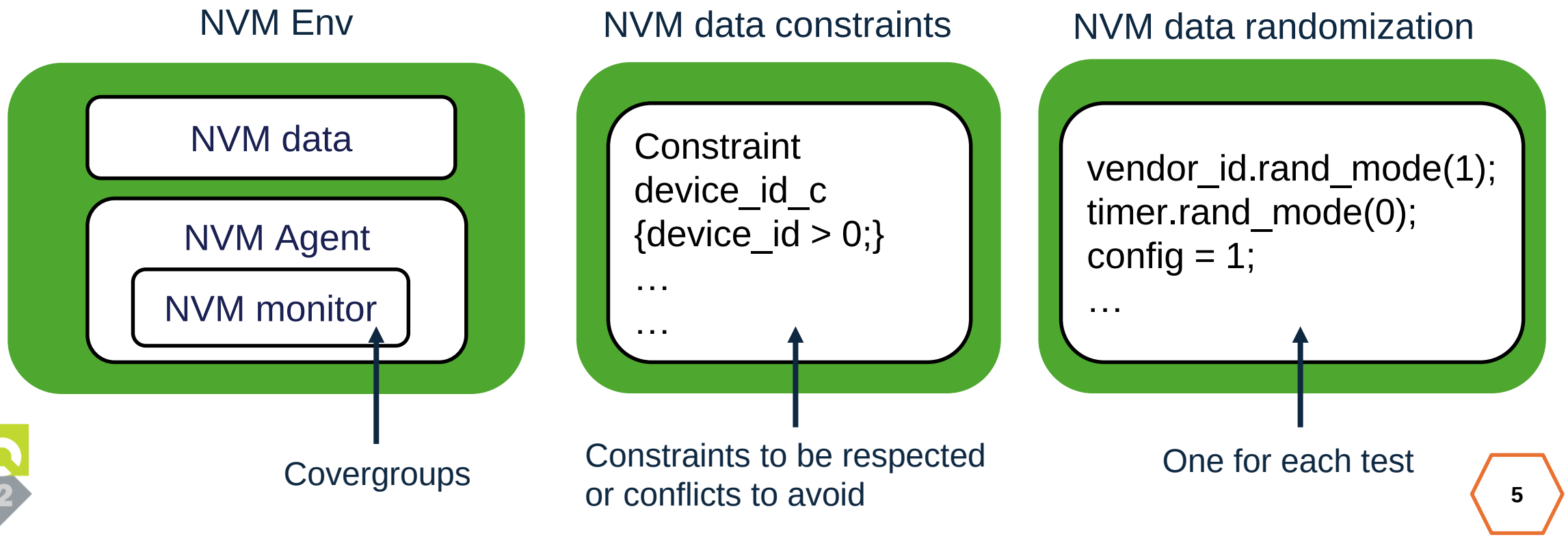
Main idea - Excel to UVM VIP

REGISTER	ADDRESS	FIELD	WIDTH	LSB	NVM Sect	NVM Word
DEVICE_ID	1	vendor_id	2	6	2	0
		device_id	6	0		
TIMER	2	timer	8	0	0	1
CONFIG	3	config	8	0	0	2
…	…	…	…	…	…	…

- The user can use any open-source language (like Python) to read the Excel file and generate a **UVM verification IP** (Universal Verification Methodology) of the **NVM** containing the expected structure of the memory.

Main idea – Creating randomization files

- A passive **UVM VIP** will be generated as required by the UVM **standard**.
- Two **custom** additional files will also be generated that need to be filled in by the user.

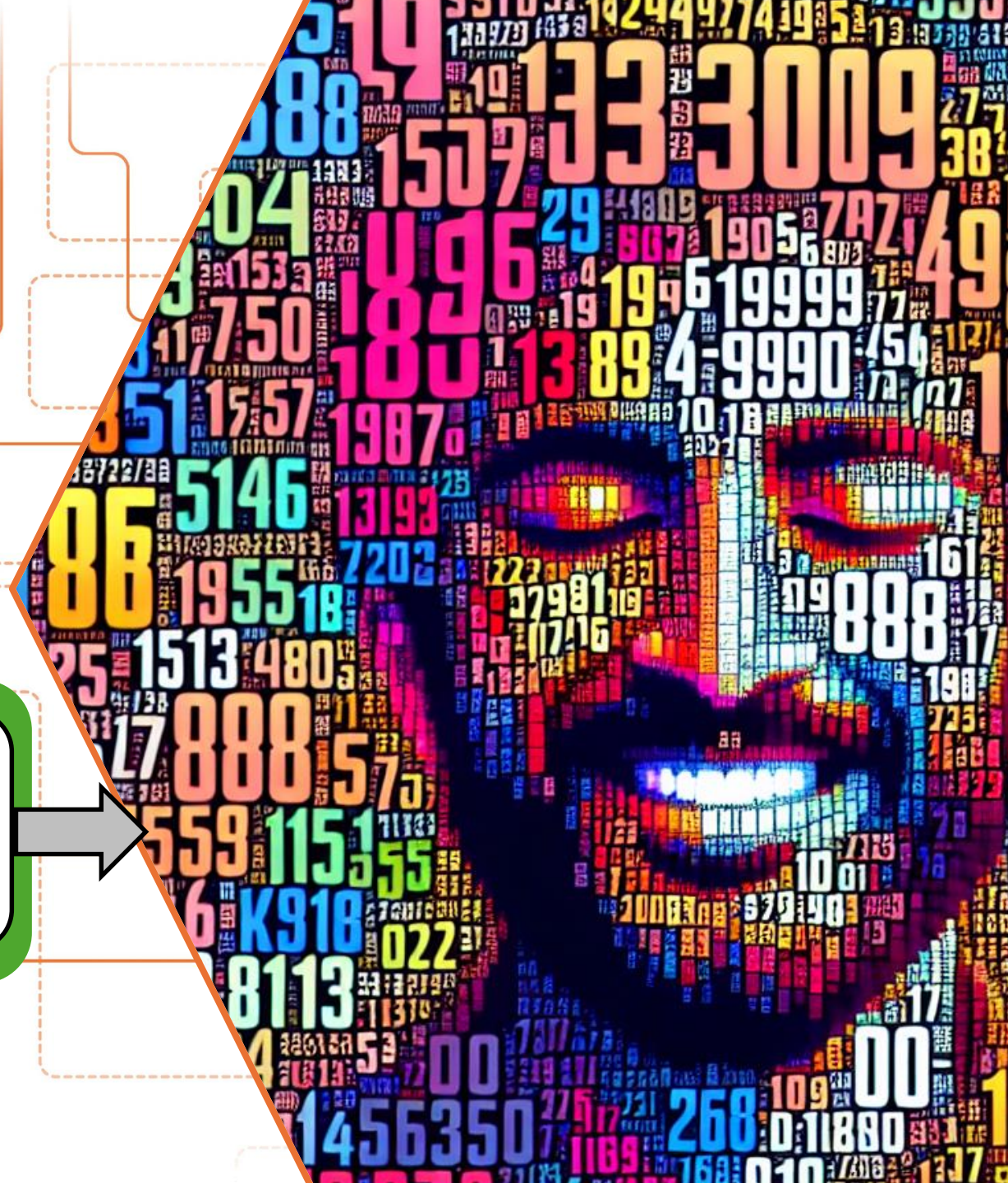


Main idea – generating test

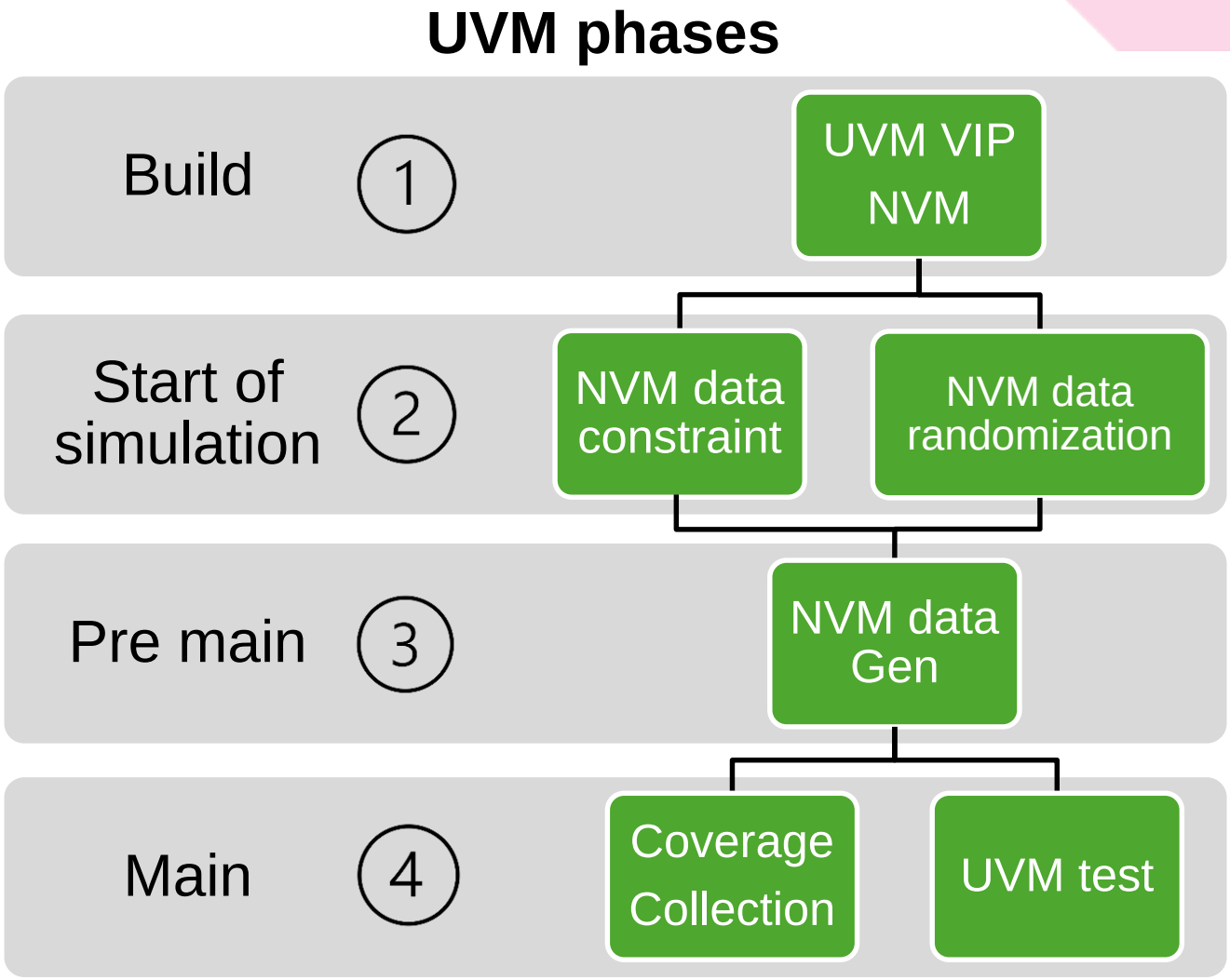
Now, it is necessary to ensure that the test writes to all registers that will be programmed in NVM.

The **uvm_test** is automatically generated starting from the excel file.

```
DEVICE_ID.write(status, nvm_data_i.device_id);
TIMER.write(status, nvm_data_i.device_id);
CONFIG.write(status, nvm_data_i.device_id);
...
```



Novel approach



// ** NVM data generation ** //

Sect0:
00000000080000000001180
Sect1:
0000000000D7D70000000200
Sect2:
008000D200D0000000220000
Sect3:
000000000000000000000008
…

Benefits

Thanks to this novel approach:


No need to remember where a field is located in NVM


Ease of assigning new values to fields

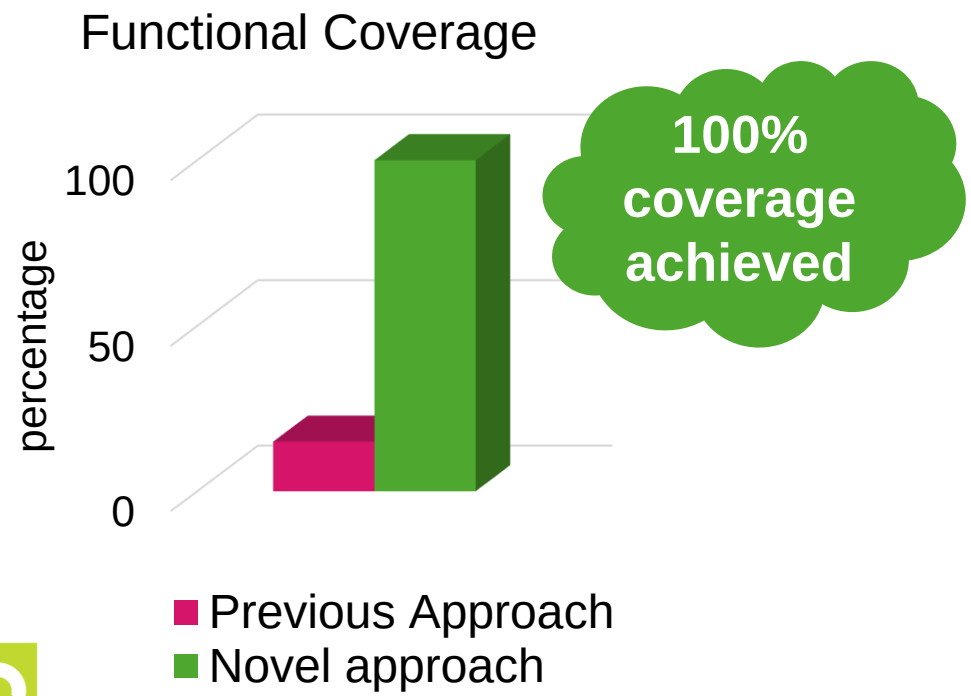

Automated flow, no errors due to manual intervention


Randomization & coverage collection

Evidence – functional coverage

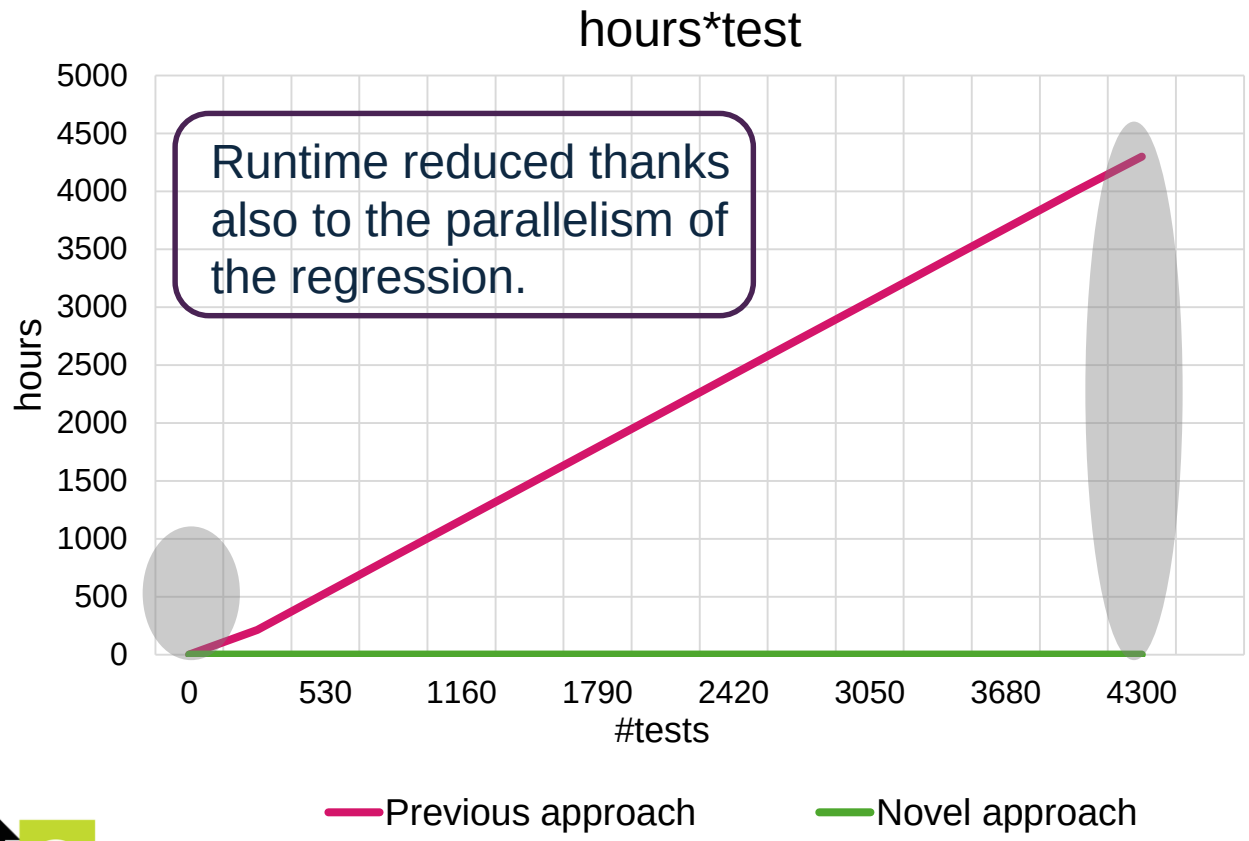
The SoC under analysis has the following NVM:

- 4Kb, 930 fields



- **Previous approach:**
  - 160 test.
  - ~15% of values verified.
  - Unknown number of sequences/tests needed for full scenario coverage.
- **Novel approach:**
  - 4300 tests, 100% of values verified.
  - No additional tests are needed, all scenarios covered.
  - Found bugs.

Evidence – time spent in generating all scenarios



Setup + simulation time for each test:

- Novel approach: ~1 minute.
- Previous approach: ~1 hour.

Novel approach:

- the fastest way to verify all NVM values.
- It is **2000 times** faster than the previous approach.

Conclusion

- This novel approach allows to:
  - Speed up the generation of random NVM values **up to 2000 times**
  - Achieve maximum coverage (**100%**) in timeframes **compatible with product development**
  - Improve the quality of silicon

- The idea is tool (simulator) and language independent: this makes it a solution easily extendable to any project without any limitations.