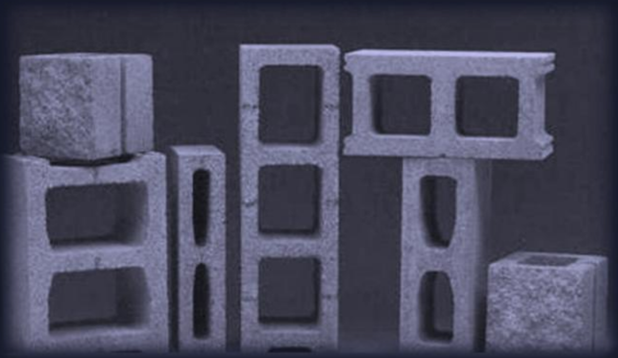
Methodology for the Design of Analog Integrated Interfaces Using Contracts

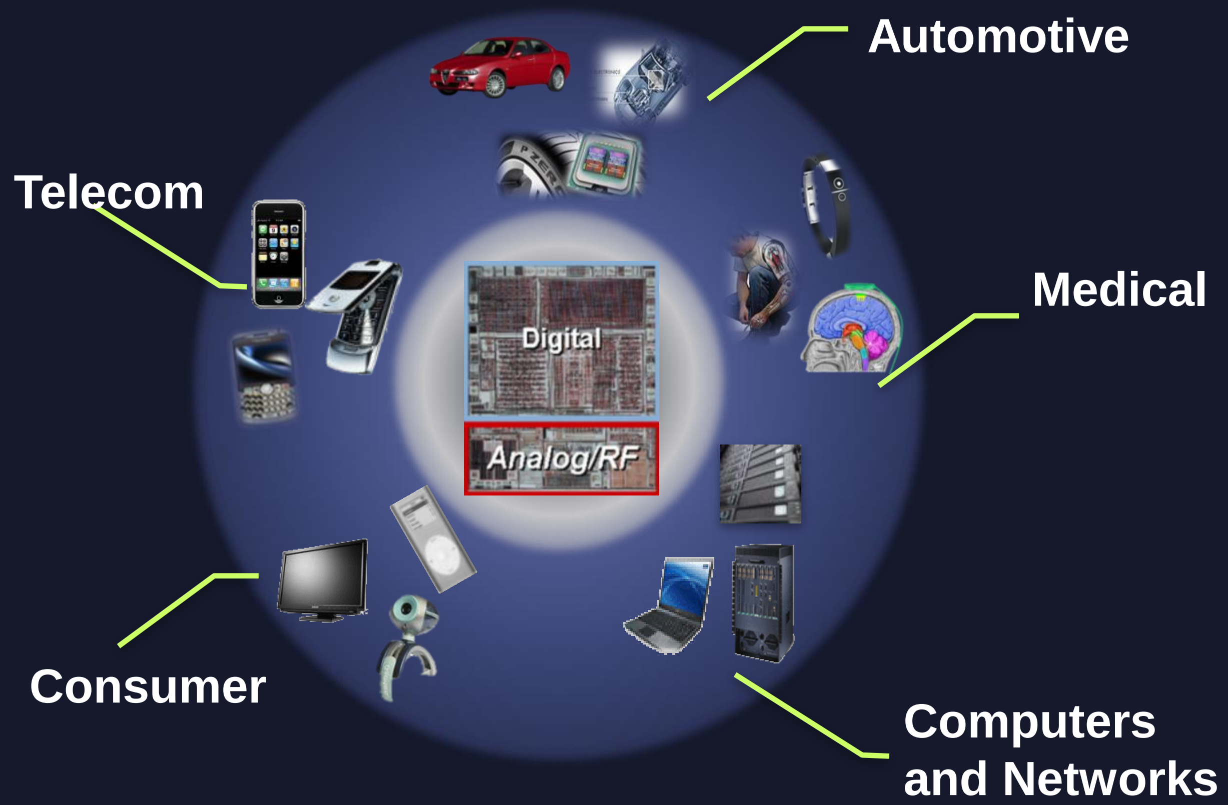
P. Nuzzo, A. Sangiovanni-Vincentelli, X. Sun, A. Puggelli

EECS Department, U.C. Berkeley

Presenters: Pierluigi Nuzzo, Safa Messaoud and Ben Zhang



The problem: complexity increases also for SoC!



Abstractions are indispensable to efficiently design and verify today's sense and control platforms

The challenge: effective system-level analog/RF design

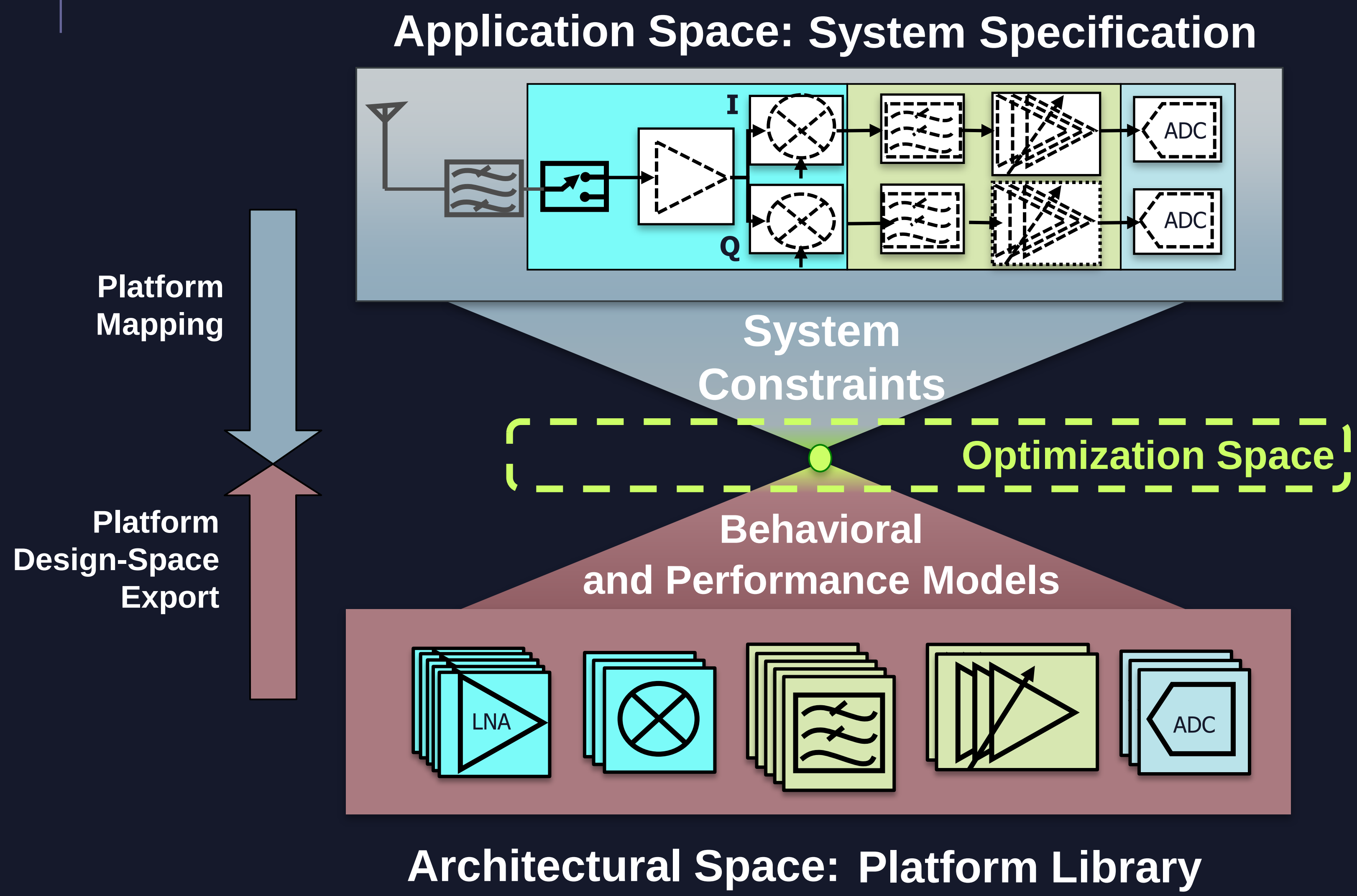
- Abstraction
  - Analog behaviors are closely tied to device physics
  - Non-ideal loading conditions (e.g. parasitics) change analog circuit performance, which changes the system behavior
- Decomposition
  - Circuits might not behave as desired outside the environment they were designed for
  - Interface effects handled with *ad hoc* modeling guidelines and interconnection components
- Structured methodology
  - Miscommunication and "false" assumptions between system and circuit designers cause many design iterations
  - Which constraints/cost should we use to optimize the design?

Outline

- Contracts for analog and mixed-signal (AMS) systems (Pierluigi)

- Contract-based design of an ultra-wide band (UWB) receiver front-end for Intelligent Tires (Safa and Ben)

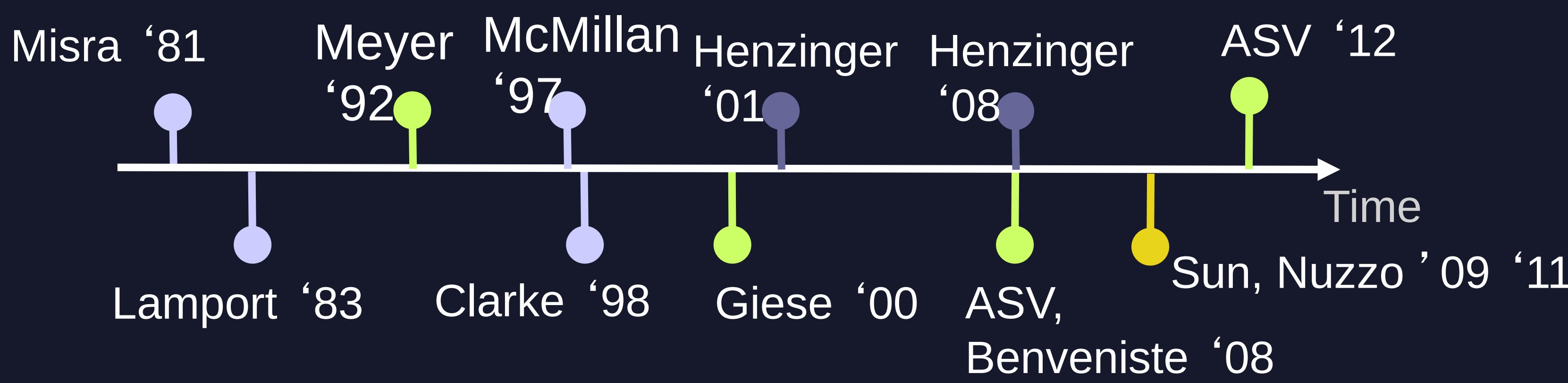
Analog Platform-Based Design as a meet-in-the-middle approach



Compositional reasoning for correct-by-construction refinement

Derive **global properties** of systems based on **local properties** of components

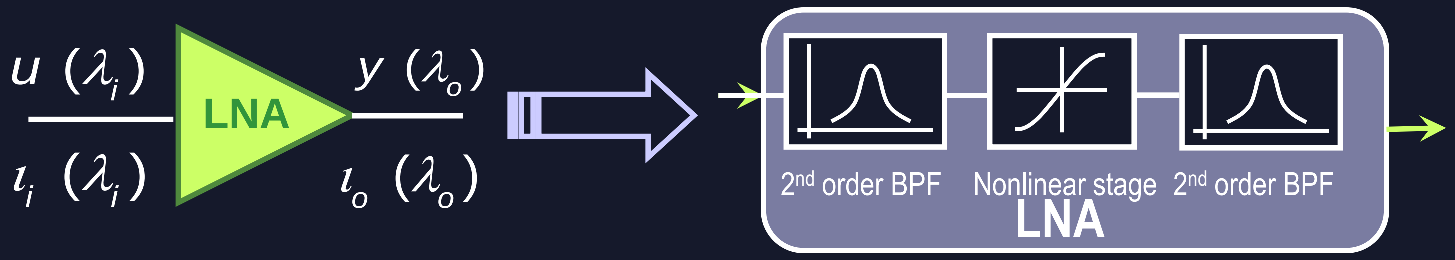
- **Contracts** as **Assume-Guarantee** pairs
- **Component** properties guaranteed under a set of assumptions on the **environment**



Need a notion of contracts to effectively abstract continuous-time infinite-state-space systems for AMS design

Analog Platform Component

- Analog Platform Component:
  - Input, output, state, configuration, interface and port domains, tolerance $\delta \in \mathcal{D}$ — $\mathcal{U}, \mathcal{Y}, \mathcal{X}, \mathcal{K}, \mathcal{I}, \Lambda$
  - Behavioral model — $\mathcal{F}(u, y, x, \kappa, \iota) = 0$
  - Feasible performance model — $\varphi_y(u, \kappa, \iota)$
  - Assumptions — $\mathcal{A}(u, y, x, \kappa, \iota, \delta) \le 0$

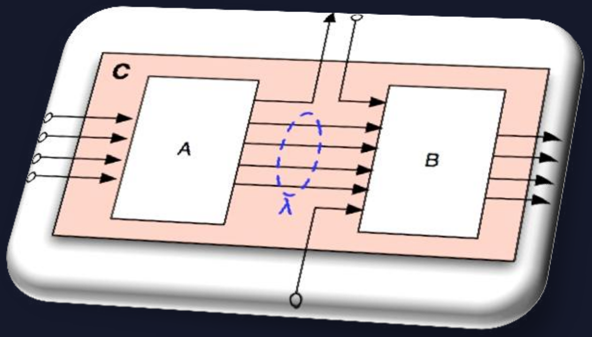


Enforcing correct compositions: Horizontal Contracts

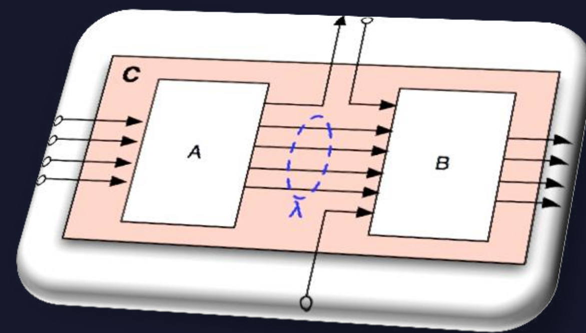
- Given **A** and **B** generate component **C = A×B**

- A *contract* is a set of assume-guarantee pairs $\mathcal{C}$ = ***{(A$_i$,G$_i$)}***
  - A_i: set of input, output, internal, configurations and interface variables satisfying a set of assumption constraints $\mathcal{A}_i$ (properties) with margin δ
  - G_i: set of output (performance) variables satisfying a set of guarantee properties $\mathcal{G}_i$ with margin ε
  - $A_i \Rightarrow G_i$

- **C** is "legal" iff
  A and **B** are compatible
  (see next slide)



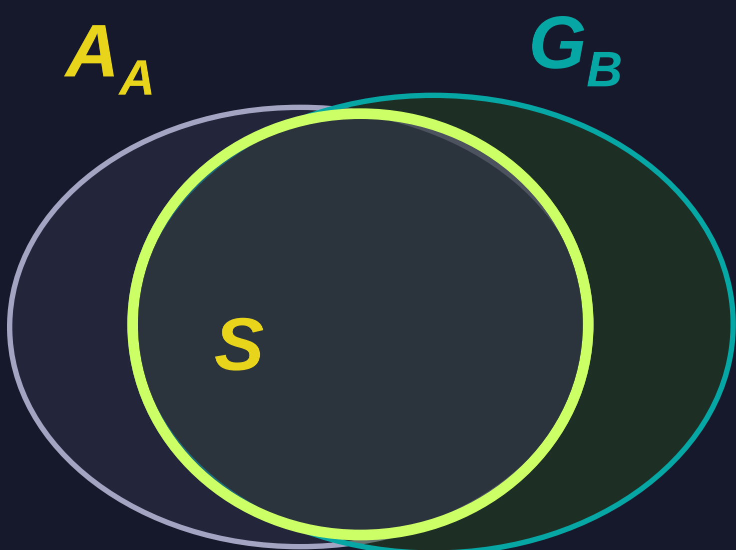
Compatible components

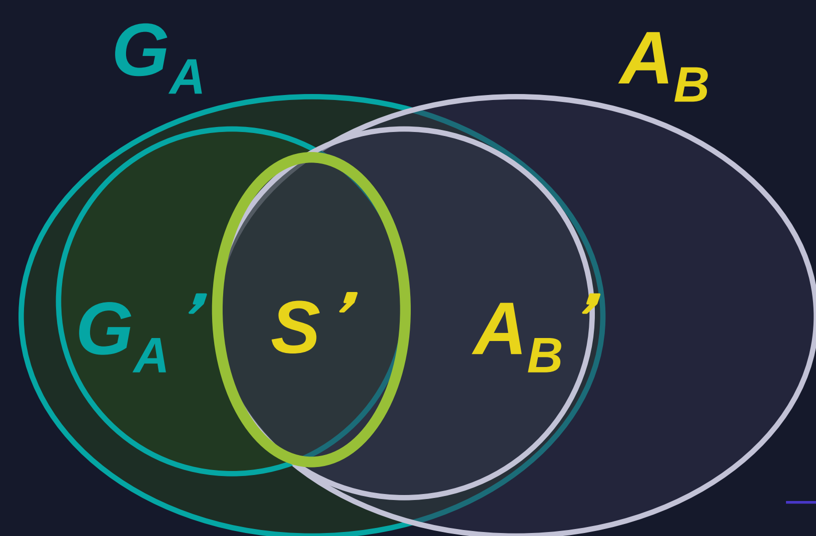


- **A** and **B** compatible at λ
  - $S = \mathscr{P}_{\lambda}(A_A) \cap \mathscr{P}_{\lambda}(G_B) \neq \varnothing$
  - $\exists\ G'_A \subseteq G_A$ and $A'_B \subseteq A_B$

    $s.t.\ \ S \Rightarrow \mathscr{P}_{\lambda}(G'_A), \quad \mathscr{P}_{\lambda}(A'_B) \Rightarrow S, \quad \mathscr{P}_{\lambda}(A'_B) \cap \mathscr{P}_{\lambda}(G'_A) \neq \varnothing$

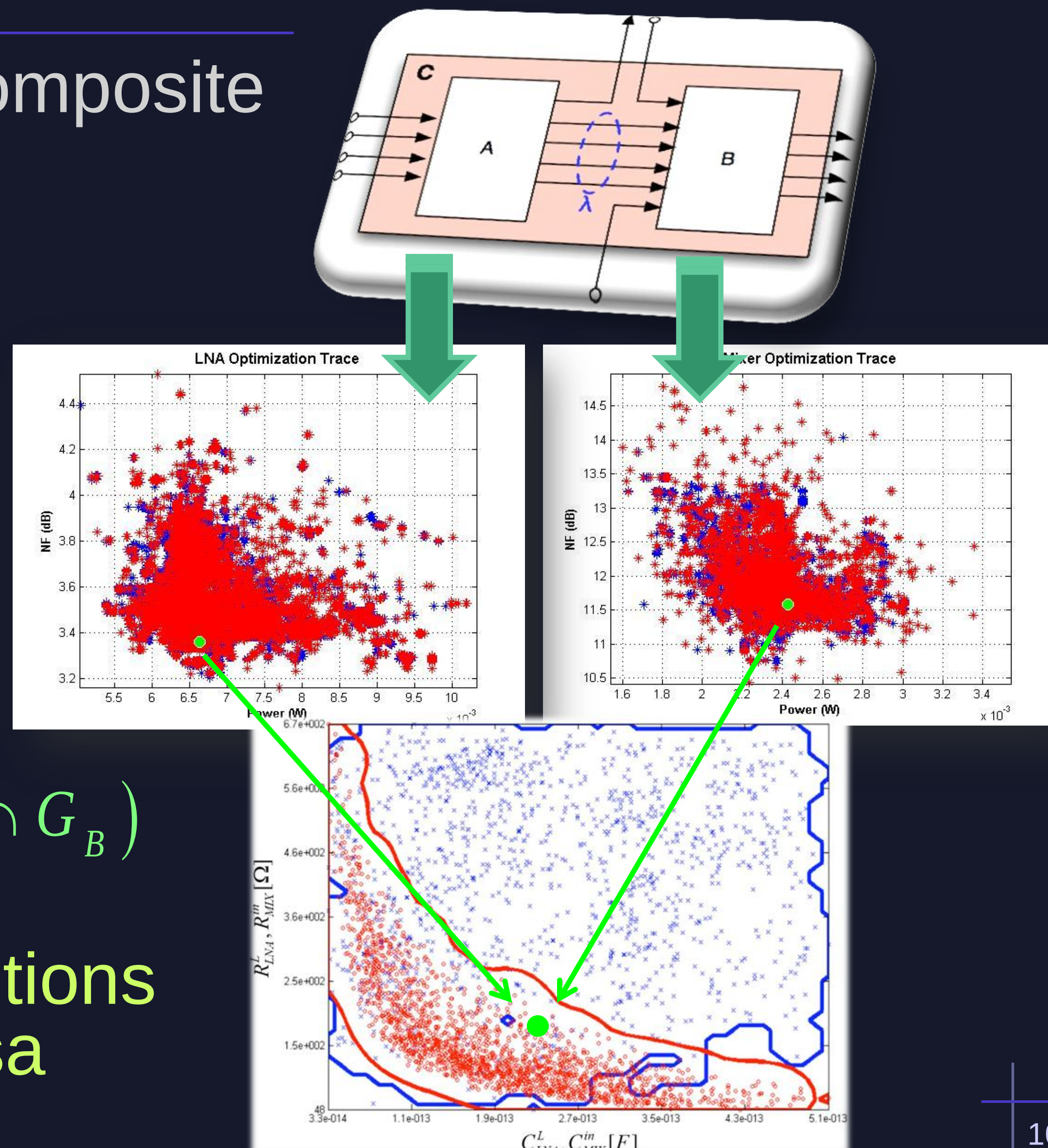
  $\mathscr{P}_{\lambda}(S)$ is the projection of *S* onto the subspace associated with λ





Contract composition

- What about the composite contract $C_A \otimes C_B$?
- Need to satisfy all assumptions!

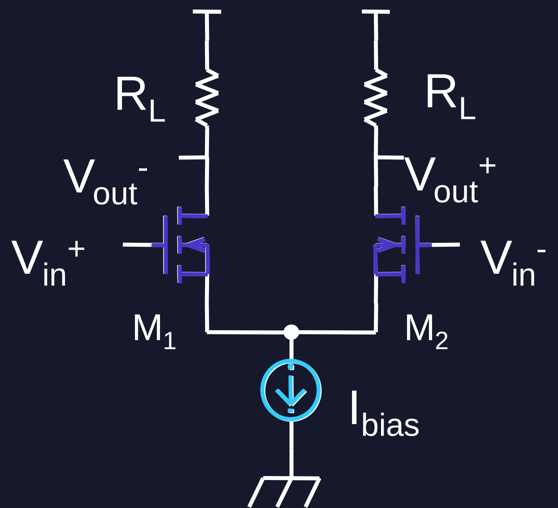


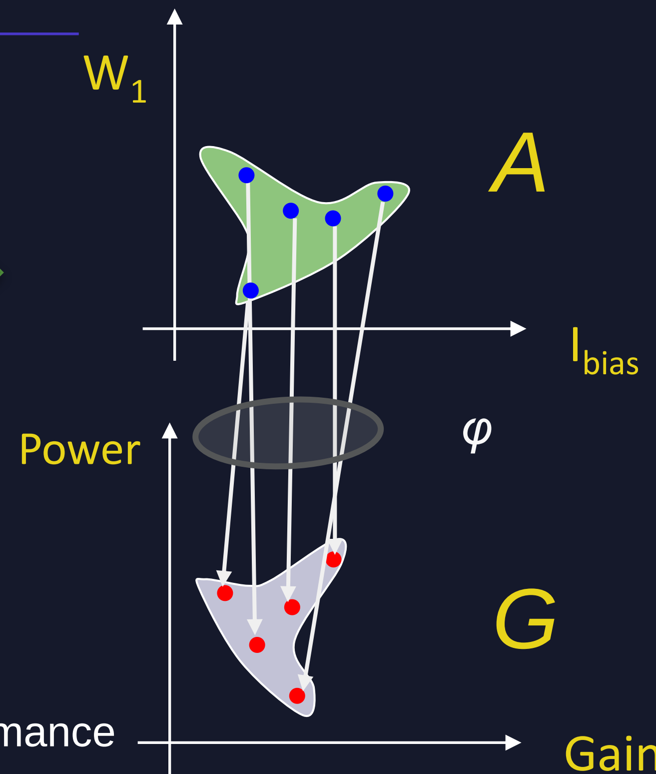
$$C_A \otimes C_B = (A, G):$$

$$\begin{cases} G_A \cap G_B \\ (A_A \cap A_B) \cup \neg(G_A \cap G_B) \end{cases}$$

- **B** relaxes assumptions of **A** and vice-versa

Feasible performance and contracts





- Given $\bar{u}, \bar{\iota}, \delta$, sample $\mathcal{K}$ such that
  $\mathcal{A}\left(\bar{u}, y, x, \kappa, \bar{\iota}, \delta\right) \le 0$
- Generate **guarantee points** using the **performance map** $g = \varphi_y\left(u, \kappa, \iota\right)$

  κ – configuration variables u – input variables ι – interface parameters

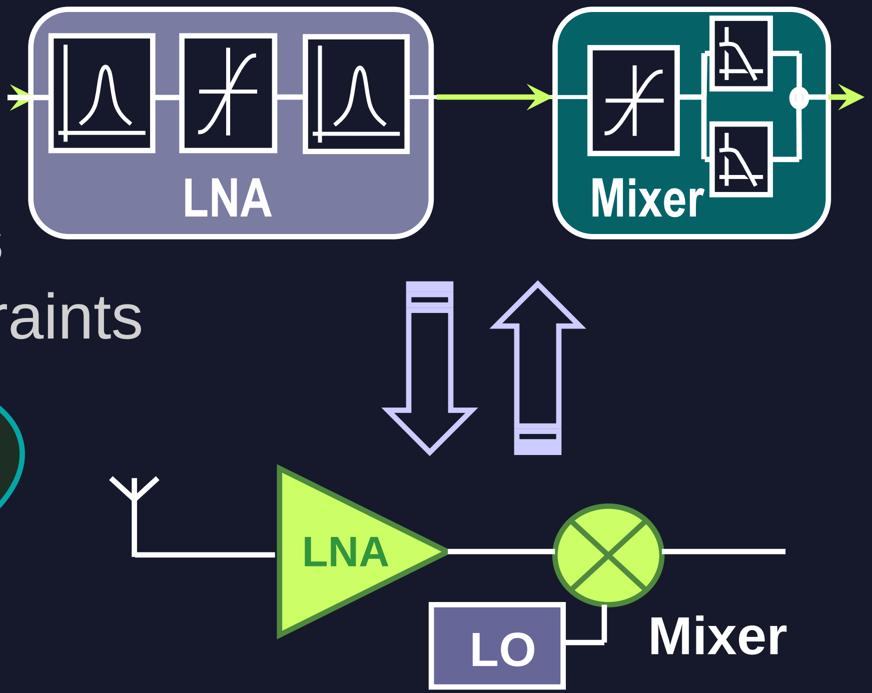
- **Describe** a contract-**consistent region** as a **classifier**

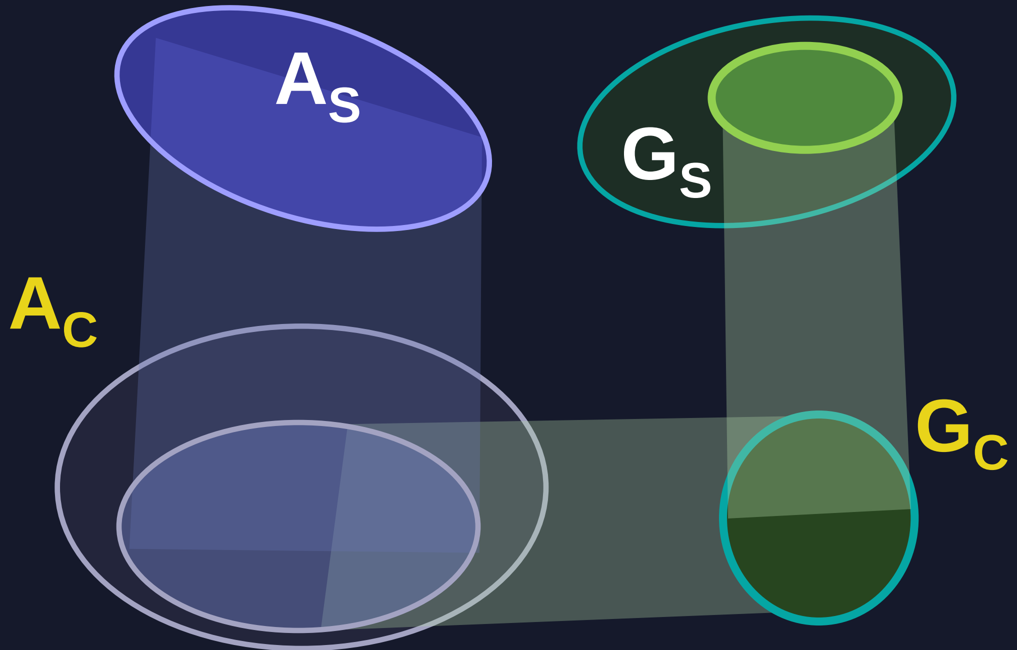
$$\mathcal{P}\left(g\right) = 1 \Leftrightarrow \exists\ \kappa \ \text{ s.t. } g = \varphi_y\left(\bar{u}, \kappa, \bar{\iota}\right)$$

- Build a **continuous approximation** from simulated points using statistical learning techniques (Support Vector Machines)

Vertical contracts and mapping

- High-level architectures may not directly match low-level architectures
  (e.g., different number
  of components)
- Vertical contracts handled as
  additional optimization constraints





Analog Contract-Based Design Flow