A Mutually-Exclusive Deployment of Formal and Simulation Techniques Using Proof-Core Analysis

Keerthikumara Devarajegowda, Jeroen Vliegen

Infineon Technologies AG

Goran Petrovity, Kawe Fotouhi

Cadence design Systems

accellera SYSTEMS INITIATIVE

infineon cadence

2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

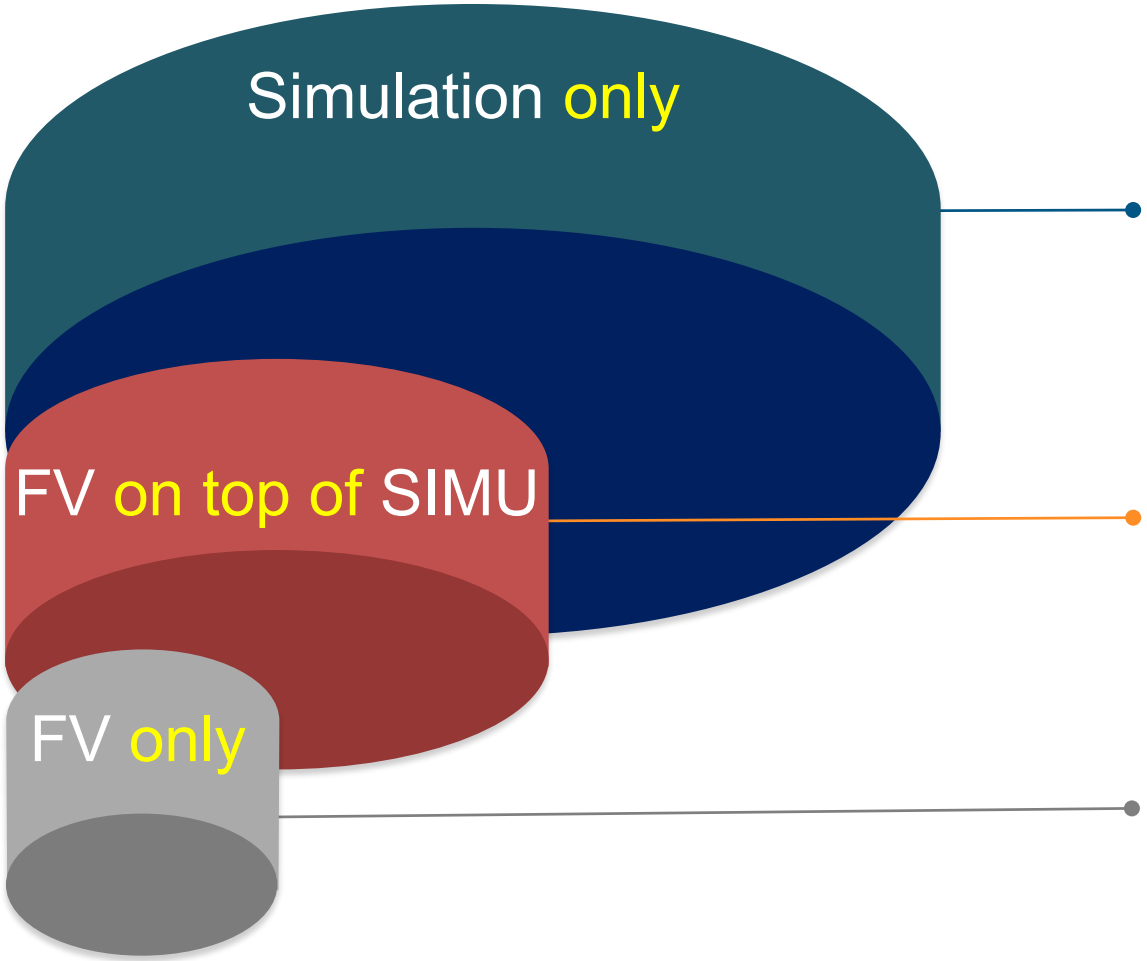
Agenda

- Problematic
- Proposed Flow
- Application on a Mixed-Signal Design
  - Verification Split
  - Formal Verification Flow
  - Simulation Reductions
- COV DB Merging
- Conclusion

Agenda

- **Problematic**
- Proposed Flow
- Application on a Mixed-Signal Design
  - Verification Split
  - Formal Verification Flow
  - Simulation Reductions
- COV DB Merging
- Conclusion

Usage of Formal & Simulation on the same Design

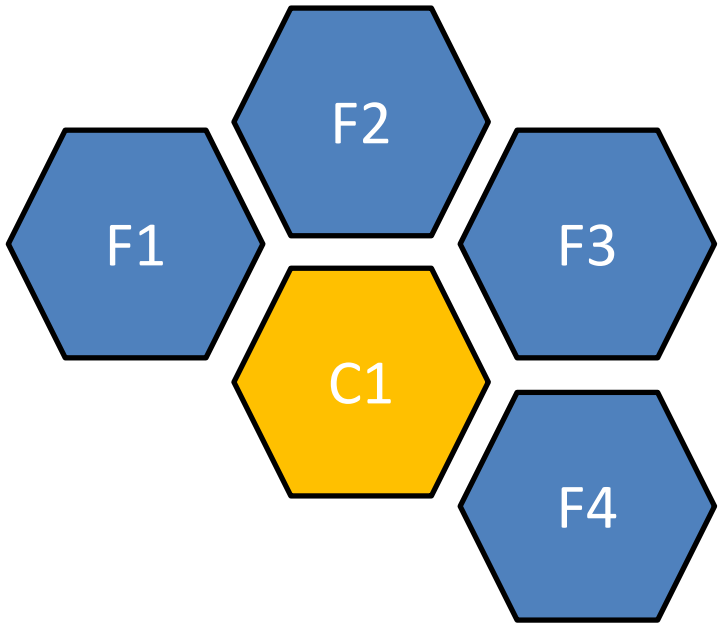


- Usage : 80-100%
- Solid UVM methodology
- Scales from IP upto SoC
- …
- Usage : 10-20%
- Hunt for corner cases
- Prove SoC connectivity
- …
- Usage : 5-10%
- @ IP boundary
- Automated with APPS
- …

Formal On Top of Simulation

Given an IP with:

4 basic and 1 complex feature



Case: formal on top of simulation

- all features are verified with simulation

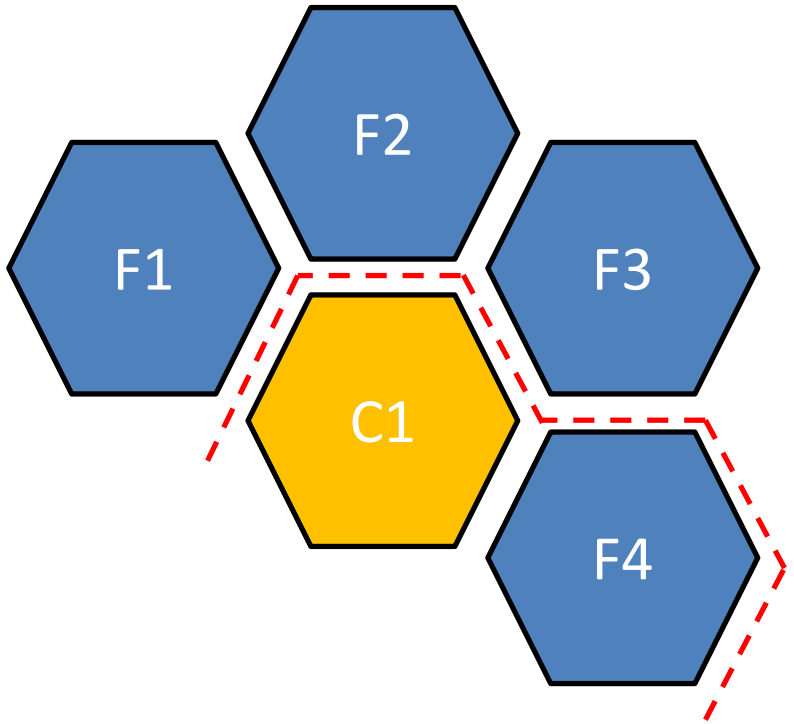
- C1 is checked in addition with formal

We have:

- increased efforts

+ increased quality

Formal Mutually-Exclusive to Simulation



Case: formal mutually exclusive to simulation

- 

are simulated

- Features 

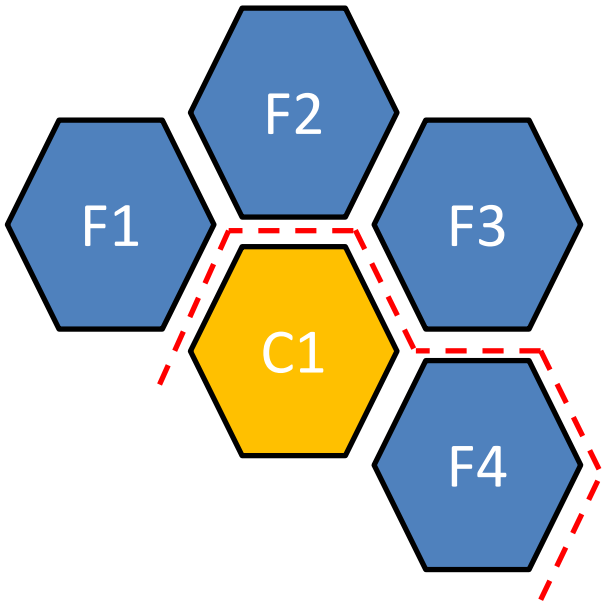
are formally proven

We have:

+ reduced simulation efforts

+ increased quality

Obstacles for going Mutually Exclusive

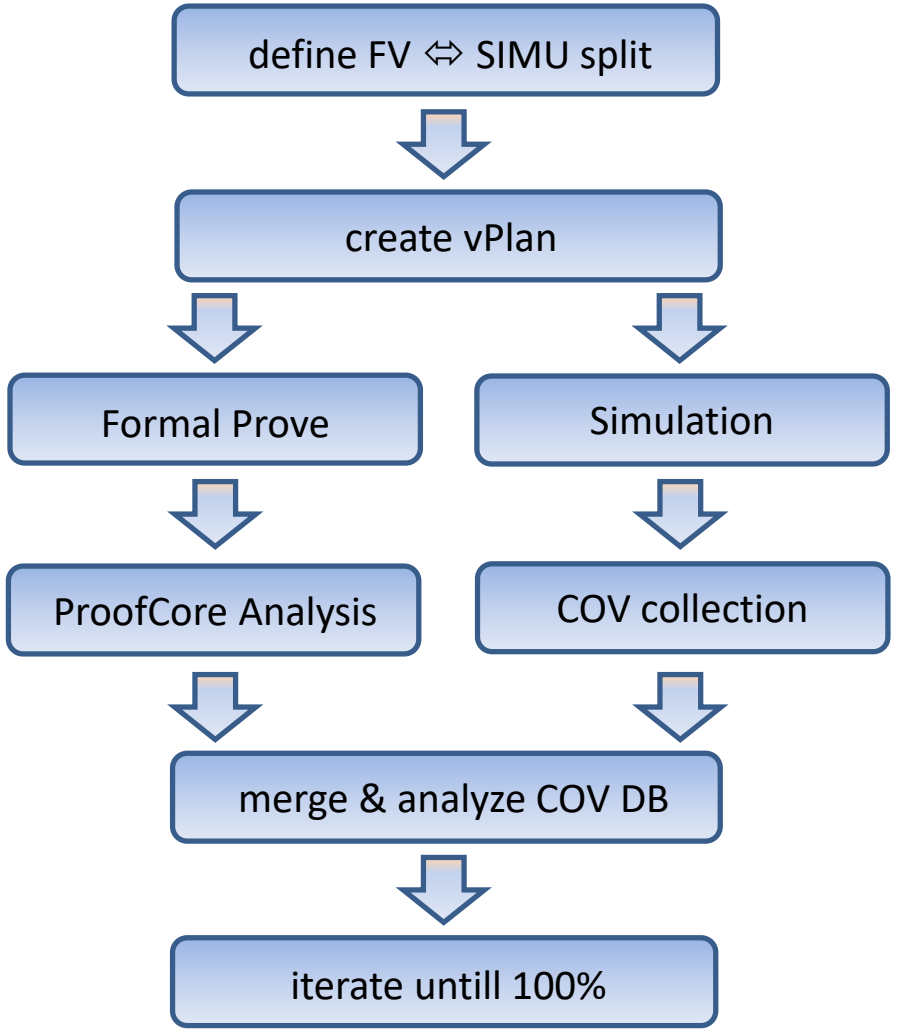


- Making a smart decision on the split requires expertise
- Features may traverse module boundaries
- APP automation might be missing
- …
- Assessing the simu & formal results to conclude whether we are „done“ might not be trivial
- A uniform and profound completeness metrics analysis spanning across formal and simulation is often in-existent
- …

Agenda

- Problematic
- **Proposed Flow**
- Application on a Mixed-Signal Design
  - Verification Split
  - Formal Verification Flow
  - Simulation Reductions
- COV DB Merging
- Conclusion

Proposed Flow

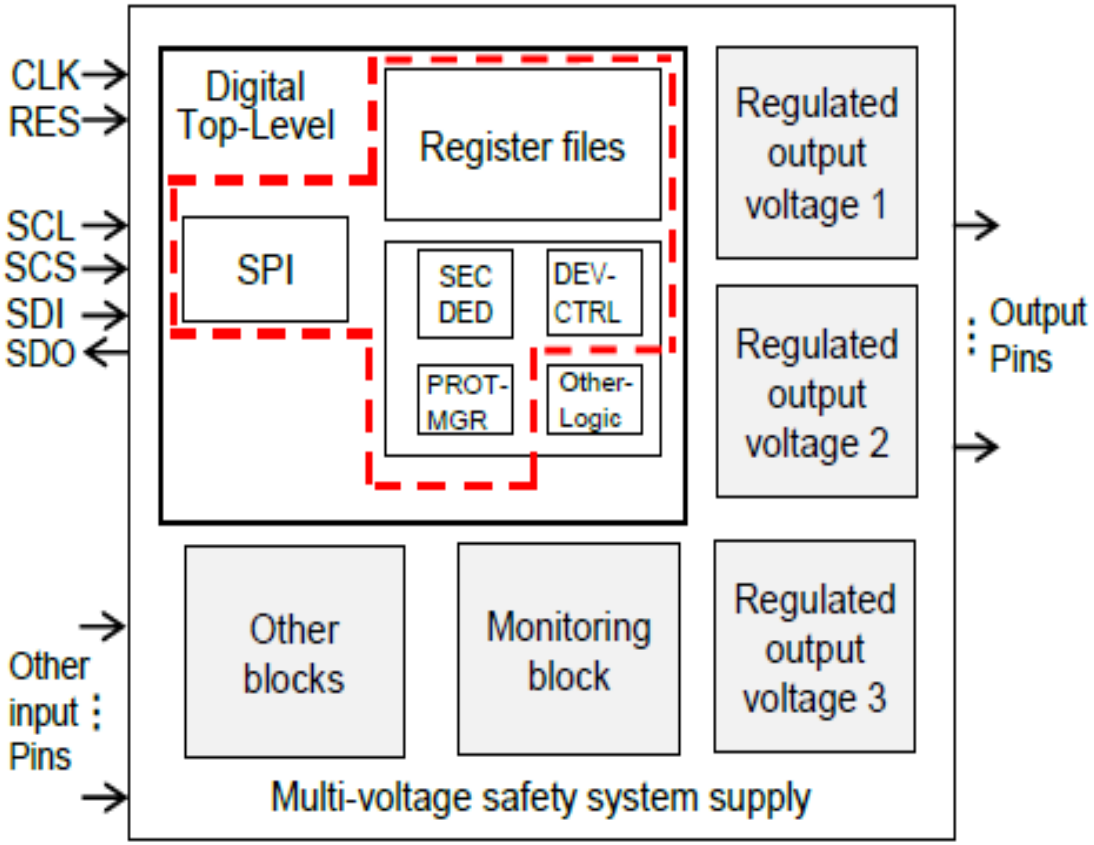


- The flow on the left was created within the scope of a master thesis as a Proof-of-Concept in order to combine formal and simulation on the same design.
- So far it has not been integrated in the Infineon design flow yet

Agenda

- Problematic
- Proposed Flow
- **Application on a Mixed-Signal Design**
  - **Verification Split**
  - Formal Verification Flow
  - Simulation Reductions
- COV DB Merging
- Conclusion

DUT & VE



- DUT: ADAS multi voltage safety system supply
- SIMU: UVM SystemVerilog, NCSIM/Xcelium
- Property language: SVA
- Formal Tool: JasperGold

Choosing Formal Friendly Blocks

Concurrent blocks

Sequential blocks with low sequential depth

Type of Design-block

Sequential blocks with high sequential depth

Control blocks

Data Transfer blocks

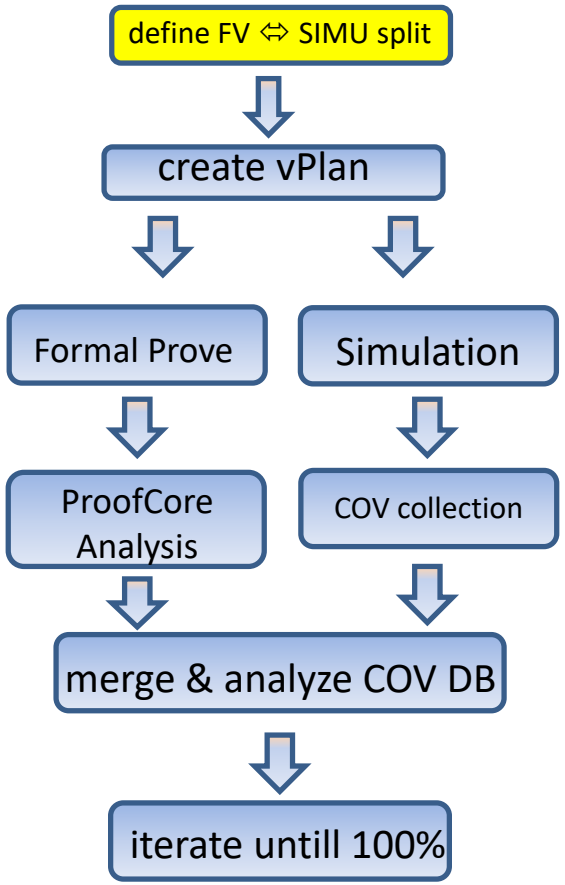
Type of Operation

Data Transform blocks with intensive arithmetic operations

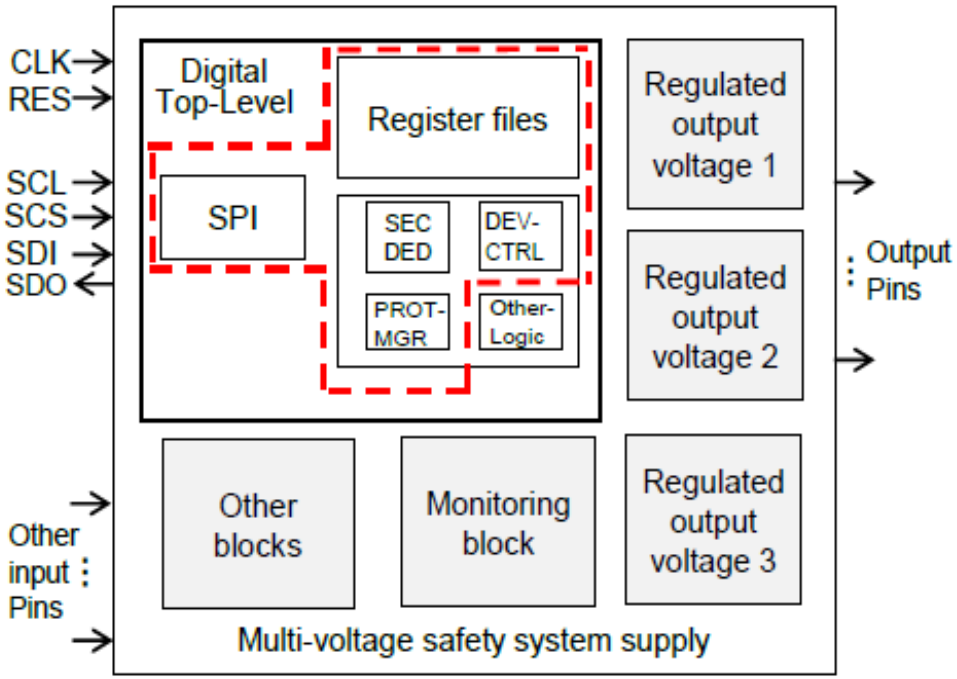
Data Transform blocks with less intensive arithmetic operations

Size of Design-block (# flip-flops)

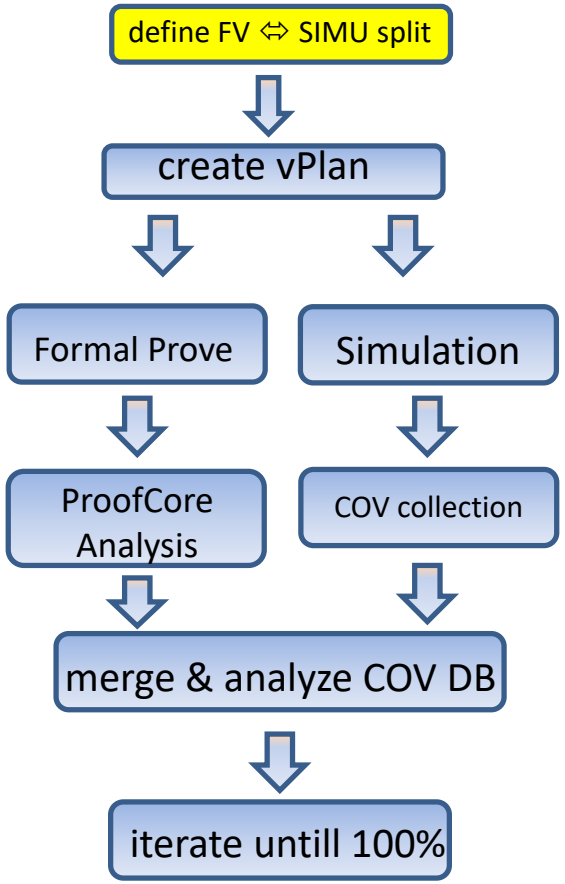
Consider Capabilities of formal engines, Bounded-proofs & Abstraction techniques!



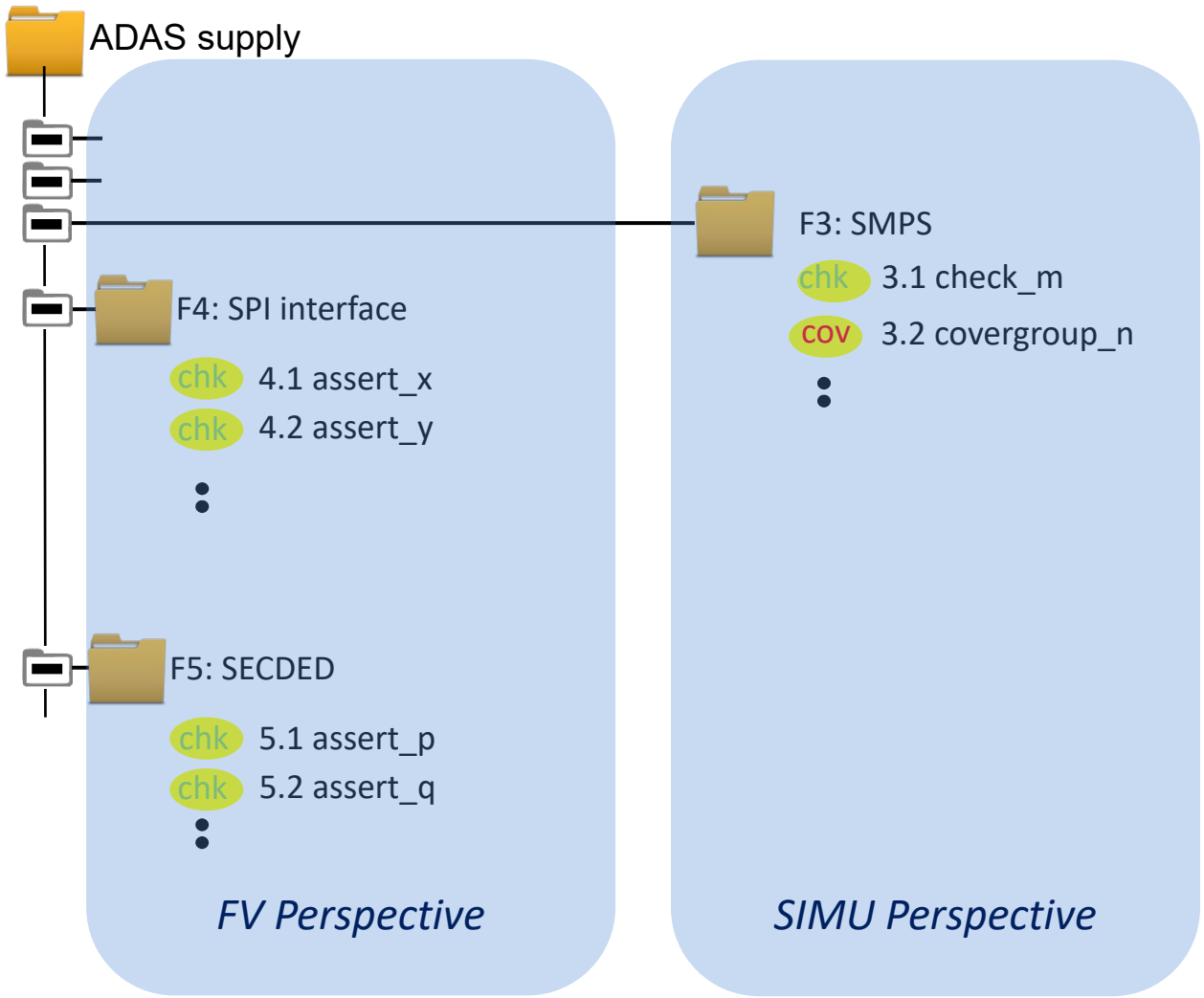
Chosen Blocks for Formal



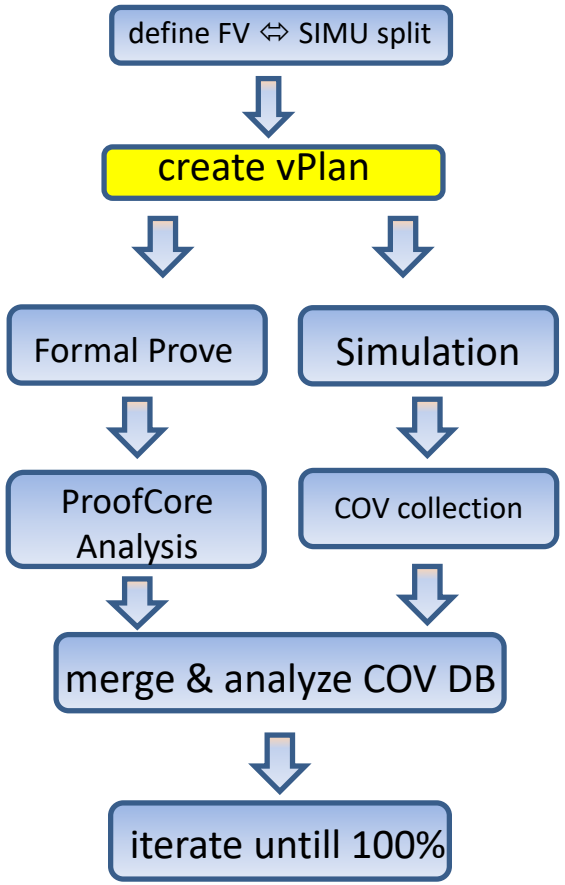
- ✓SPI interface
- ✓Register files
- ✓SECDED logic
- ✓PROT logic
- ✓DEVCTRL logic
- ✓(potentially more)



Verification Plan



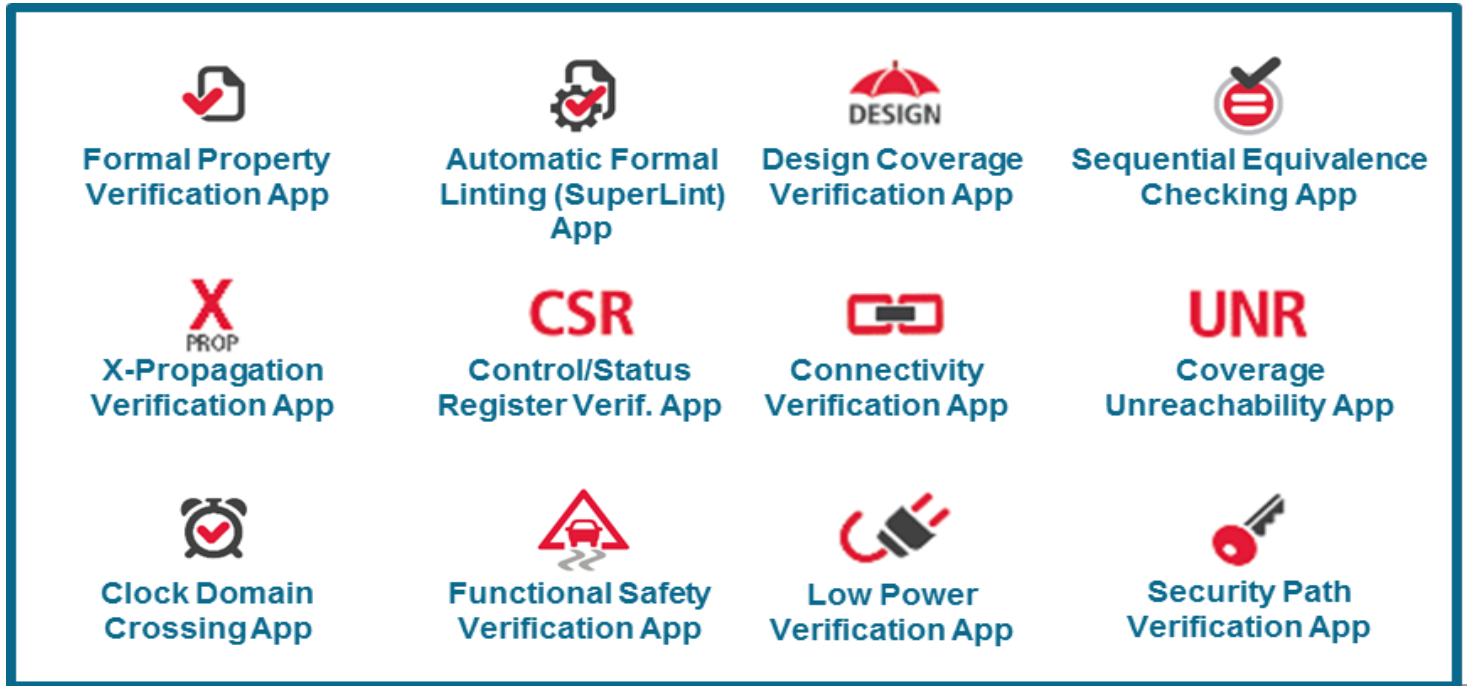
- ✓ Feature tree
- ✓ SIMU perspective
  - ✓ Covergroups
  - ✓ Checks
- ✓ FV perspective
  - ✓ Assertions
- ✓ Assume-Guarantee
  - ✓ left out (simplicity)

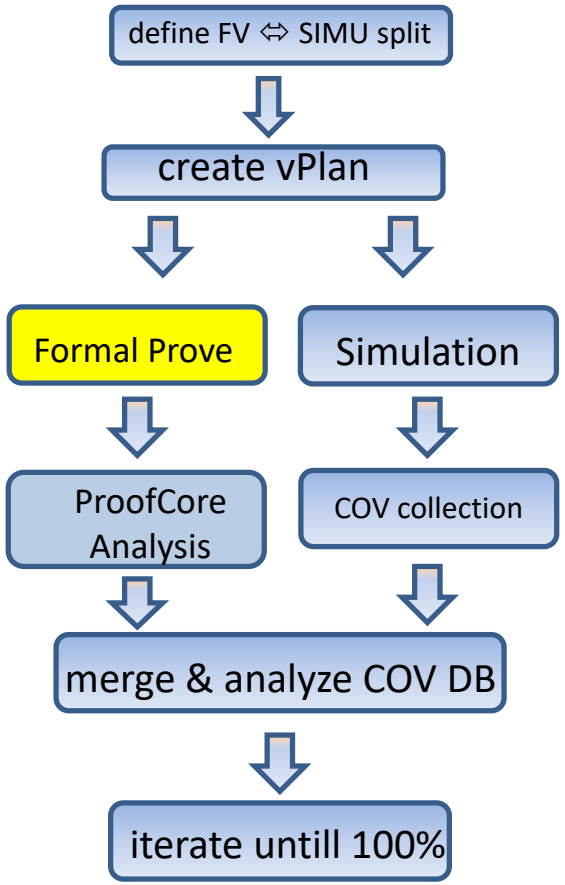


Agenda

- Problematic
- Proposed Flow
- **Application on a Mixed-Signal Design**
  - Verification Split
  - **Formal Verification Flow**
  - Simulation Reductions
- COV DB Merging
- Conclusion

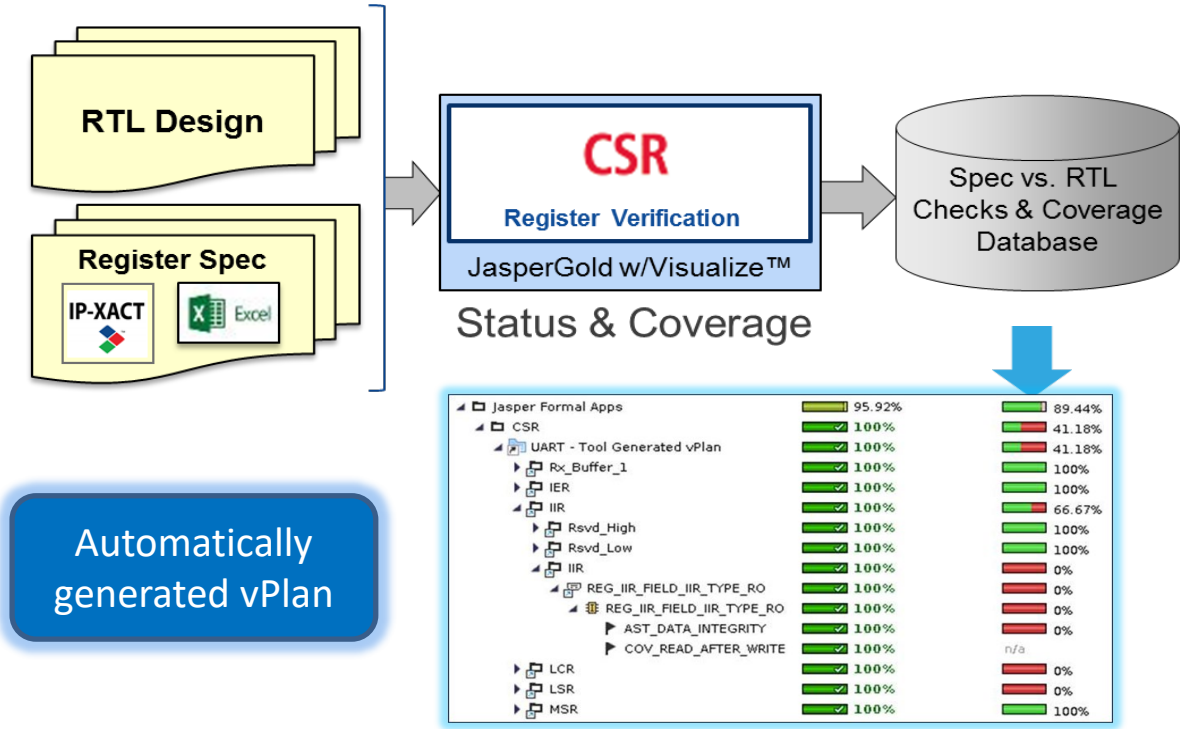
Formal Automated Apps

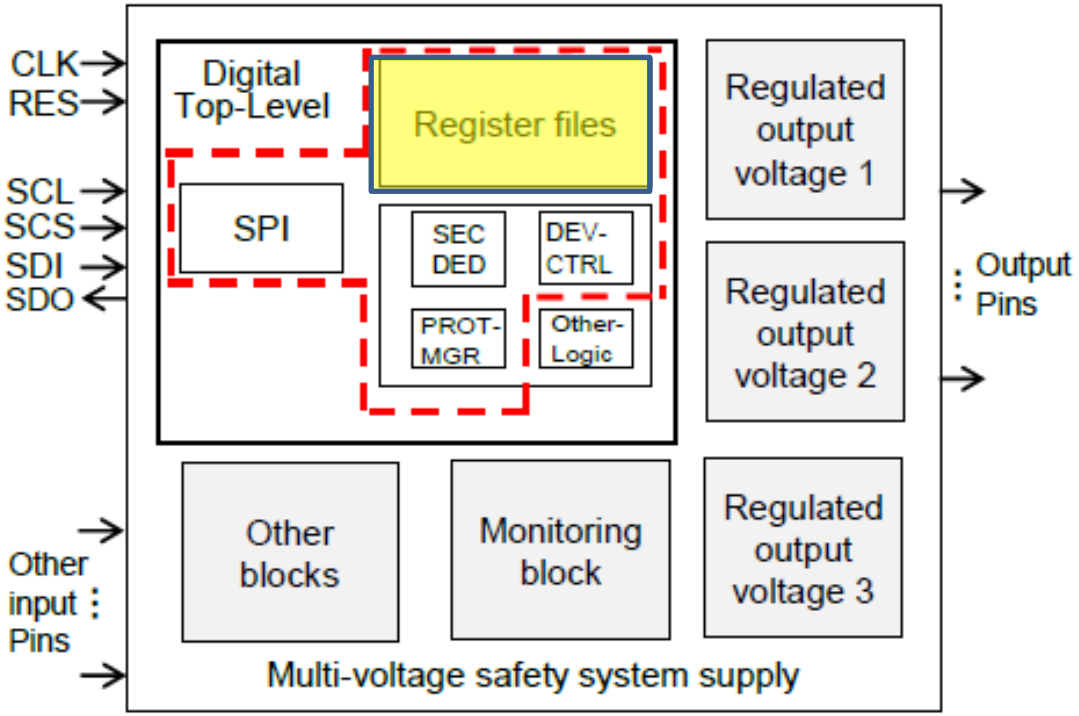




- We used JasperGold for the formal prove
- The register files were proven with the **C**ontrol and **S**tatus **R**egister App
- All other properties were handwritten

CSR Verification

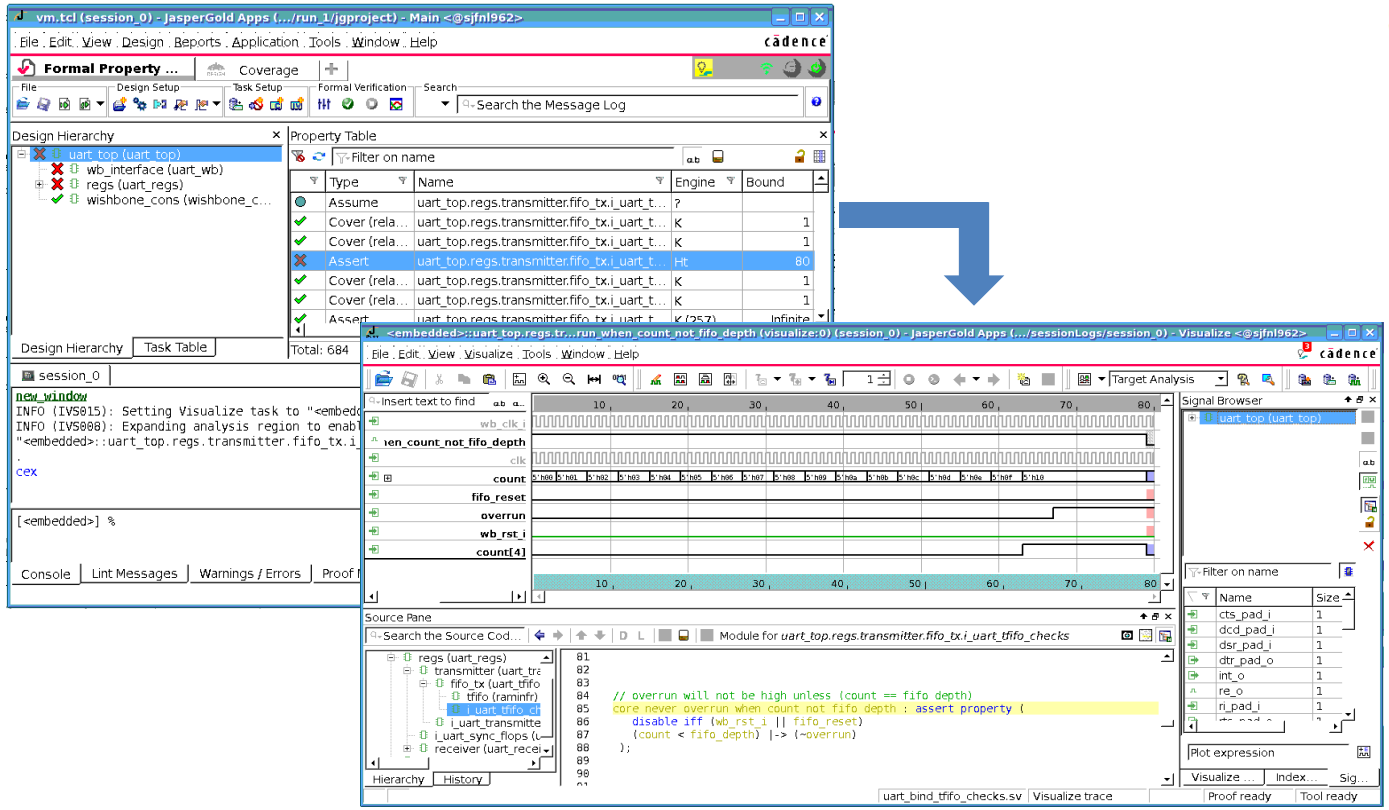


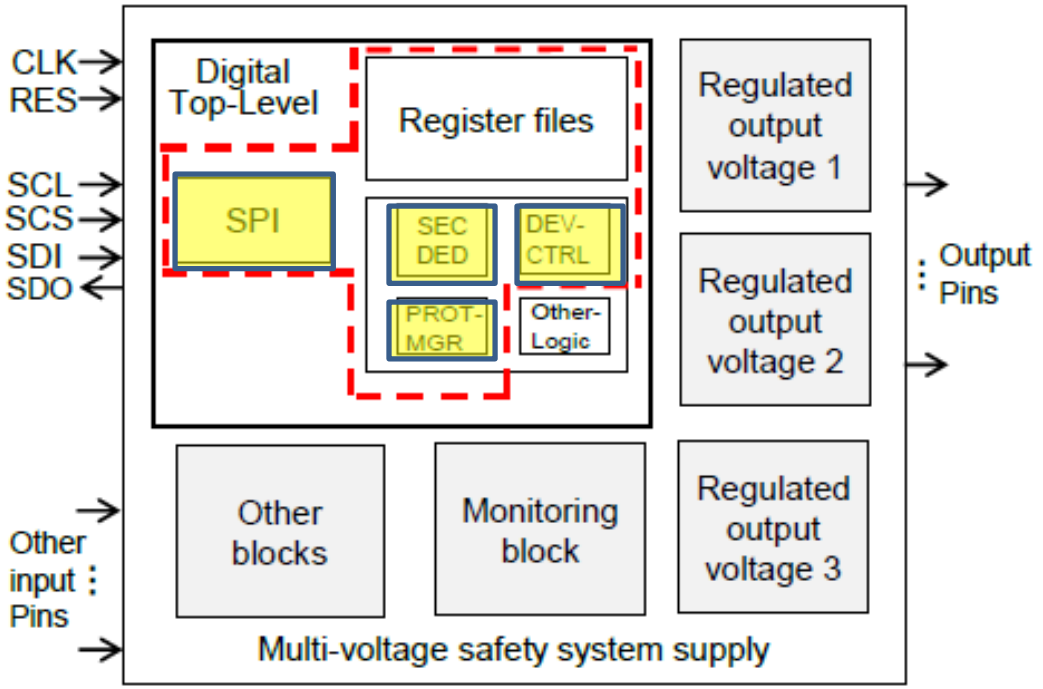


- IP-XACT automatically extracted (IFX automated flow)
- 62 regs -> ~3.7K props
- Several access policies used, volatile accesses, …

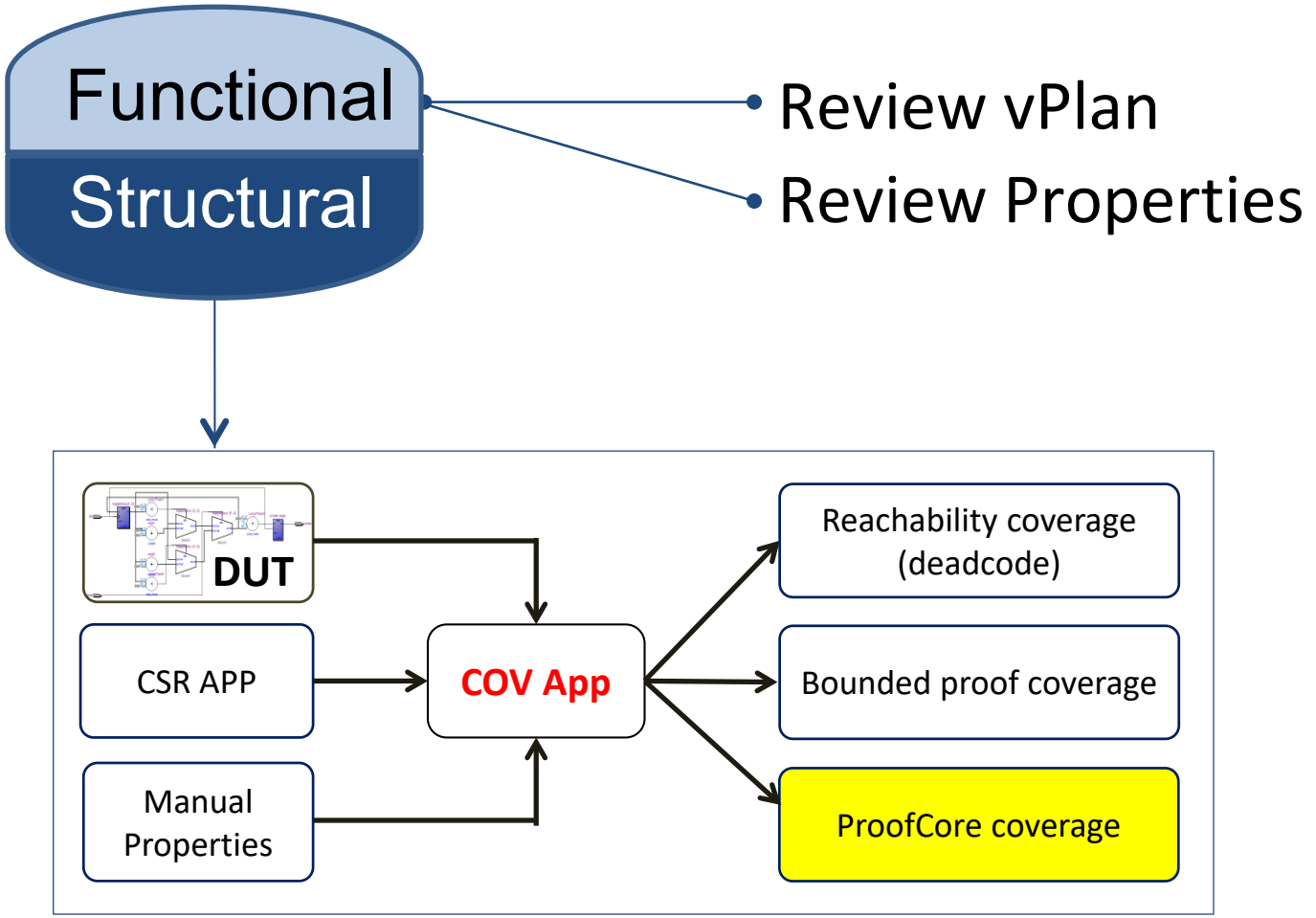
Handwritten properties

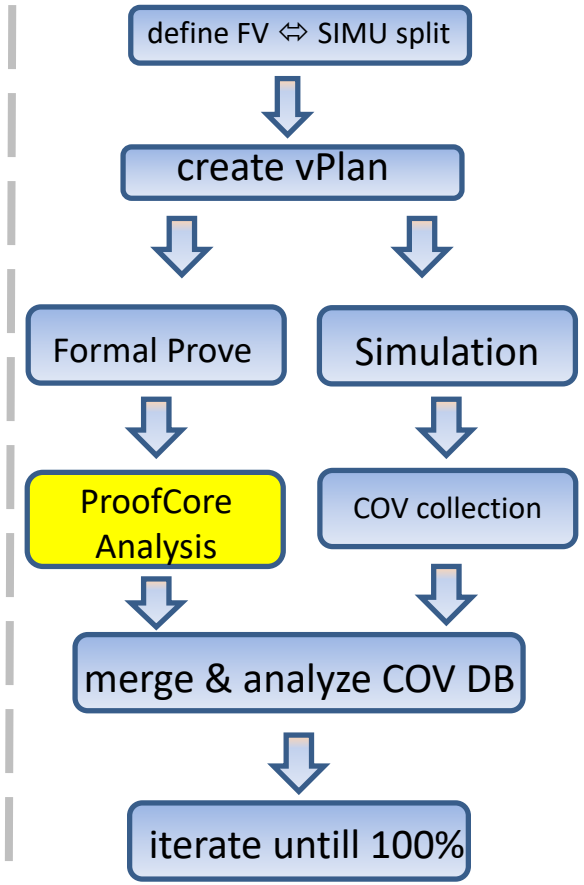
- Written 140 props (by hand)
- Proven in ~16 minutes





Completeness

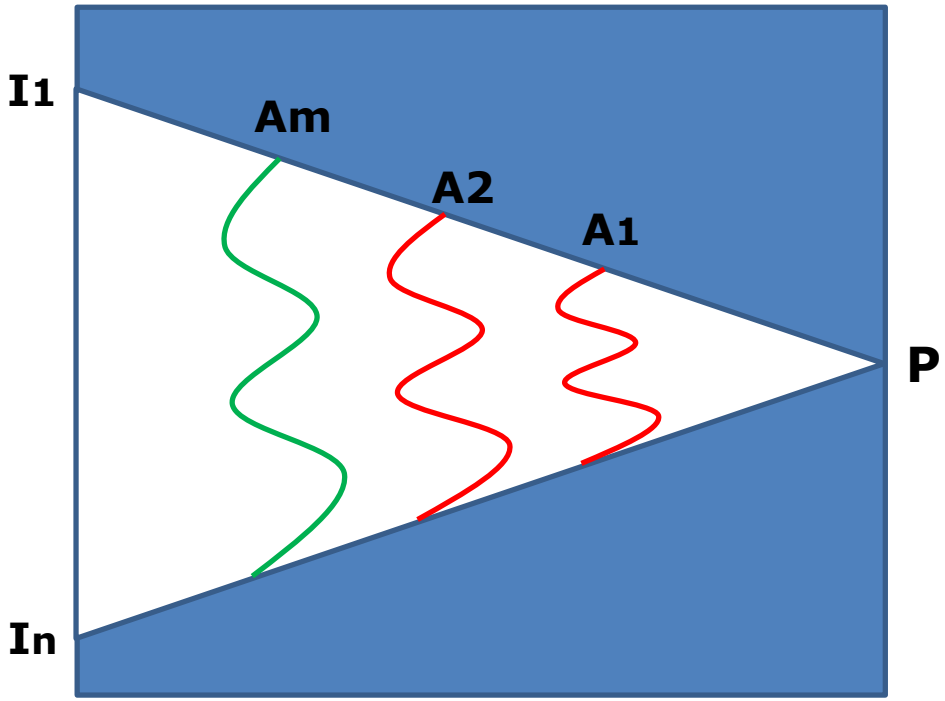




COI abstraction process => ProofCore

Given:

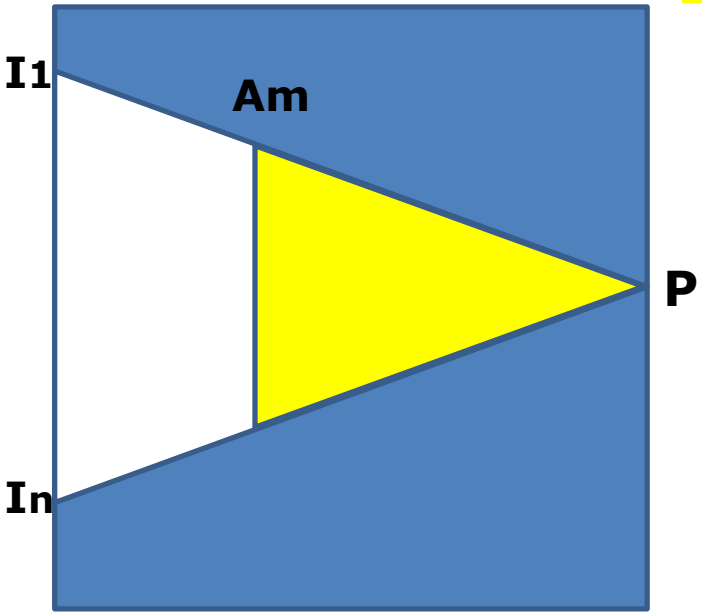
- Property **P**
- **COI = P I1 In**



Proof Process:

- Abstract the COI -> Ax
- Prove (P) on the abstracted COI
  - if P fails -> CEX
    - Prove (P) on full COI
    - if CEX -> true negative !
    - else -> extend abstraction
  - if P holds
    - ProofCore = **P Am**

ProofCore

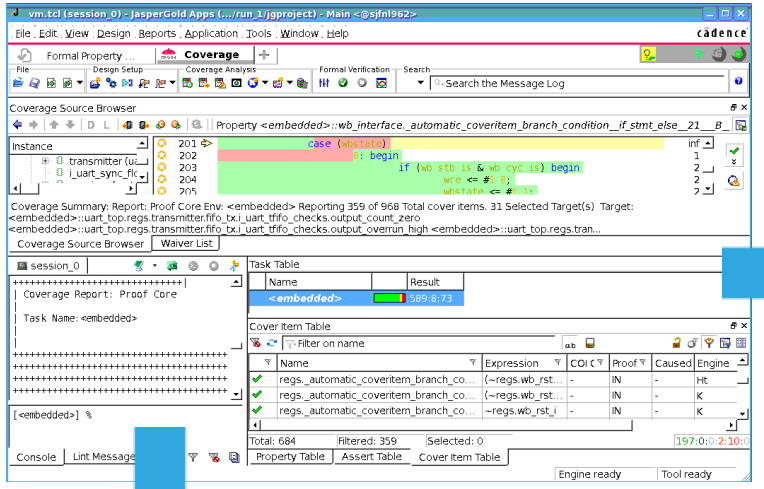


Given:

- Proofcore **P Am**

Structural Metrics:

- RTL code coverage
  - Today
    - Block
    - Statement
    - *Expression*
  - Coming
    - Toggle



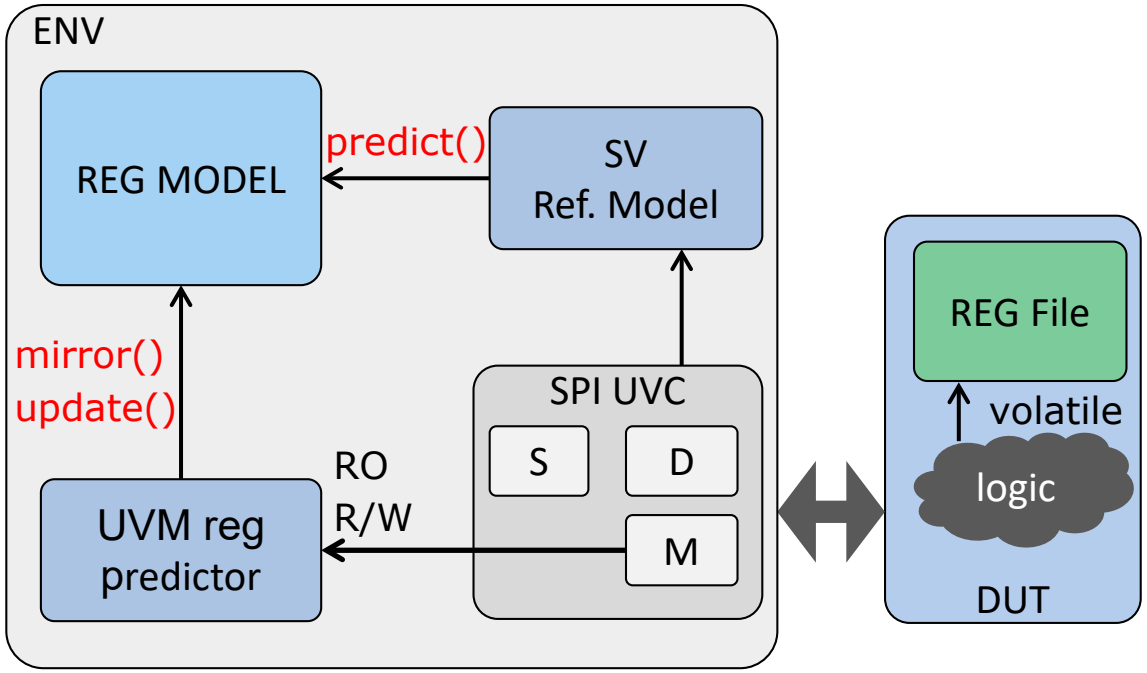
cadence **JasperGold® COVERAGE App - Proof Coverage Report**

Instance Name	Coverage	Excluded Covers
− uart_top	44.43% (359/808)	160
wb_interface	71.43% (25/35)	13
− regs	43.21% (334/773)	147
− transmitter	71.63% (101/141)	15
− fifo_tx	60.00% (21/35)	6
tfifo	0.00% (0/5)	0
i_uart_tfifo_checks	0.00% (0/0)	0
i_uart_transmitter_checks	0.00% (0/0)	0
i_uart_sync_flops	72.73% (8/11)	4
− receiver	66.39% (160/241)	40
− fifo_rx	29.17% (21/72)	22
rfifo	0.00% (0/5)	0
i_uart_rfifo_checks	0.00% (0/0)	0
i_uart_receiver_checks	0.00% (0/0)	0

Agenda

- Problematic
- Proposed Flow
- **Application on a Mixed-Signal Design**
  - Verification Split
  - Formal Verification Flow
  - **Simulation Reductions**
- COV DB Merging
- Conclusion

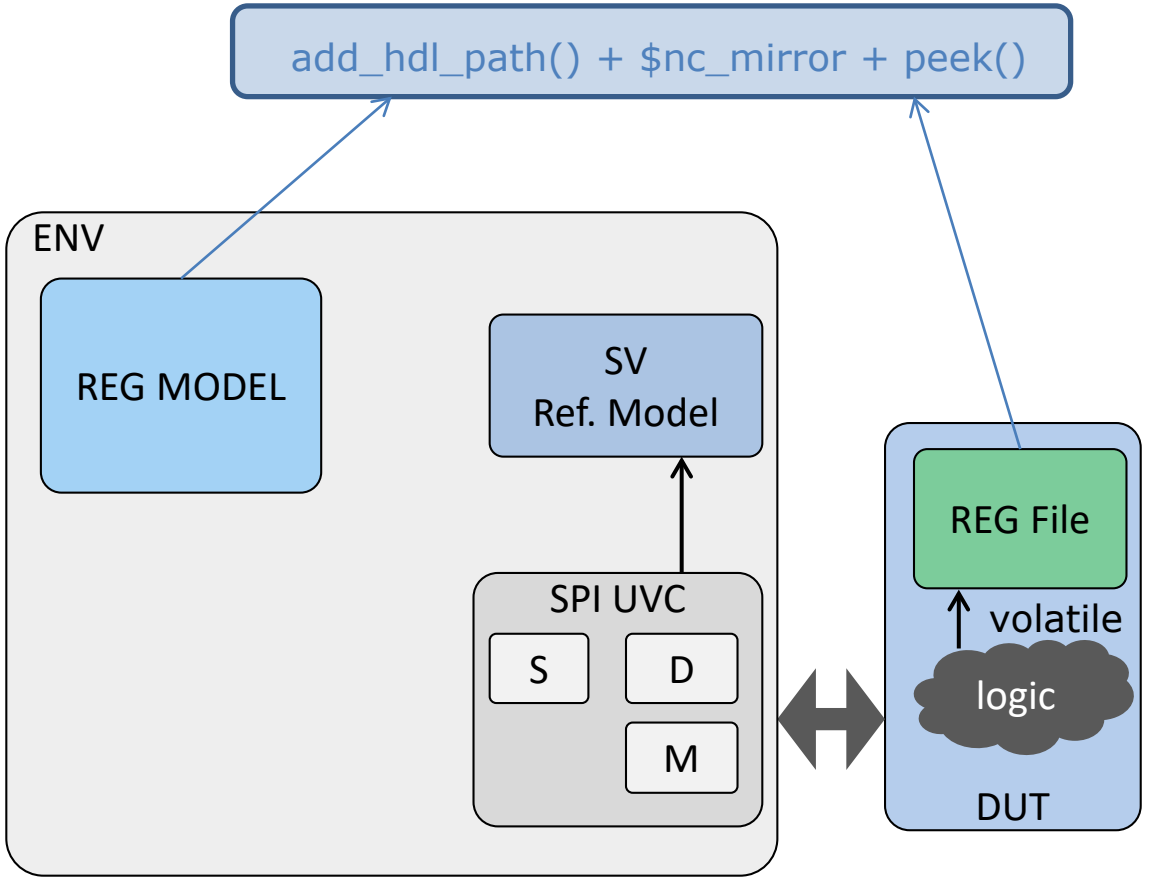
Simulation Reductions



Regmodel updating uses :

- UVM reg predictor : update(), mirror()
- SV ref model : predict()

Simulation Reductions



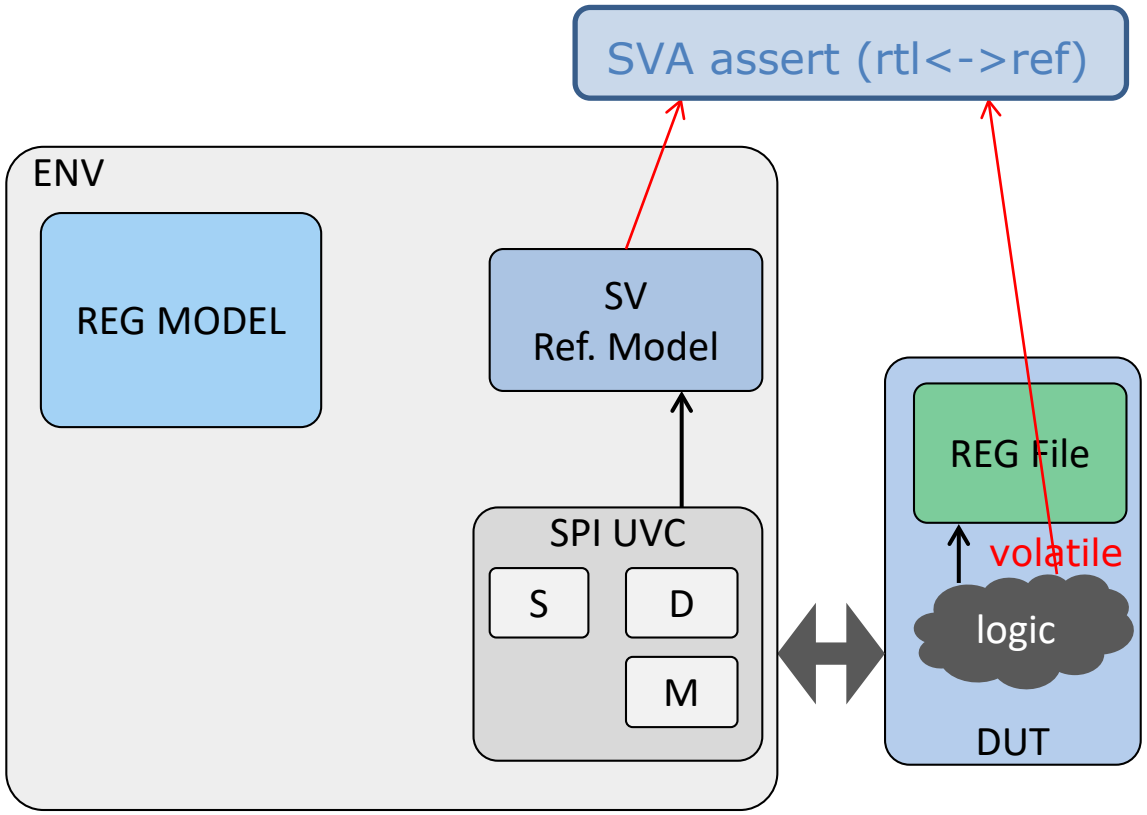
First remove :

- the UVM reg predictor : update(), mirror()
- the SV ref model : predict()

Next change the regmodel updating to :

- Hook the reg model directly to the HDL regs using add_hdl_path()
- Use $nc_mirror to detect HDL changes
- Upon a change use peek() to update the reg model

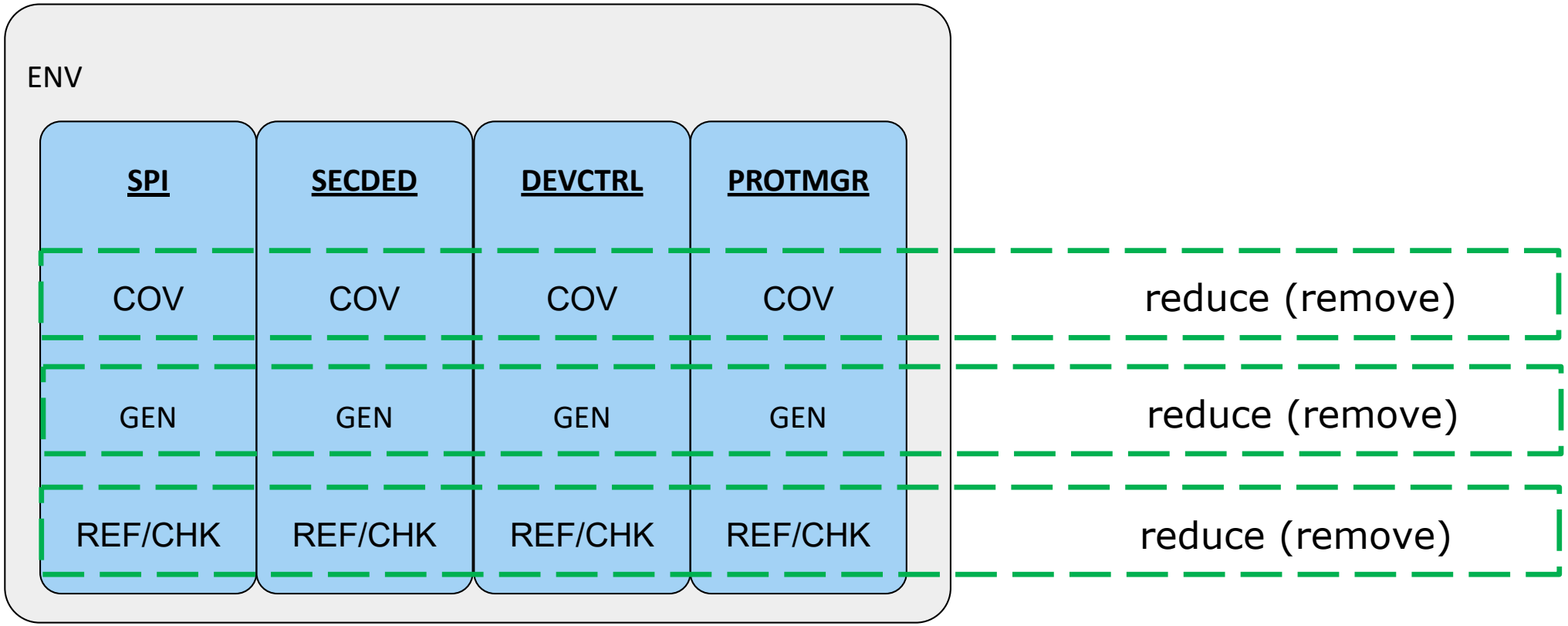
Simulation Reductions



- add SVA assertions to check
  - RTL volatiles (status, flags, ..)
  - versus SV ref model (former predict())
- SVA is bi-directional !
  - [RTL volatile] implies [SV ref]
  - [SV ref] implies [RTL volatile]

++ no more SPI reads needed to check status!
++ SVA is more precise! (quality ++)

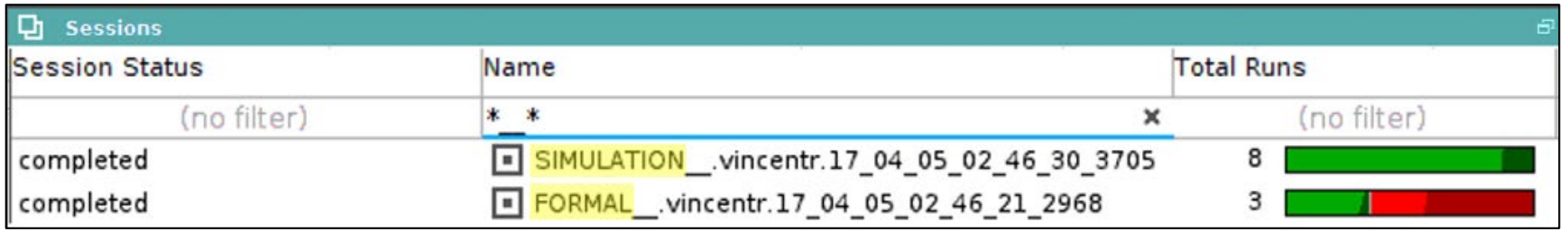
Simulation Reductions

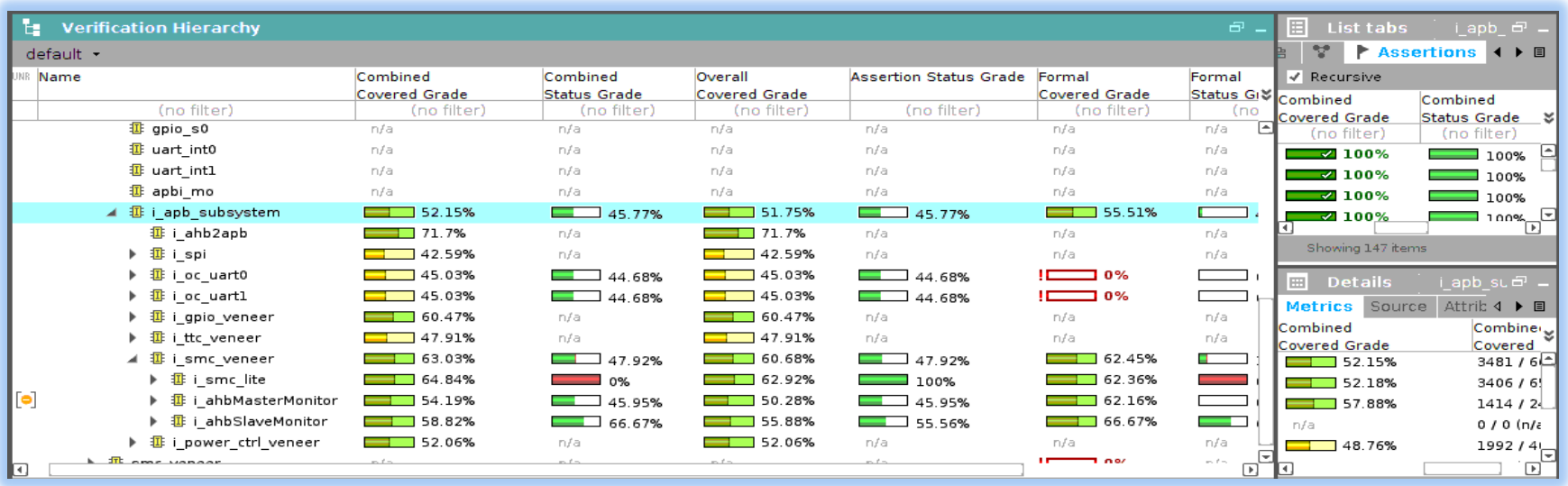


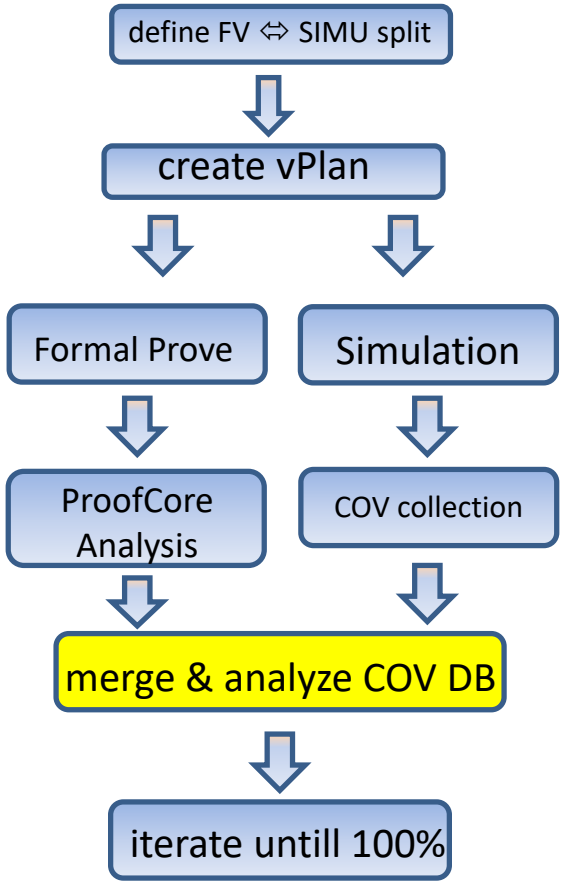
Agenda

- Problematic
- Proposed Flow
- Application on a Mixed-Signal Design
  - Verification Split
  - Formal Verification Flow
  - Simulation Reductions
- **COV DB Merging**
- Conclusion

COV DB Merging

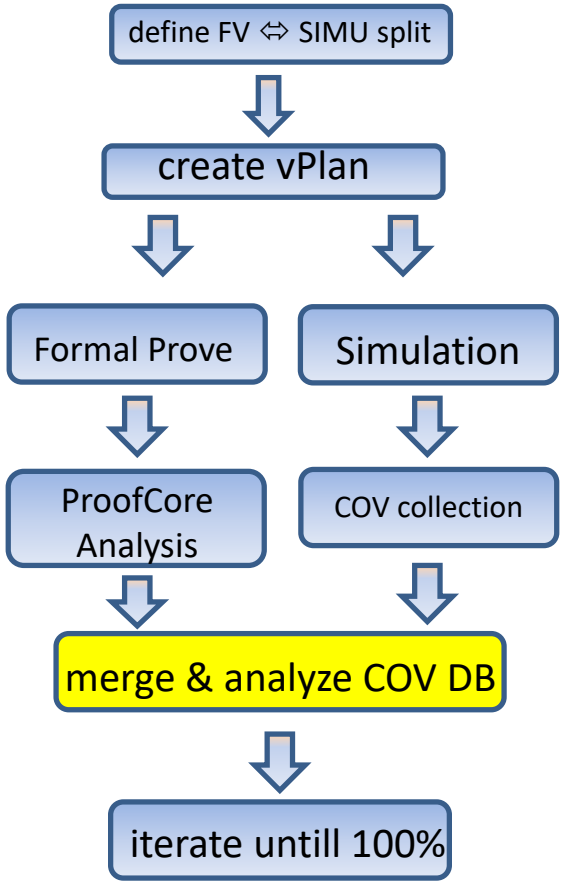






Confirm SIMU ⇔ FV split

Name	Simulation		Formal	
	Coverage	Checks	Coverage	Checks
	(no filter)	(no filter)	(no filter)	(no filter)
multiEngineMDV	53.03%	59.3%	76.86%	81.15%
1 Interfaces	56%	56%	76%	41.89%
1.1 UART	68.97%	68.97%	96.55%	89.29%
1.2 AHB	47.83%	47.83%	63.04%	13.04%
2 Memory Transactions	100%	50%	100%	0%
3 Assumption Validation	88.89%	88.89%	n/a	n/a
4 Registers	n/a	n/a	100%	41.18%
4.1 Jasper CSR (automatic vPlan)	n/a	n/a	100%	41.18%
4.1.1 Rx_Buffer_1	n/a	n/a	100%	100%
4.1.2 IER	n/a	n/a	100%	100%
4.1.3 IIR	n/a	n/a	100%	66.67%
4.1.4 LCR	n/a	n/a	100%	0%
4.1.5 LSR	n/a	n/a	100%	0%
4.1.6 MSR	n/a	n/a	100%	100%
5 Code Coverage	52.94%	n/a	73.12%	n/a
5.1 Jasper UNR	100%	n/a	n/a	n/a
5.2 uart	61.29%	n/a	83.29%	n/a
5.3 apb_subsystem	51.9%	n/a	62.26%	n/a
6 Integration Testing	n/a	n/a	91.84%	91.84%
6.1 Jasper CONN	n/a	n/a	91.84%	91.84%



Agenda

- Problematic
- Proposed Flow
- Application on a Mixed-Signal Design
  - Verification Split
  - Formal Verification Flow
  - Simulation Reductions
- COV DB Merging
- **Conclusion**

Conclusion

- bridges the gap between FV and SIMU
- enables a structural analysis
- enables a confirmation of FV ⇔ SIMU split

- we apply an Optimal Mixture of FV & SIMU
- over time, we get a Quality Increase (formal proof)
- over time, we get an Efficiency Increase (simu reductions)

Questions

Thank you for your attention!