IMMS

Institut für Mikroelektronik- und Mechatronik-Systeme gemeinnützige GmbH

Before It Hits the Chip: A Gentle Introduction to Mixed-Signal IC Verification

Martin Grabmann

USES2, Online Workshop #3, 15.09.2025

Sites in Thüringen, Germany

Institute headquarters
next to Technische Universität
Ilmenau

Branch office in **Erfurt**
in the centre of Thüringen's microelectronics industry

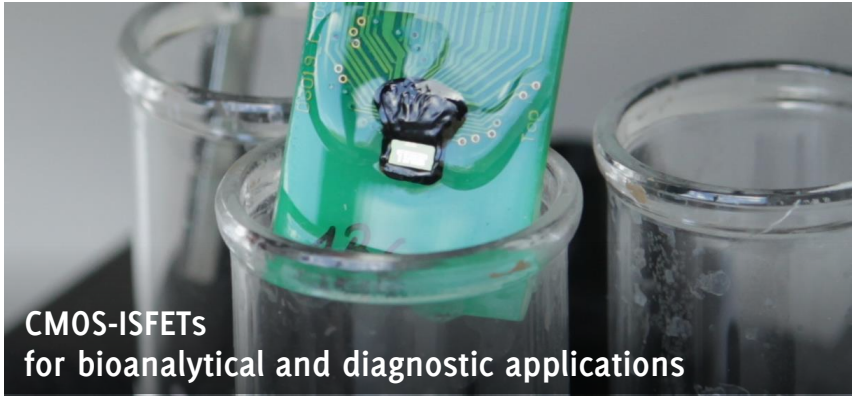
Microelectronics team at IMMS: 21 people (as per 01.07.2025)

- design of complex mixed-signal ICs
- services / specialists for:

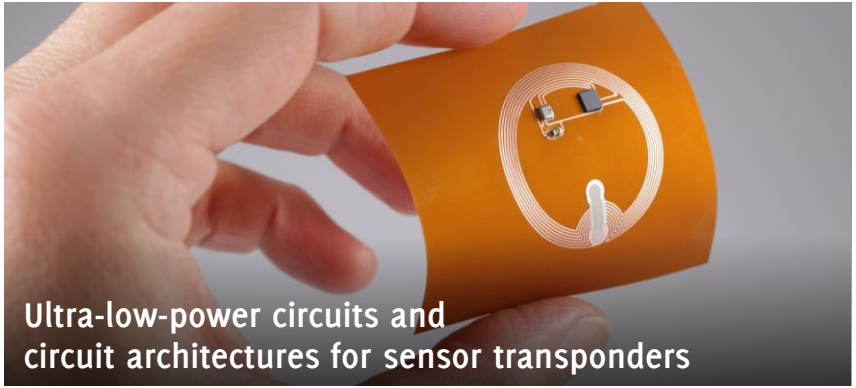
  analog IC design, digital IC design, mixed-signal verification, IC layout, design methodology, EDA support, embedded firmware
- approx. 10 ASICs per year for R&D for and with industrial partners
- approx. 15 students closely involved in project work

Our research in IC design

Optical sensors for in-vitro diagnostics based on SPADs and lock-in pixels



CMOS-ISFETs for bioanalytical and diagnostic applications



Ultra-low-power circuits and circuit architectures for sensor transponders

Design methods to make the complexity of IC design manageable and safe

Motivation: Why do we verify our IC designs?

Motivation: Some famous bugs

- Intel Pentium FDIV floating-point division bug (1994)
  - x - (x/y) * y = 0? Not true for x = 4195835.0 and y = 3145727.0
  - Missing entries in lookup table for division algorithm
  - Recall cost: 475 million $

Economic Loss

- Therac-25 computer-controlled radiation therapy machine (1985–1987)
  - Software race conditions led to lethal overdoses during radiation of cancer patients.
  - 6 overexposure incidents, 3 people died

Safety Issue

- Intel Meltdown/Spectre security flaws in micro processors (2017)
  - Access protected memory (leak internal information) as part of the out-of-order execution
  - Software mitigations necessary, at the cost of reduced performance

Security Issue

Source: https://www5.in.tum.de/persons/huckle/bugse.html

Motivation: More bugs / flaws – also in Mixed-Signal ICs

- Mixed-signal integrated circuit = analog and digital circuitry integrated on single semiconductor die
- e.g.

TEXAS INSTRUMENTS

MSP430F5438A, MSP430F5437A, MSP430F5436A
MSP430F5435A, MSP430F5419A, MSP430F5418A
SLAS655H – JANUARY 2010 – REVISED MAY 2021

MSP430F543xA, MSP430F541xA Mixed-Signal Microcontrollers

25-MHz MCU with 256-KB flash, 16-KB SRAM, 12-bit ADC, DMA, UART/SPI/I2C, timer, HW multiplier

- MSP430F5438A device errata sheet
  - 46 functional advisories

    (exceptions of functional specification)

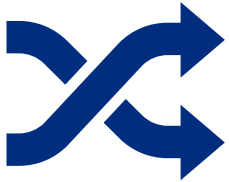
ADC27	***ADC Module***
Category	Functional
Function	Integral and differential non-linearity exceed specifications
Description	The ADC12_A integral and differential non-linearity may exceed the limits specified in the data sheet under the following conditions: - If the internal voltage reference generator is used and - If the reference voltage is not buffered off-chip and - If fADC12CLK > 2.7 MHz The non-linearity can be up to tens of LSBs. This is due to the internal reference buffer providing insufficient drive for the switched capacitor array of the ADC12_A.

➔ **IC could be useless for certain application**

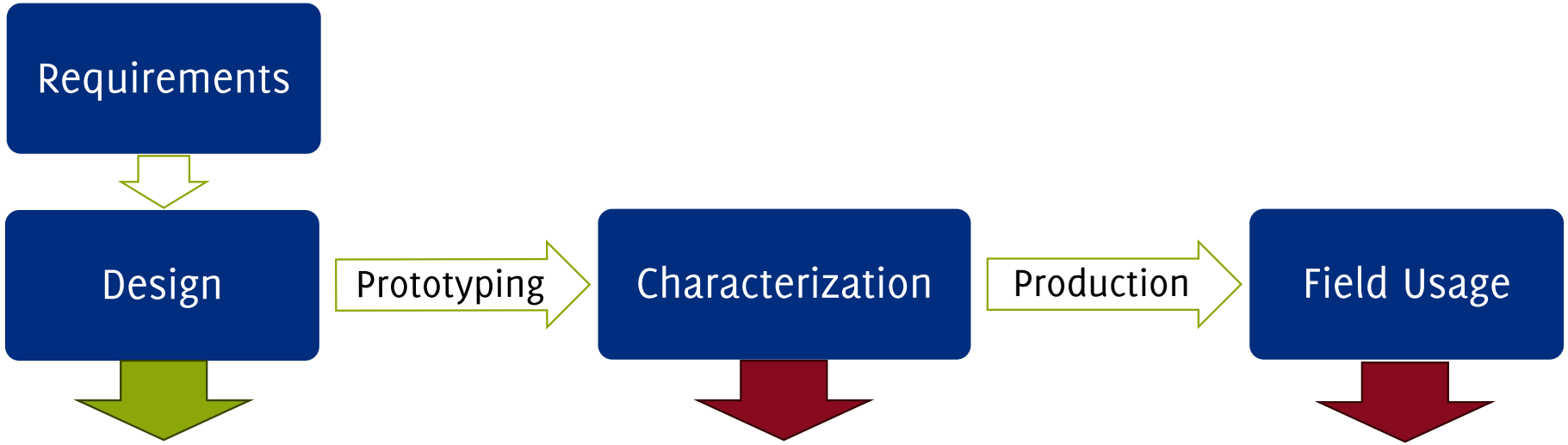
Sources:
Datasheet: https://www.ti.com/lit/gpn/msp430f5438a
Errata: https://www.ti.com/lit/pdf/slaz290

Outline

- Motivation: Why do we verify or IC designs?
  - Perspective of becoming a “verification expert”
- Methodologies
  - Verification vs Validation
  - Which methods exist to verify IC designs?
  - How to measure verification quality?
- Deeper dive into challenges in **mixed-signal IC verification**
  - What can go wrong and how to find those bugs **efficiently**
  - Mixed-signal **simulation**
  - Complexity handing and verification planning



Motivation: Consequences of Bugs



- Best to find bugs here for **first time right** designs!

- Bugs cause:
  - Re-designs
  - New prototype production
  - → Additional costs
  - → Product delays

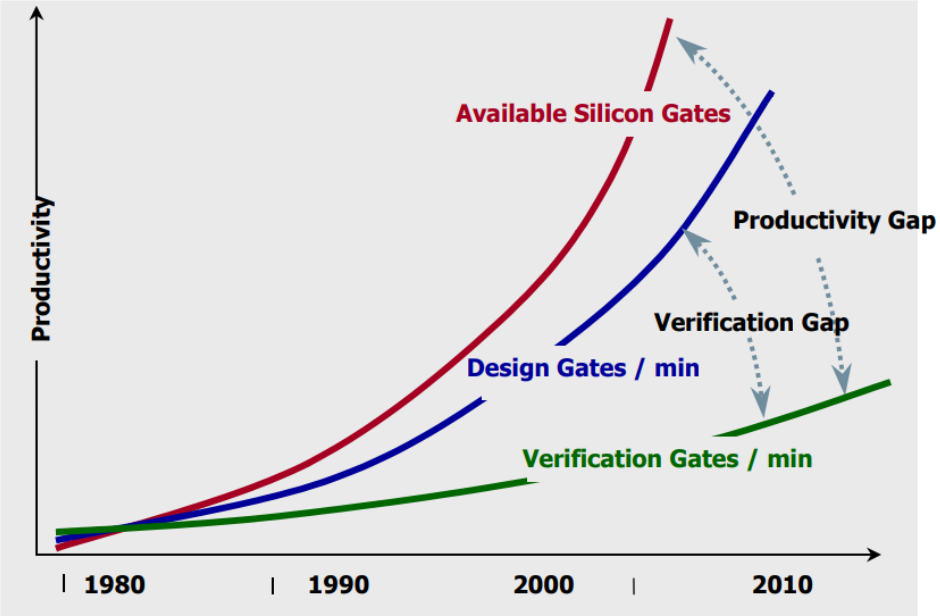
- Bugs cause:
  - Image loss
  - Accidents (automotive, medical,…)
  - Product recalls

Motivation: Design and Verification Gap

- IC complexity is still increasing
  - **Moore's Law (digital systems)** – still more or less valid
    - Intel Pentium (1993) - 3.1M transistors
    - Apple M3 Ultra (2025) - 184B transistors
  - **More than Moore (mixed-signal / heterogenous systems)**
    - Integrated sensors (photodiods, ISFETs, …)
    - Microsystems (mechanic, photonic, …)
    - RF integration (wireless, radar, …)
    - embedded non-volatile memory (eNVM)
    - Power electronics

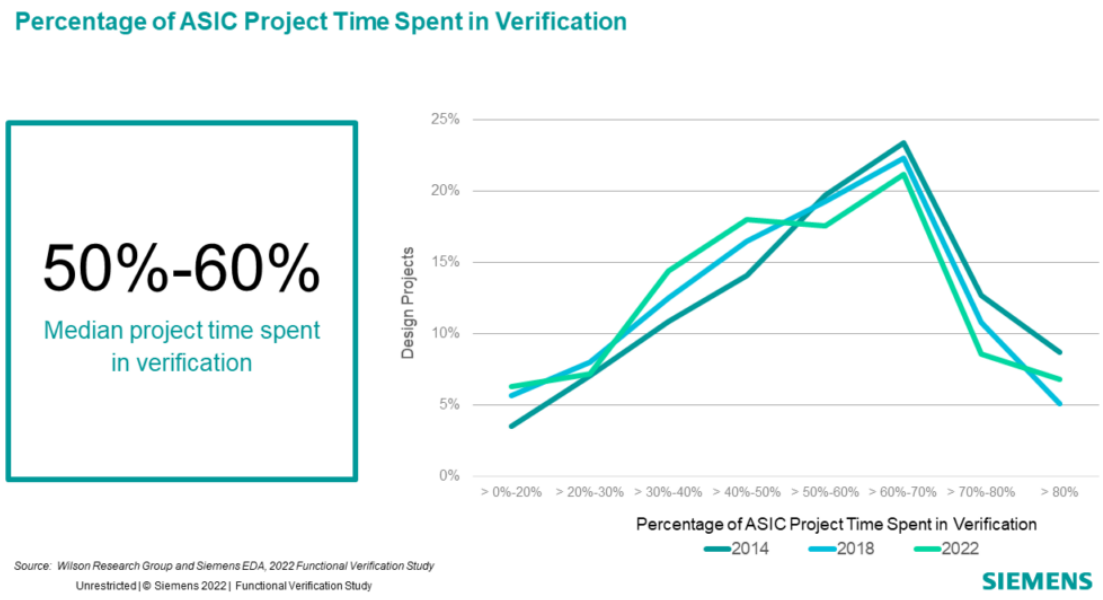
➔ Technology surpasses design

➔ Design surpasses verification



Source: L. Rizzatti, "High-Level Synthesis to Design in the New Millennium", IEEE Tech Forum, 2002. https://californiaconsultants.org/wp-content/uploads/2014/05/200205.pd

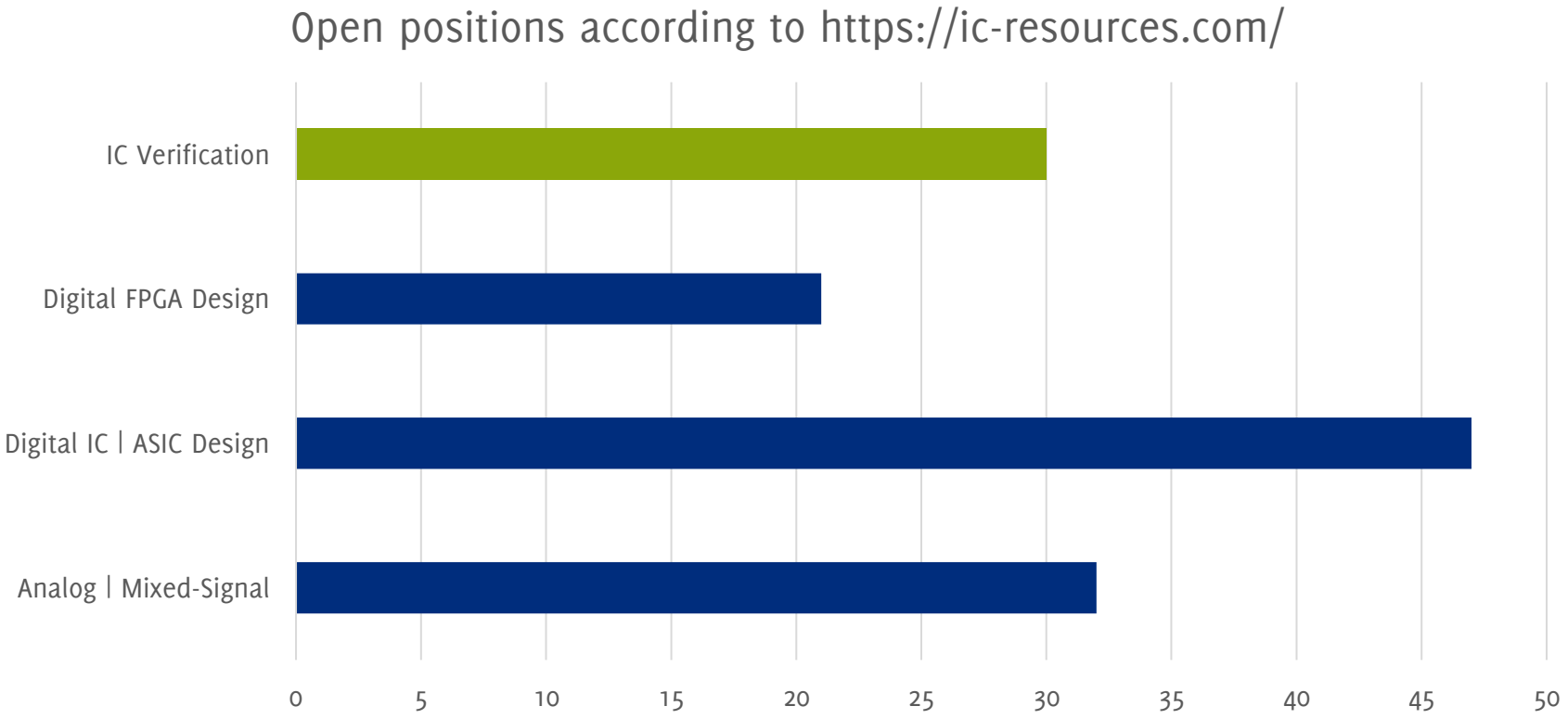
Motivation: More and More Time is spent in Verification



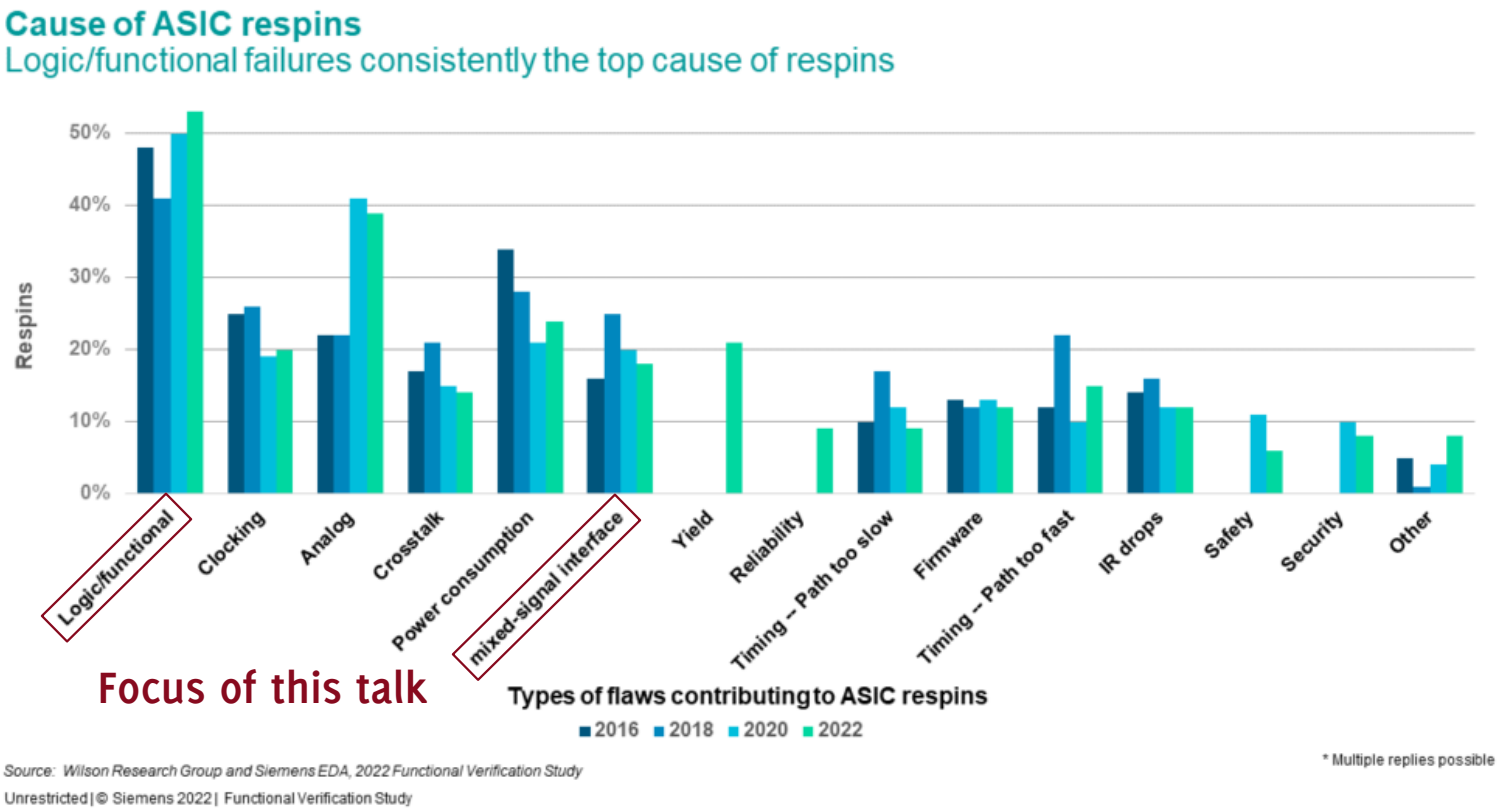
Source: https://blogs.sw.siemens.com/verificationhorizons/2022/12/12/part-8-the-2022-wilson-research-group-functional-verification-study/

- Teams need more verification power!
- Past: verification was designer's task
- Special skill set and expertise is required for efficient verification
  - system-level view
  - complexity handling and organization
  - tool handling and automation
- **Today: verification is specialized profession!**

Motivation: Open Positions



Where do bugs hide?



Source: https://blogs.sw.siemens.com/verificationhorizons/2023/01/09/part-12-the-2020-wilson-research-group-functional-verification-study-2/

Terms and Methodologies: What do we need to know?

Definitions: Verification vs Validation

- IEEE 1012-2016 - Standard for System and Software Verification and Validation

- **Verification:** “The process of providing objective evidence that the system, software, or hardware and its associated products conform to requirements (e.g., for correctness, completeness, consistency, and accuracy)..”

Are we building the product right?

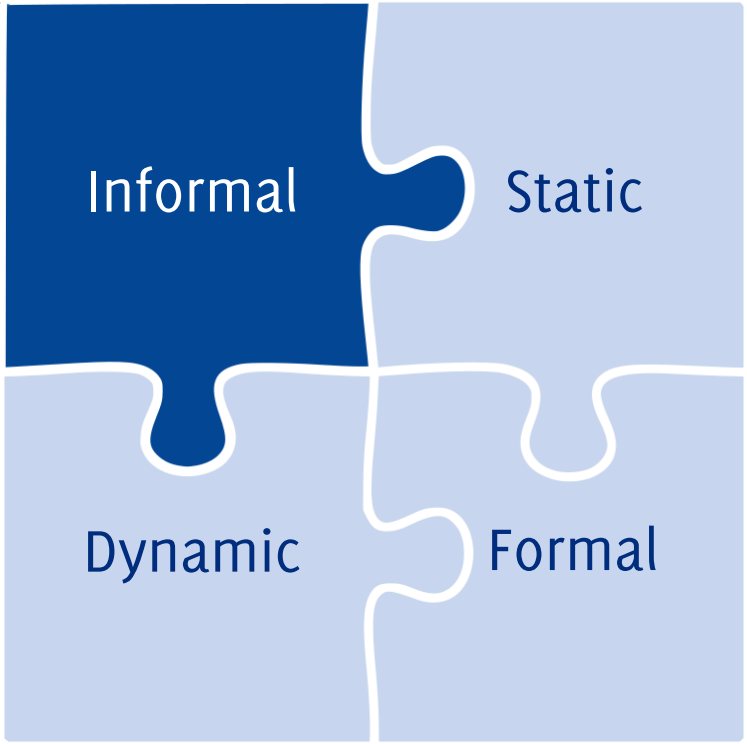
- **Validation:** “The process of providing evidence that the system, software, or hardware and its associated products satisfy requirements allocated to it […], solve the right problem […], and satisfy intended use and user needs.

Are we building the right product?

Verification methods: Overview

- **Reviews**

- Simulation
  - Directed testing
  - Constrained-Random
- Emulation



- Linting
- Coding Style
- Schematic Style

- Model checking
- Equivalence checking

Verification methods: Informal

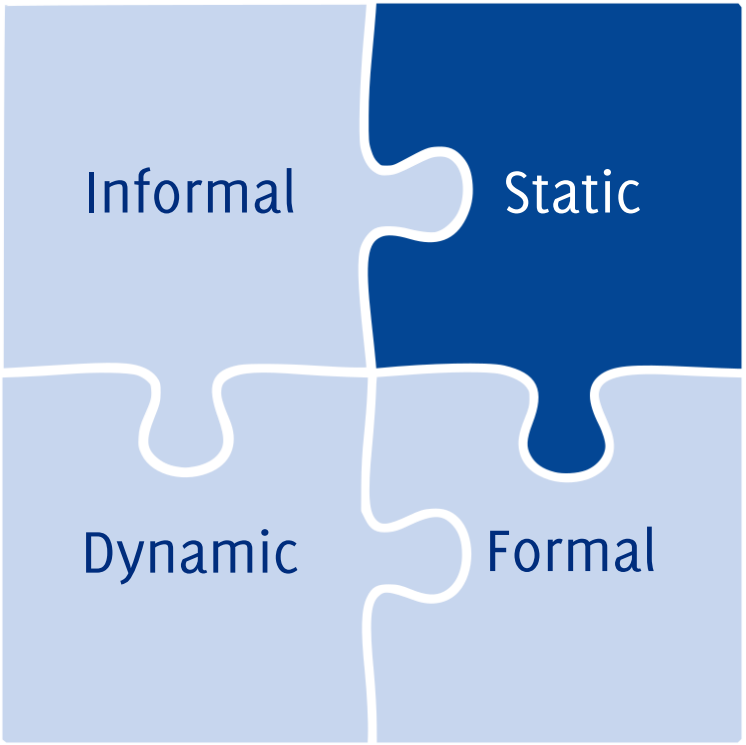
- Review meetings with colleagues
  - **"Two heads are better than one"** - „Vier Augen sehen mehr als zwei“
  - Talk to colleagues and explain what has been build

Cons	Pros
Time-consuming for complex designs	Design flaws, wrong logic or inconsistencies might be discovered early
Has to be repeated each time the design changes	Improves design quality, Supports knowledge sharing and learning
Hard to document	Time efficient method for checking best practices which are not yet automatically checkable

Verification methods: Overview

- Reviews

- Simulation
  - Directed testing
  - Constrained-Random
- Emulation



- **Linting**
- **Coding Style**
- **Schematic Style**

- Model checking
- Equivalence checking

Verification methods: Static

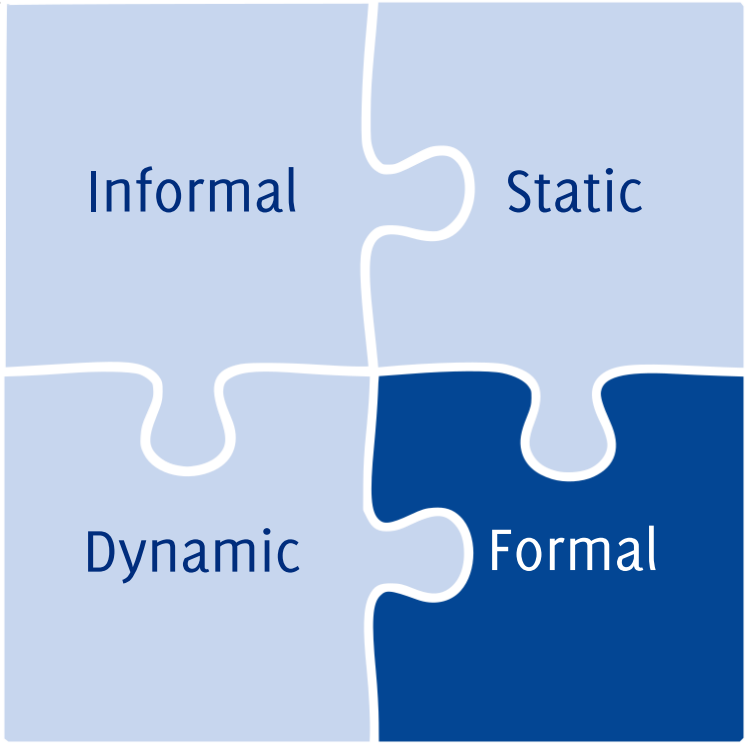
- Guidelines on module, instance, port and signal naming
  - Improved readability and maintainability
  - Prevent design flaws when integrating blocks on system level
  - Examples:
    - Low-active reset (RSTN) or high-active reset (RST)
    - Enable signals e.g. PU_EN (pull-up enable or pull-up enable negated?)

- Linting: Tool analysis of code for programmatic and stylistic problems.
  - Possible in VHDL, Verilog, …
- Design Rule Checking (DRC)
  - Layout is checked by rules defined by silicon foundry

Verification methods: Overview

- Reviews

- Linting
- Coding Style
- Schematic Style

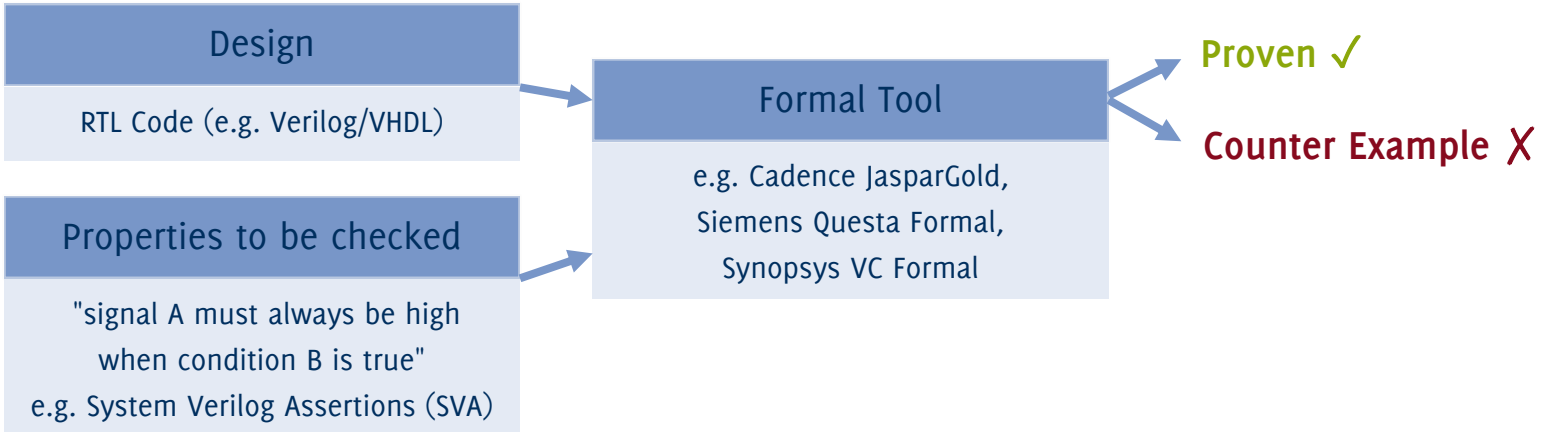


- Simulation
  - Directed testing
  - Constrained-Random
- Emulation

- **Model checking**
- **Equivalence checking**

Verification methods: Formal

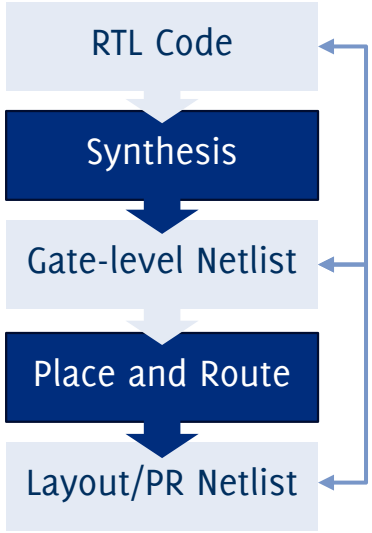
- Model Checking: Mathematical method used to prove that a design full fills properties under all conditions

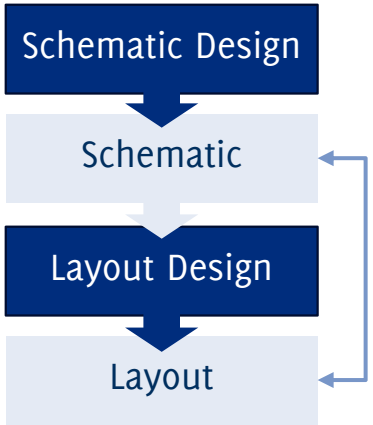


Pros	Cons
Systematic method, covers all inputs / states	Properties must be correctly written be engineer
No testbench / stimuli must be written by hand	Complexity: Hard to scale for very large designs
	State space explosion for deep sequential logic

Verification methods: Formal

- Equivalence Checking: Prove that different design views are equivalent



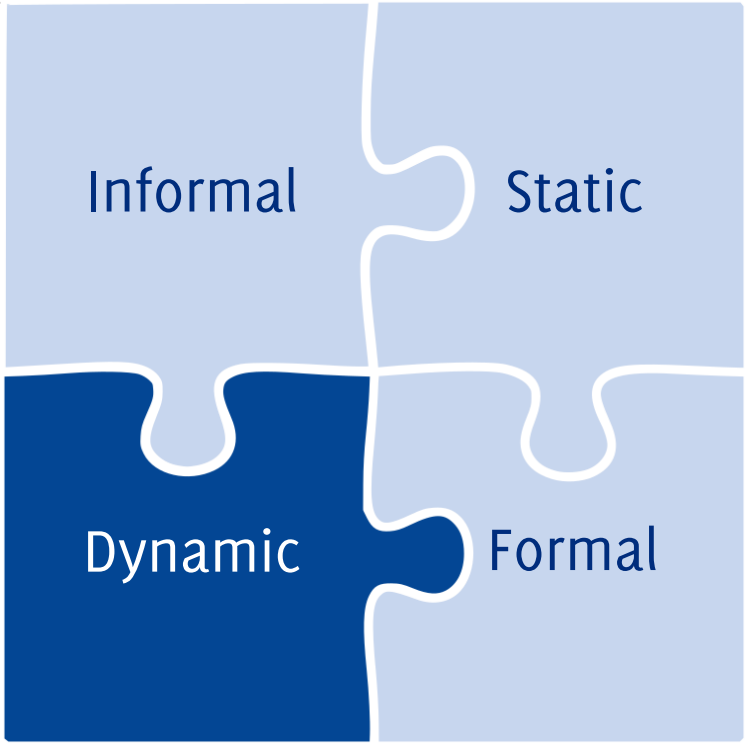


- Digital: Logic Equivalence Check (LEC)
  - Compare RTL code with design after synthesis or place-and-route
- Analog: Layout-vs-Schematic (LVS)
  - Extract netlist from layout and compare it with schematic netlist

Verification methods: Overview

- Reviews

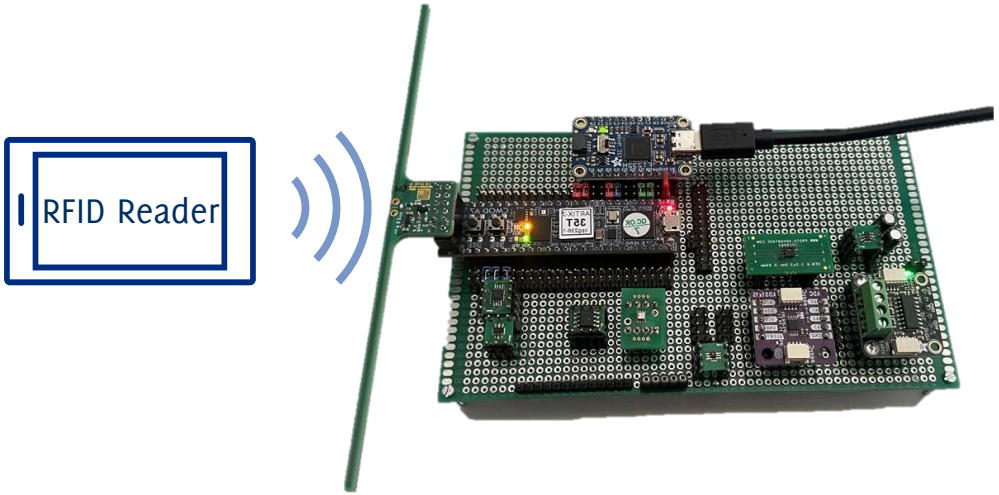
- **Simulation**
  - **Directed testing**
  - **Constrained-Random**
- **Emulation**



- Linting
- Coding Style
- Schematic Style

- Model checking
- Equivalence checking

Verification methods: Dynamic

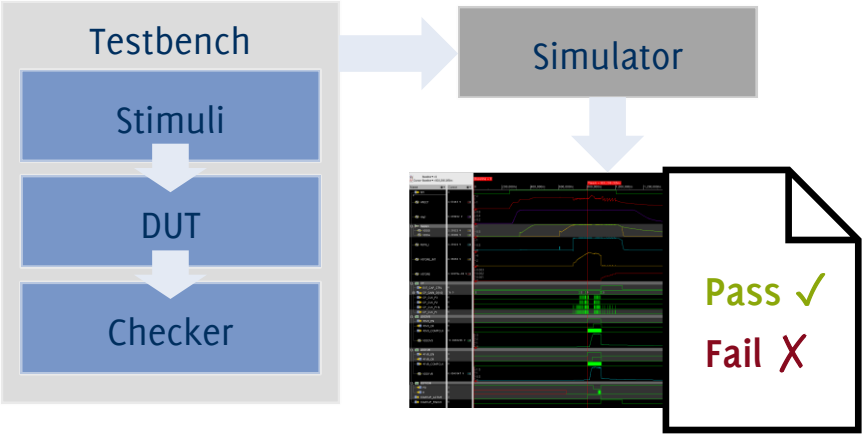


FPGA prototype of RFID tag with discrete RF frontend, antenna and external peripherals

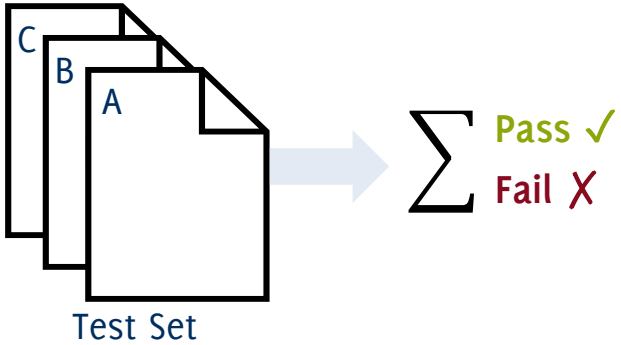
- Emulation
  - Map the design into specialized hardware e.g. on FPGA (field programable gate array)
  - Run test almost in real-time
  - Interface to other hardware is tested
  - Only feasible for digital design
  - Complex debugging

Verification methods: Dynamic

- Simulation:
  - Execute transient simulation with simulator software
  - Check waveforms manually by “eye”
    - Must be repeated each time the design changes
    - Does not scale for hundreds or thousands of tests
      - ➔ Not possible for complex designs ✗



  - Better approach: ✓
    - 1. Write checkers to built a **self-checking** testbench
    - 2. Perform **regression testing:**
      Rerun set of testbenches on regular base to check overall design state



Verification methods: Simulation

How to define stimuli?

- Directed Testing
  - Manually write test cases for specific scenarios
  - Ideal for verifying specific corner cases or complex operational sequences
  - Limited coverage

Example:
Write to a specific device register via I2C and check the expected action being executed.

- Constrained Random Testing
  - Automatically generate test cases based on defines constraints with randomization.
  - Can uncover hidden bugs.
  - Requires well-defines constraints and coverage model.
  - UVM (Universal Verification Methodology)

Example:
Test a memory controller by randomly generating read/write transactions with constraints on address and data patterns. Check protocol compliance.

How to build checkers? Option A) Plain SystemVerilog

- Procedural checks
  - Embedded in test sequence
  - e.g. Check correct reading of register value

```
initial begin
...
I2C.read(address, data);
if (data == expected_data) begin
  $display("Info: data match at time %t", $time);
end else begin
  $display("Error: data mismatch at time %t", $time);
  $error;
end
...
end
```

  - $error: Error is logged.
  - Other severities: $info, $warning, $fatal

- Concurrent checks
  - Run in parallel to the test sequence and observe certain properties
  - e.g. non-overlapping clock CLK_P1 and CLK_P2 are never allowed to be high at the same time

```
always @(CLK_P1, CLK_P2) begin
    if (CLK_P1 && CLK_P2) begin
        $display("Error: CLK_P1 and CLK_P2 are high
                  simultaneously at time %t", $time);
        $error;
    end
```

How to build checkers? Option B) Assertion Language

- Can be used for simulation and formal verification
- Property Specification Language (PSL): Language-independent (Verilog, SystemVerilog, VHDL, SystemC)
- **SystemVerilog Assertions (SVA):** Native to SystemVerilog

Immediate SVA:

- check at specific simulation time

```
initial begin
...
I2C.read(address, data);
assert (data == expected_data) else begin
  $display("Error: data mismatch at time %t", $time);
  $error;
end
...
end
```

Concurrent SVA:

- monitor behavior across multiple cycles
- special temporal logic to define properties

```
property handshake;
  @(posedge clk) req |-> ##1 ack;
endproperty
assert property (handshake);
```

= Every rising edge of clk it is checked, that:

if req is 1, exactly 1 clock cycle later ack must be 1 as well

Coverage Metrics: Overview

- How to measure verification progress?
- When am I done with verification?
- **Coverage metrics measure which parts of the specification or DUT have been verified.**
- Explicit: **“Functional Coverage”**
  - Define by hand which cover points / scenarios must be covered
- Implicit: **“Code Coverage”**
  - Extract cover points from RTL code
  - Block Coverage: Checks execution of each each **code block** (e.g., if, case, always).
  - Expression Coverage: Checks if all **boolean expressions** (e.g., a && b, x || y) are evaluate to both true and false.
  - Toggle Coverage: Check whether **each bit of a signal toggled** from 0 to 1 and vice versa.
  - FSM Coverage: Ensures that all **states and transitions** in a state machine are visited.

Coverage Metrics: Different Code Coverage Metrics

Example: FSM Coverage

SystemVerilog code snippet:

```
enum {EEPROM_IDLE, WAIT_FOR_LDO, EEPROM_STARTING,
EEPROM_RUNNING, EEPROM_PRE_POWERDOWN, EEPROM_POWERDOWN}
eeprom_state;
always_ff @(posedge CLK or posedge RST_ASYNC_I)
begin
    …
    EEPROM_STARTING:
    begin
        EEPROM_STANDBY <= 0;
        if (startup_timer < eeprom_startup_cycles - 1)
        begin
            startup_timer  <= startup_timer + 1;
            eeprom_running <= 0;
        end
        else begin
            eeprom_running  <= 1;
            eeprom_state  <= EEPROM_RUNNING;
            startup_timer <= 0;
        end
        if (ENERGY_GOOD == 1'b0)
        begin
            eeprom_state <= EEPROM_IDLE;
            eeprom_running  <= 0;
        end
    end
```

not covered!

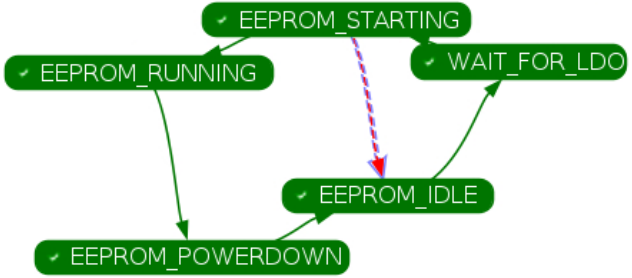
Cadence IMC Tool:

Transition Covered Grade: 83.33% | State Covered Grade: 100%

List Tabs | eeprom_state

States | Transitions & Arc Sources

Ex	UNR	Index	From State Name	To State Name	Score
		(no filter)	(no filter)	(no filter)	(no filter)
		0	EEPROM_IDLE	WAIT_FOR_LDO	121
		1	WAIT_FOR_LDO	EEPROM_STARTING	121
		2	EEPROM_STARTING	EEPROM_RUNNING	119
		3	EEPROM_STARTING	EEPROM_IDLE	0
		4	EEPROM_RUNNING	EEPROM_POWERDOWN	119
		5	EEPROM_POWERDOWN	EEPROM_IDLE	1



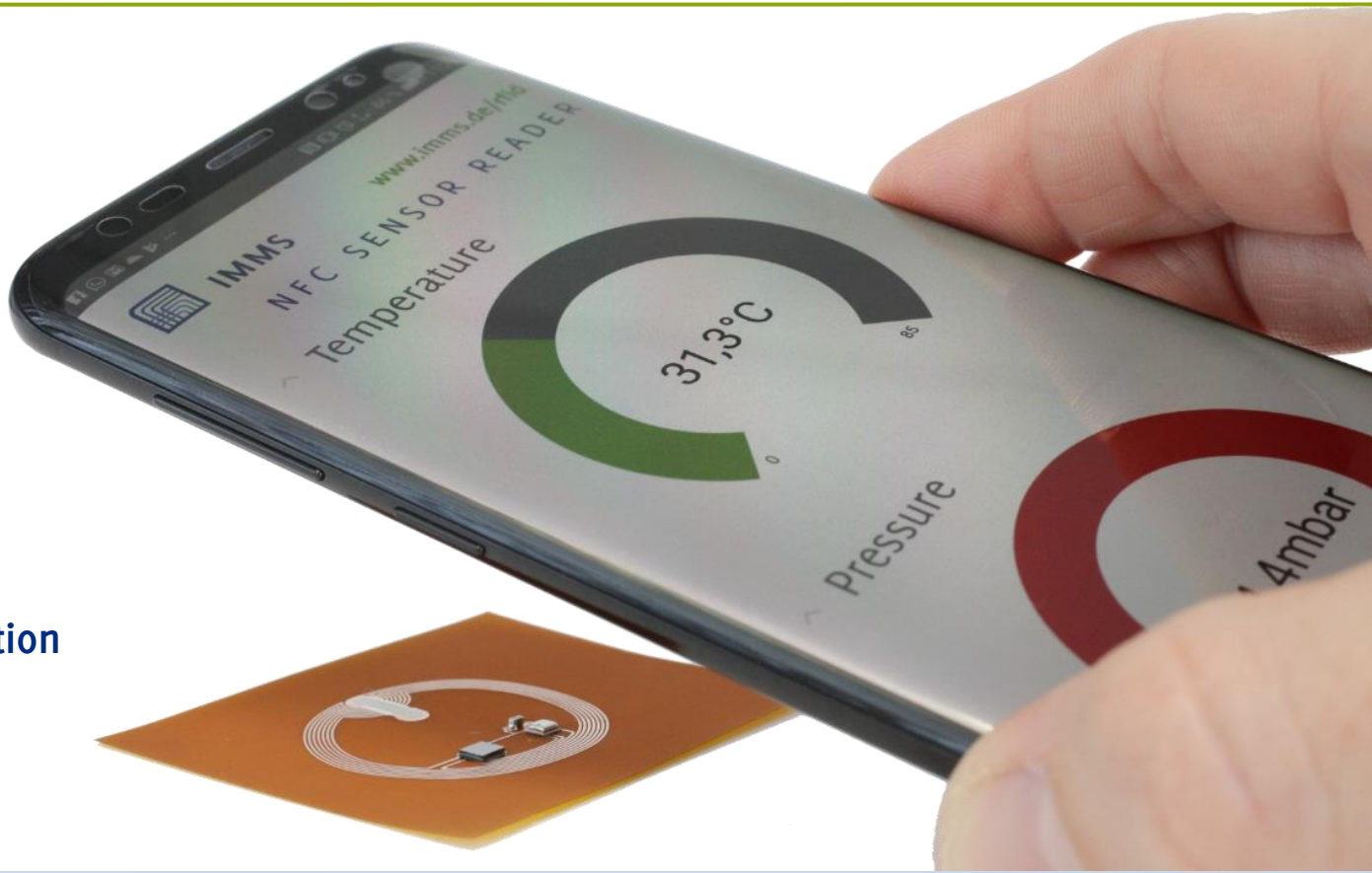
Coverage Metrics: Coverage in A/MS Verification

- Coverage is harder to define in analog and mixed-signal domains.
- Analog signals are continuous in value and time, unlike digital signals.
  - Discretization is required to define coverage metrics for analog behavior.
  - Sampling is possible, but fine-grained sampling is computationally expensive.
  - ➔ In practice, corner cases and scenarios are manually defined and verified using test stimuli and checkers.

- **What Is Typically Done?**
- **Functional Coverage**

  ➔ Derived from requirements and scenario definitions.
- **Structural Coverage**

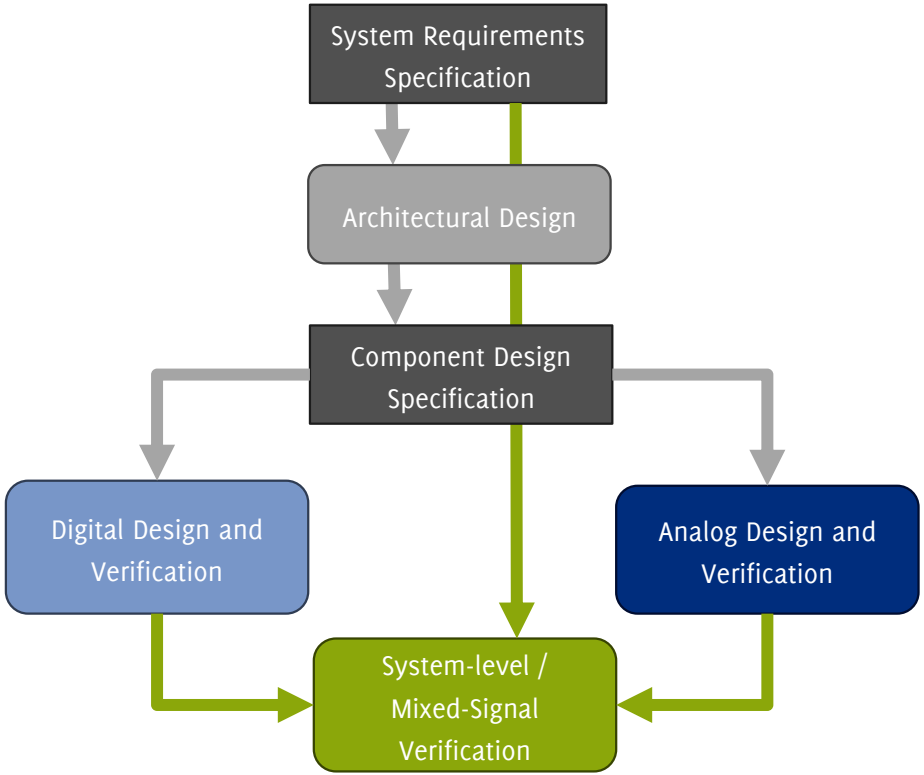
  ➔ e.g. ensure every signal between analog and digital domains is exercised.



Mixed-Signal IC Verification

Mixed-Signal: Challenges

- Understand both languages – of analog and digital designers
- Complexity Handling → Abstraction
- Various different tools require know-how
  - Analog Simulator
  - Digital Simulator
  - Connect Modules
- Various different design views
- …

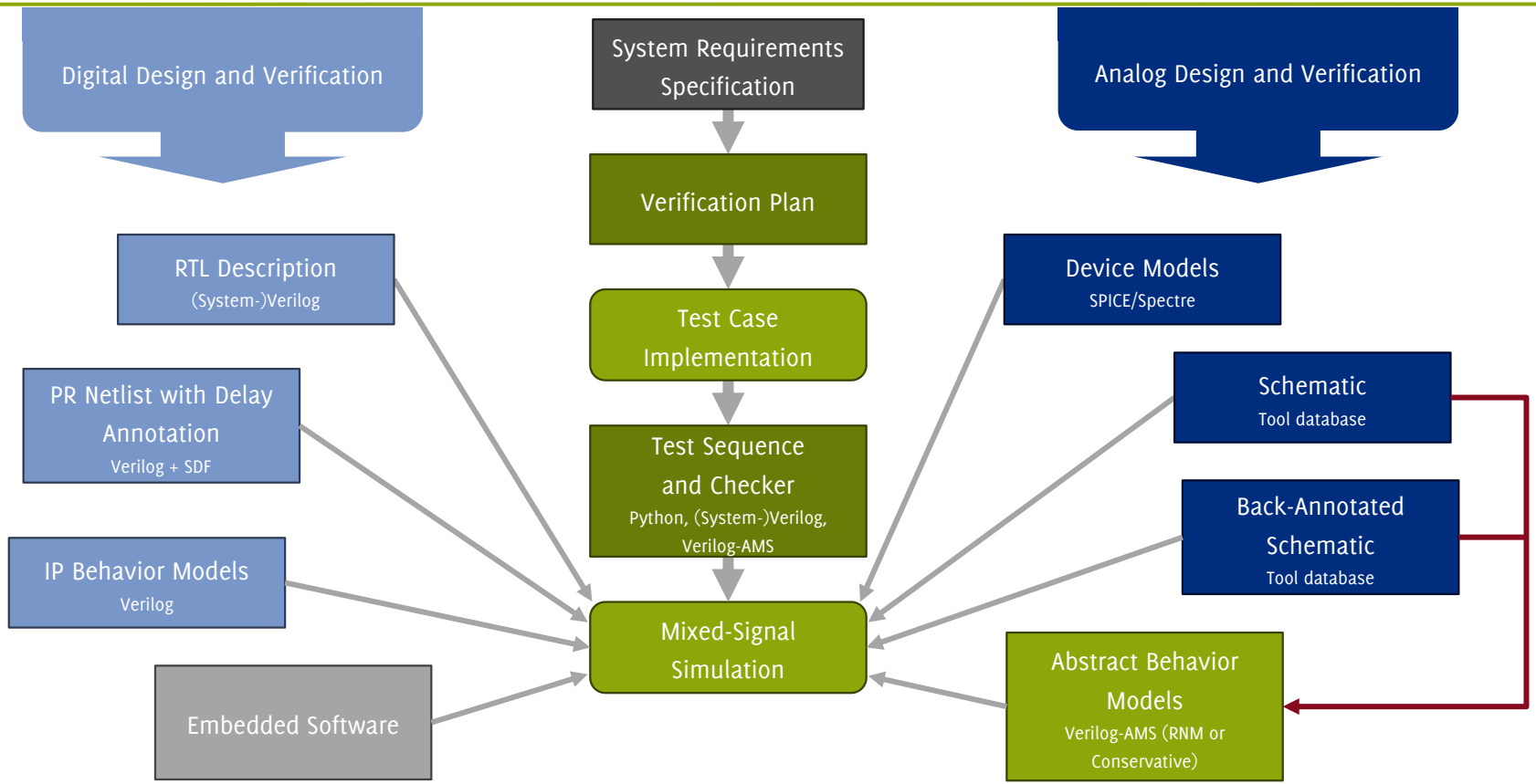


Mixed-Signal: What can go wrong?

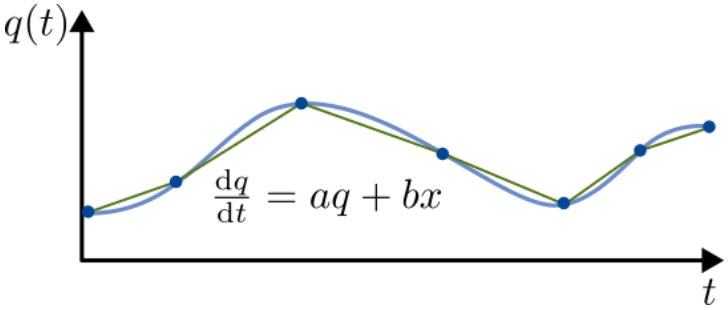
Test Objective	Potential Bugs
System Startup	• Reset signal is not generated • Short circuit • Wrong initial register values • Wrong operating point
D→A or A→D Control Interfaces and Glue logic	• Interchanged signals • Wrong polarity • Wrong bus coding (OneHot, Thermometer,…) • Missing Level-Shifters • **Wrong functionality on either side**
D→A or A→D Data Interfaces	• Sampling at wrong clock edge • Glitching signals
Embedded MCU	• Bug in ROM Software • Wrong Hardware/Software interface

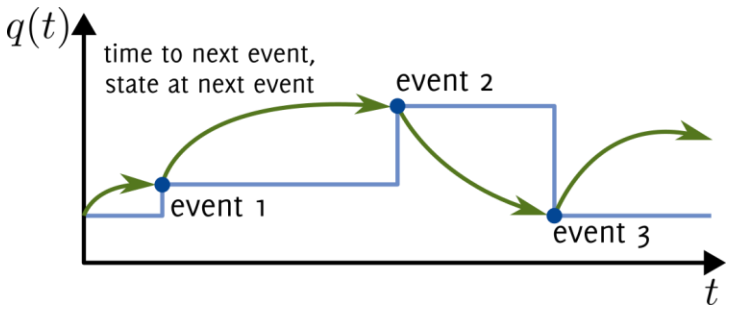
ALL of these could make the whole IC being useless!

Where are the Challenges?



Simulation Methods (1)





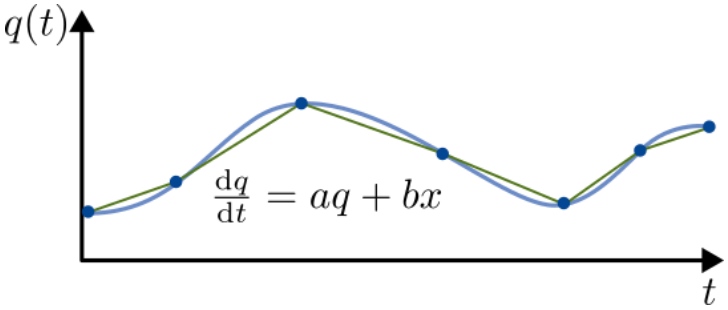
Continuous Time "Analog" Simulation

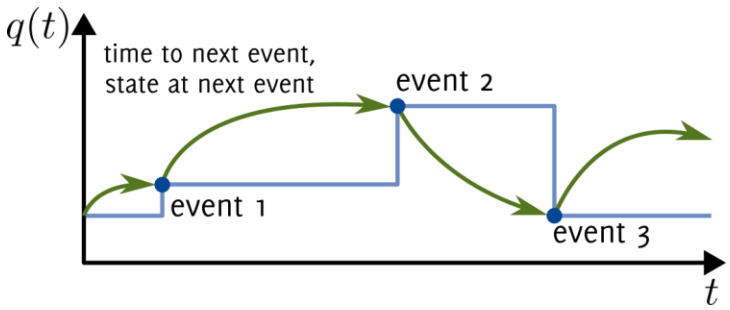
- Model: State is changing continuously according to differential equation
- Simulation: Numeric integration evaluates system at discrete time points
- If intervals are sufficiently small resulting curve appears continuous

Discrete Event "Digital" Simulation

- Model: State is changing only at discrete points in time
- Simulation: Time is advanced to the next scheduled event

Simulation Methods (2)





Continuous Time "Analog" Simulation

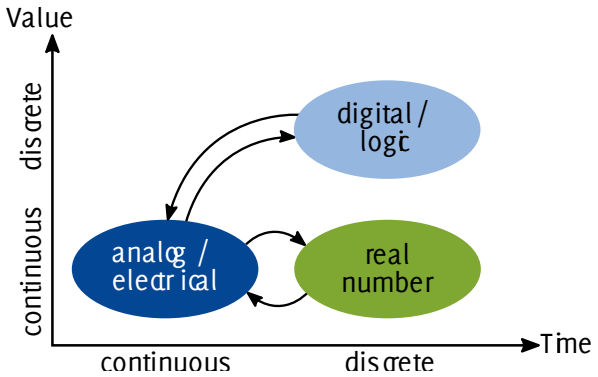
- Typical implemented by SPICE algorithm
  - 1. Formulate equation modified node analysis
  - 2. Linearize
  - 3. Numerical integration
- "Conservative" – Kirchhof's Laws apply
- Voltage and current domain

Discrete Event "Digital" Simulation

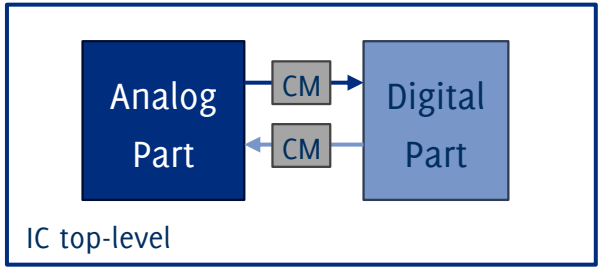
- Typical Logic Domain
- Real Valued Datatypes
  - Represent either voltage or current modeling
  - Non conservative – only allowed if feedback can be ignored!

Mixed-Signal Simulation

- Co-Simulation of SPICE-level and DES simulation
- Coupling necessary to transfer signals between domains
- Two separate processes, that are synchronized at specific points in time

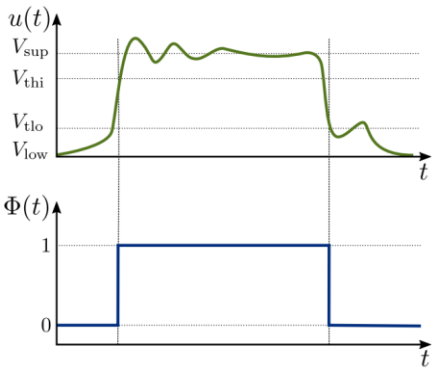


- **Option A)**
  Automatically let the simulator insert Connect Modules

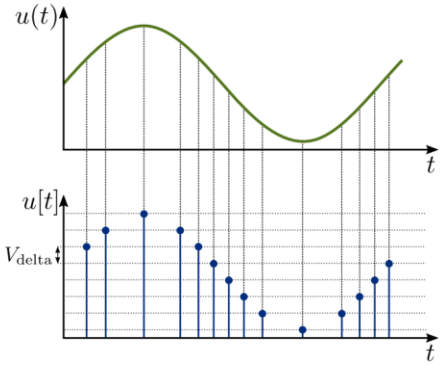


Connect Modules

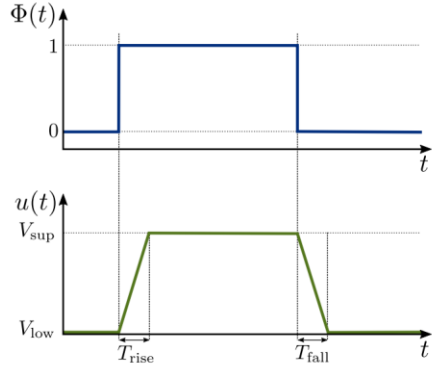
- Connect modules must be correctly configured!
  - E2L: threshold voltages (V_{thi}, V_{tlo}) for digital high and low level
  - E2R: voltage discretization (V_{delta})
    Only as small as needed!
  - L2E: supply high and low (V_{sup}, V_{low}) and rise and fall time (T_{rise}, T_{fall})
    Too steep edges cause convergence problems in analog domain!
  - R2E: rise and fall time (T_{rise}, T_{fall})



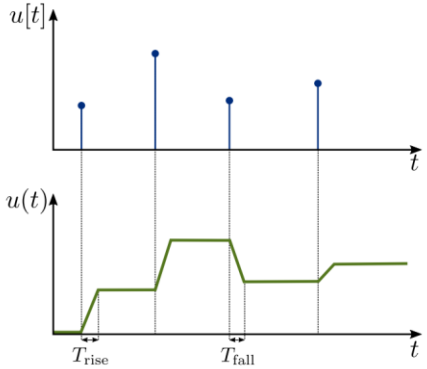
Electrical to Logic (E2L)



Electrical to Real (E2R)



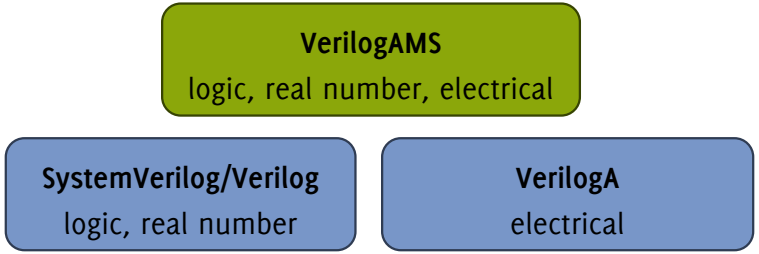
Logic to Electrical (L2E)

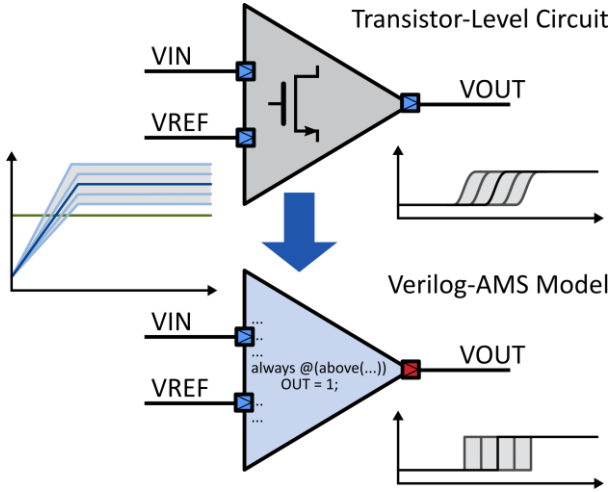


Real to Electrical (R2E)

Mixed-Signal Models

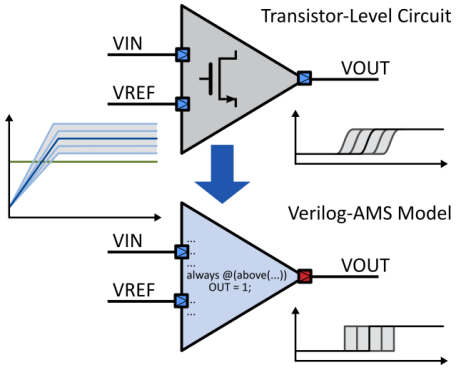
- **Option B) Write mixed-signal models**
- Different Verilog flavors allow different descriptions:

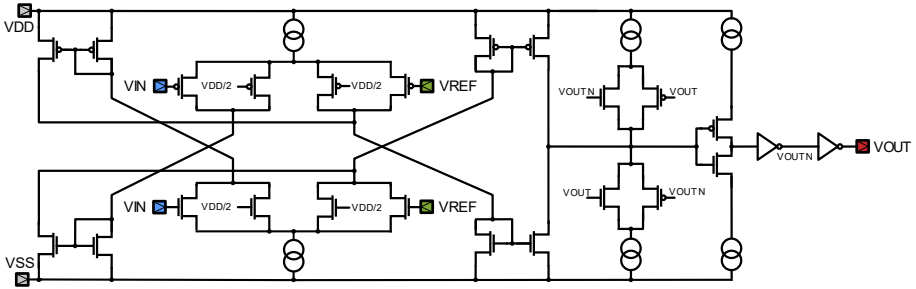




- VerilogAMS implements special functionality for simulation coupling
  (connect modules are also implemented with it)

Mixed-Signal Models: Comparator example





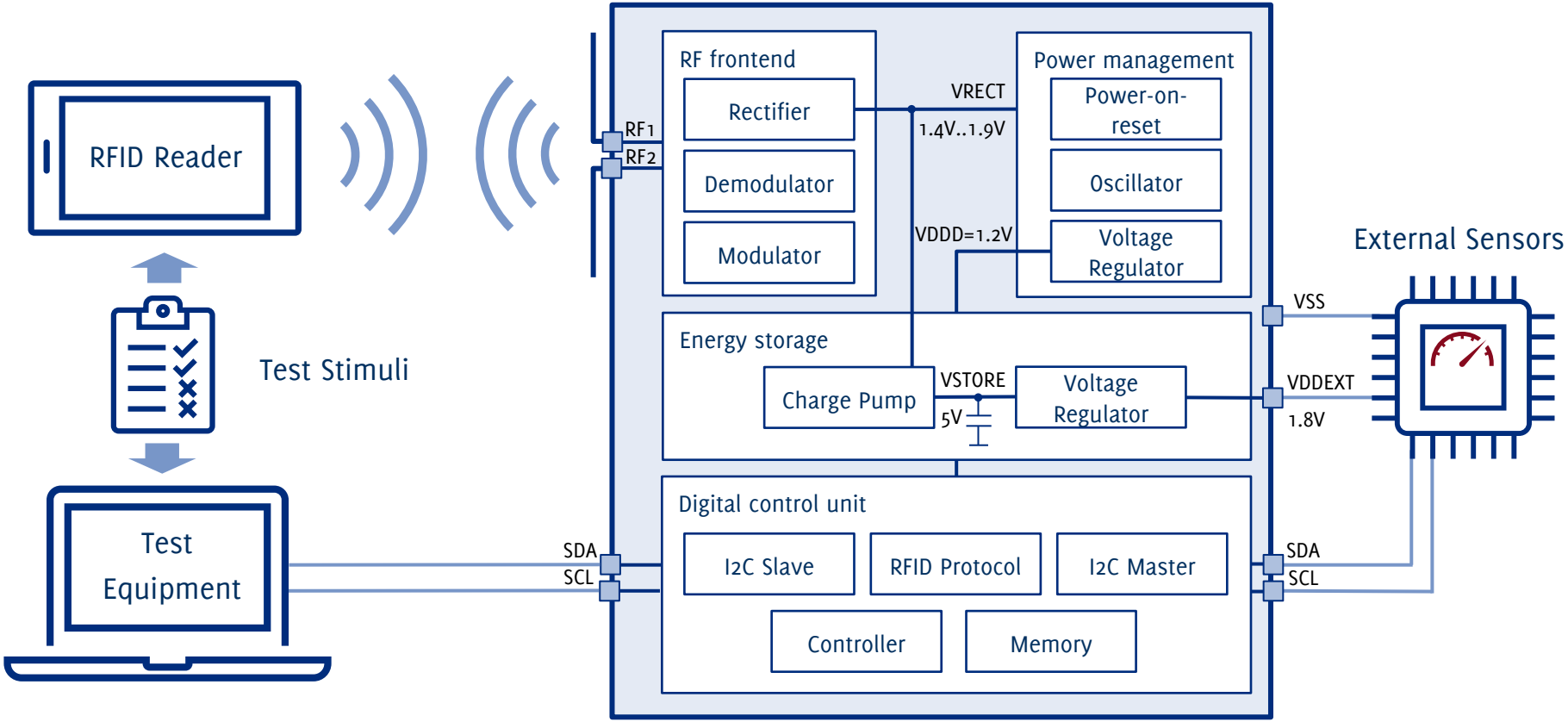

```
module comparator(VIN, VREF, VOUT, VDD, VSS);
    input VIN, VREF, VDD, VSS;
    electrical VIN, VREF, VDD, VSS;
    output VOUT;
    reg VOUT;

    parameter real off = 0.1;        // offset voltage [V]
    parameter real hyst = 0.05;      // hysteresis voltage [V]
    parameter real time_tol = 100p; // time tolerance [s]

    always @(above(V(VREF, VIN) - off, time_tol)) begin
        VOUT <= 1;
    end
    always @(above((off-hyst) - V(VREF, VIN), time_tol)) begin
        VOUT <= 0;
    end
endmodule
```

Behavior modeling to speedup simulations

- SPICE-level simulations are the bottleneck in mixed-signal simulations
  - ➔ Write abstract behavior models out of transistor-level schematics
  - ➔ Eliminate fast switching signals from analog domain

Subject	Examples	Abstraction Idea
Clock generation	• Oscillator • Phase-Locked-Loop	• Modell all clock signals in discrete domain
Switched-cap circuits	• DC/DC Converter • SC-Amplifier • Sample and Hold	• Modell all switch control signals in discrete domain • Idealize switches to eliminate analog time constants
Digital-assisted analog	• Pipeline ADC with digital calibration	
HF communication	• Demodulator • Mixer	• Model communication in baseband to avoid simulation of HF carrier

Case Study Example: RFID Sensor Tag



Case Study Example: RFID Sensor Tag – Verification IP

RFID Reader

SystemVerilog implementation of UHF RFID protocol (EPC Gen 2 – 204 pages)

Test Stimuli

SystemVerilog or Python sequence controlling RFID reader and test equipment

Test Equipment

SystemVerilog: I2C Master
VerilogAMS: Power Supplies, …

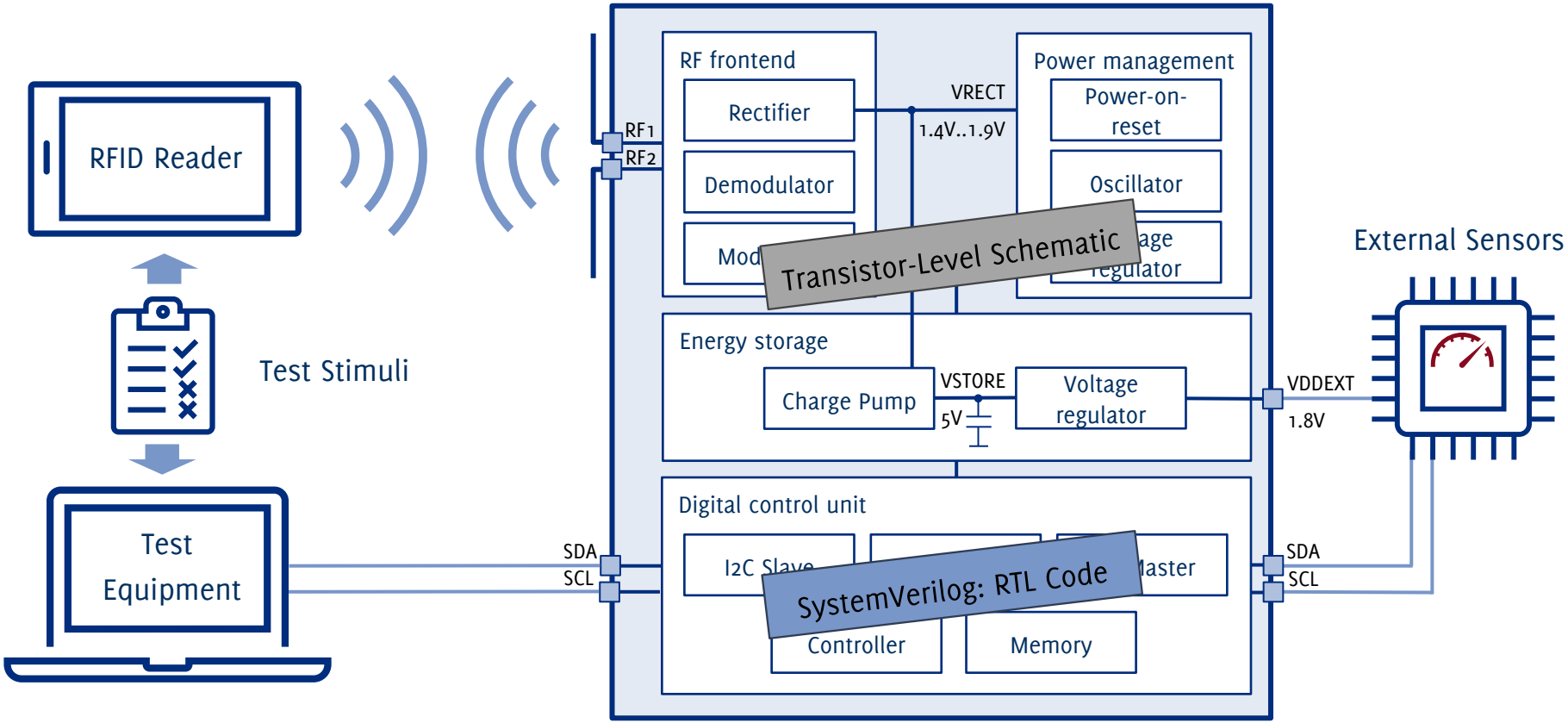
```
initial begin
    logger.info("Enable external supply voltage");
    `VRECT.set_voltage(1.75, 10e-6);

    @(posedge RSTN_I)
    logger.info("Oberserved posedge at RSTN pad.");

    check_analog_value_rel("D1036A_tb.VDDD", "Check
                            VDDD", "potential",  1.2, 0.1);

    logger.info("Read IC revision via I2C.");
    I2C.check_byte(7'h40, AHB_ADDRESS_MEM_CHIP_ID, 16'hA0);

    #(100us);
    logger.finish_simulation();
end
```

External Sensors

SystemVerilog or VerilogAMS Code modelling I2C sensor behavior

```
// Get the analog value of VDDD, by system task (specific to Cadence simulator)
$cds_get_analog_value("D1036A_tb.VDDD", potential);
```

Case Study Example: RFID Sensor Tag – Digital / Analog



Case Study Example: RFID Sensor Tag – Speed-up through behavior modeling (1)

1. Config: All analog on transistor-level schematic and digital on RTL

- 868 MHz sinus input with amplitude modulation at RF pins
- Simulation progress: ~ 1ms / 20h
- Time for startup: 1 – 3ms depending on input power

➔ Not feasible to cover different scenarios like that

2. Config: Use baseband models of RF frontend

- Eliminate 868 MHz by using baseband models of rectifier, modulator and demodulator
- Simulation progress: ~ 1ms / 15min

➔ Good to perform startup simulations

Case Study Example: RFID Sensor Tag – Speed-up through behavior modeling (1)

3. Config: Use oscillator model with digital clock output

- Replace 2 MHz oscillator by model, Don't calculate CLK signal in SPICE domain
- Simulation progress: ~ 10ms / 30min
- ➔ Perfect for checking interaction between controller and analog part

4. Config: Use charge pump model

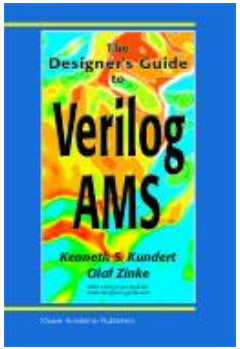
- Charge pump is circuit consisting of clocked switches and capacitances
- Slow simulation in SPICE domain
- Replace by model with digital clock input and RC equivalent circuit
- Simulation progress: ~ 1s / 30min
- ➔ Allows application scenarios with external sensors to be simulated

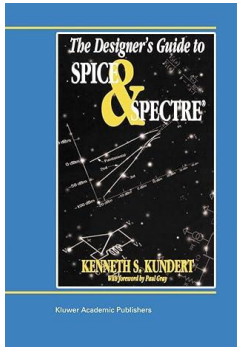
Case Study Example: RFID Sensor Tag

- Different **design configurations** for different **verification objectives**
- A well-structured **verification plan** is essential to manage complexity and ensure coverage
- **Partitioning the design** helps decide:
  - Which functions are best verified in **analog simulation**, **digital simulation**, or **mixed-signal-simulation**.
  - Which **config view** (e.g., ideal models vs. transistor-level) offers the most efficient coverage.
- Modeling accuracy and simulation speed must be balanced for effective verification.

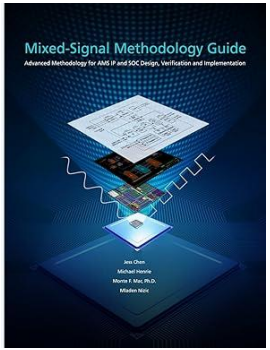
Literature

- **Kenneth S. Kundert:**
  - “Designer's Guide to Verilog-AMS”
  - “Designer's Guide to SPICE and Spectre”
  - https://designers-guide.org/
  - Practical guides to Analog and Mixed-Signal simulation and modeling





- Monte F. Mar, Jess Chen, Michael Henrie: **„Mixed-Signal Methodology Guide“**
  - Published by Cadence
  - Methodology for AMS IP and SoC design, verification and implementation
  - Practical examples for modeling and assertion-based verification



Acknowledgments

- TU Dresden: Requirements and methodologies for design of integrated circuits from industrial production perspective (summer semester 2023)
  - Advanced verification methods, Dr. Thomas Klotz, Bosch Sensortec
  - Pre- and Post-silicon Verification of Embedded Power ICs, Christoph Bukethal and Sebastian Simon, Infineon
- DVCLUB Edinburgh 2025: Mixed-Signal Verification
  - What is mixed signal verification and why you might want a career in it?,
    Peter Grove (Renesas) , Graeme Nunn (Cirrus Logic)

Summary

- Verification is a **challenging process**—without it, IC design would not function reliably.
- Multiple **verification methods** exist, each suited to different tasks (e.g., simulation, formal, emulation).
- It's a **creative discipline** requiring deep understanding of the entire design flow.
- Strong **communication skills** are essential to align across teams and domains.

Mixed-signal verification:

- Requires **understanding of analog, digital, and software** interactions.
- Presents unique challenges: achieving **100% coverage is nearly impossible**.
- Forces strategic decisions:

  → *What must be tested?*

  → *Where should effort be invested?*
- Balancing **available time** with **expected quality** is a constant trade-off.

Together with you, we would like to work on the next batch of upcoming ideas!

IMMS Institut für Mikroelektronik- und Mechatronik-Systeme gemeinnützige GmbH (IMMS GmbH)
Ehrenbergstr. 27, 98693 Ilmenau, Germany, Tel: +49 3677-8 749300, Fax: +49 3677-8 749315, **www.imms.de**
Martin Grabmann, martin.grabmann@imms.de, Tel. +49 361 663 25 22