Novel Concepts to Improve Custom Layout Automation Capabilities

Göran Jerke[1] · Thomas Burdick[2] · Vinko Marolt[1] · Peter Herth[3] · Andrew Beckett[4]

[1] Robert Bosch GmbH [2] Cadence Design Systems SAS [3] Cadence Design Systems GmbH [4] Cadence Design Systems Ltd

ISPD 2026 · International Symposium on Physical Design · Bonn, Germany

cadence BOSCH

Talk Outline

01 **Custom Layout**
Why custom layout automation matters

02 **Our Challenge – Our Motivation – Our Opportunity**
Evolution of enhanced PCells, Top-Down meets Bottom-Up

03 **Foundation Concepts**
Partitioning, Abstraction, Reification, Uncertainty, Dialectic Motion

04 **Elementary Concepts**
Entity-centric design abstractions

05 **Composite Concepts**
Practical, application-level constructs

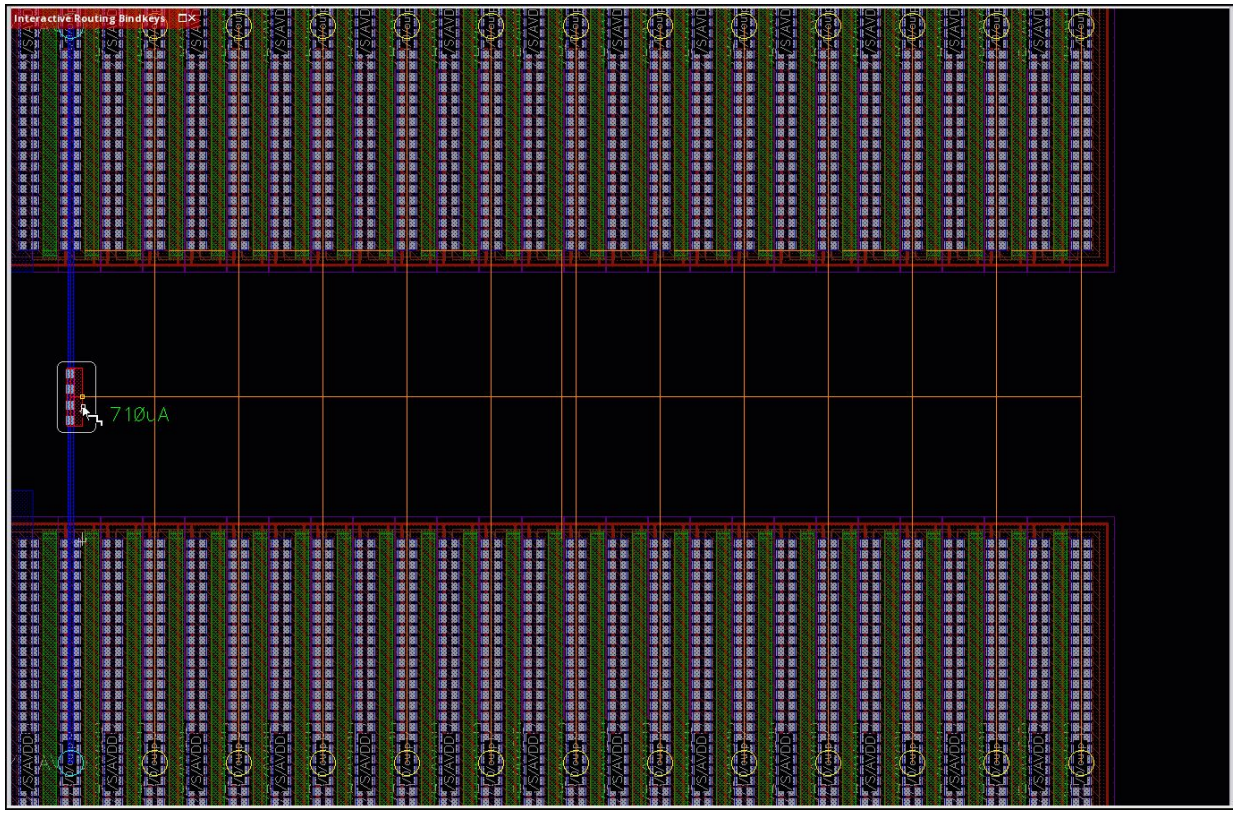
06 **Application Examples**
Concept demonstrators

07 **Summary and Outlook**
Key takeaways and future directions

The Custom Layout Challenge

Custom layout remains predominant in today's non-digital IC design domains

- Analog & Mixed-Signal
- mmW / RF
- MEMS and Sensors
- Silicon Photonics
- Power MOSFET
- Advanced Package
- Quantum Devices



Example: Custom simulation-/current-driven routing in planar node technology

The Custom Layout Challenge

Custom layout remains predominant in today's non-digital IC design domains

- Analog & Mixed-Signal
- mmW / RF
- MEMS and Sensors
- Silicon Photonics
- Power MOSFET
- Advanced Package
- Quantum Devices

Automation Barriers

- Inherent design complexity with manifold relationships
- Uncertainty from missing, incomplete, or unknown design data
- Hard to formalize constraints and design goals
- Limited compatibility within heterogeneous design flows
- Established design paradigms resist automation
- Rigid or static abstractions that fail to capture local context

Implications:

- Limited success of top-down design algorithms
- Bottom-up design with Parameterized Cells (PCells) prevails
- Human-centric "polygon pushing" is still in use

Digital vs. Analog IC Layout Automation

Digital Layout

HIGH AUTOMATION

Millions–billions of gate instances

Standardized cells reduce local solution space

Few, well-defined device parameters (w, l, nfing, nfin)

Constraint types are few and readily formalizable

Divide-and-conquer partitioning works effectively

Sparse set of bottom-up data from device instances

Algorithmic partitioning + mathematical formalization

Analog / Custom Layout

LOW AUTOMATION

Thousands of devices and nets

Complex, highly variable inter-device relationships

Information often uncertain or incomplete at request time

Semi-automated, human-centered methods remain prevalent

Hard-to-predict downstream impact of design choices

Rich set of bottom-up data needed from instances

No universal, one-size-fits-all set of algorithms

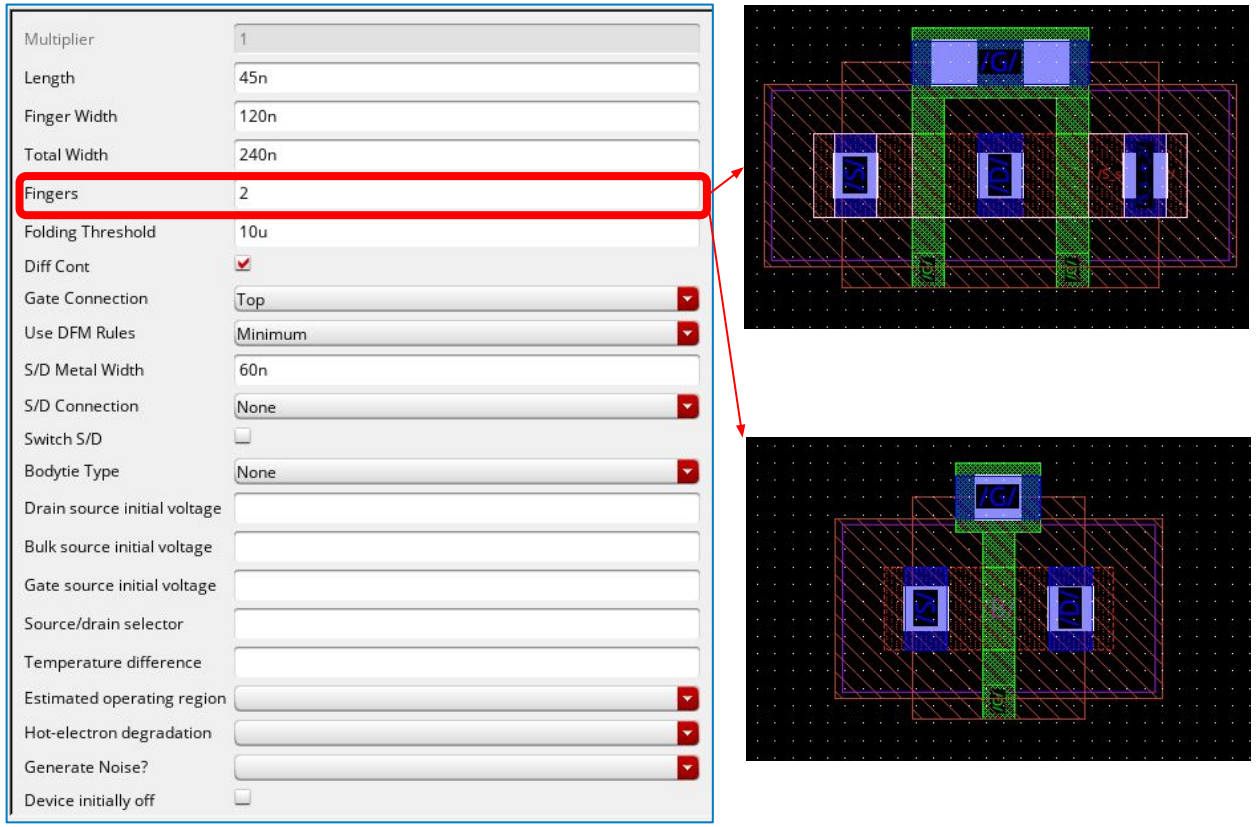
Our Challenge – Our Motivation – Our Opportunity

Classic Parameterized Cell (PCell)

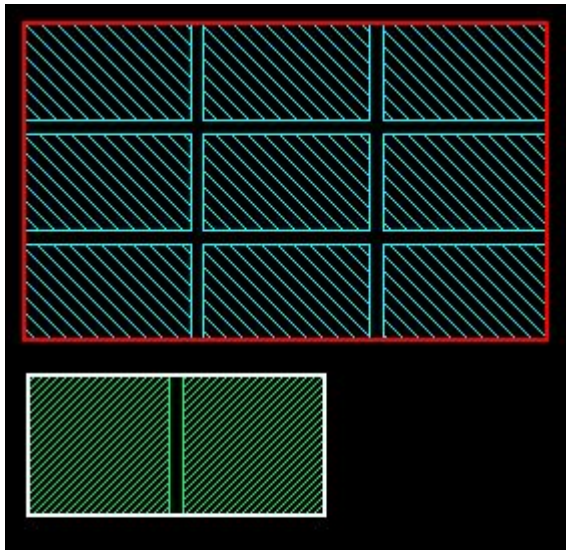
- IC Layout: Late 1980'ies
- Schematic/Symbol: ~2005
- Package Layout: 2022

- Pro's:
  - Predictable results
  - Condensed knowledge
- Con's:
  - Comparably slow and single threaded
  - Hard to verify and validate
  - Condensed knowledge

- Flavors:
  - Symbol
  - Schematic
  - Layout (IC, MEMS, Package, Power-MOS, …)



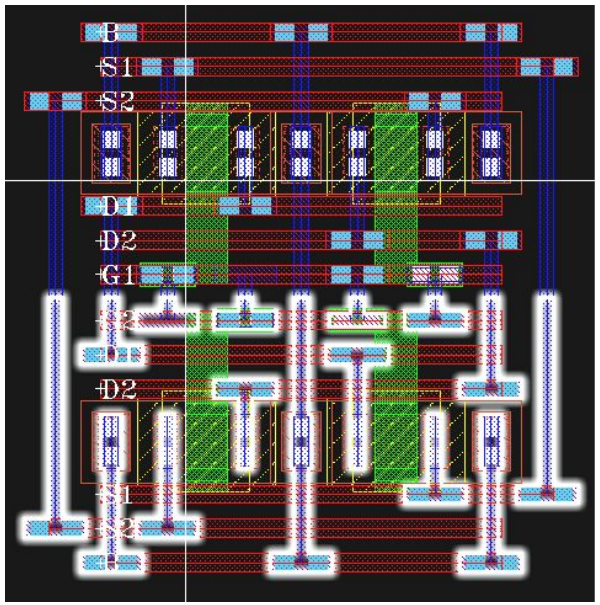
Our Challenge – Our Motivation – Our Opportunity



Command	Parameters
label	[f: pcell~>cellName] {0:0} text/drawing lowerLeft 0.1 stick <0....
group	"shape_group"
rect	poly_metal1 metal1/drawing {0.0:0.0, 0.1:0.0, 0.1:0.1, 0.0:0.1}
change layer	{Group shape_group} layer
stretch	{Group shape_group} {Group shape_group} northEast <width:l...
copy	{Group shape_group} <width + 0.4:0 R0> "t" "nil" "none" [f: n...
copy	{Group shape_group} <0:length + 0.4 R0> t - none [f: nY - 1]

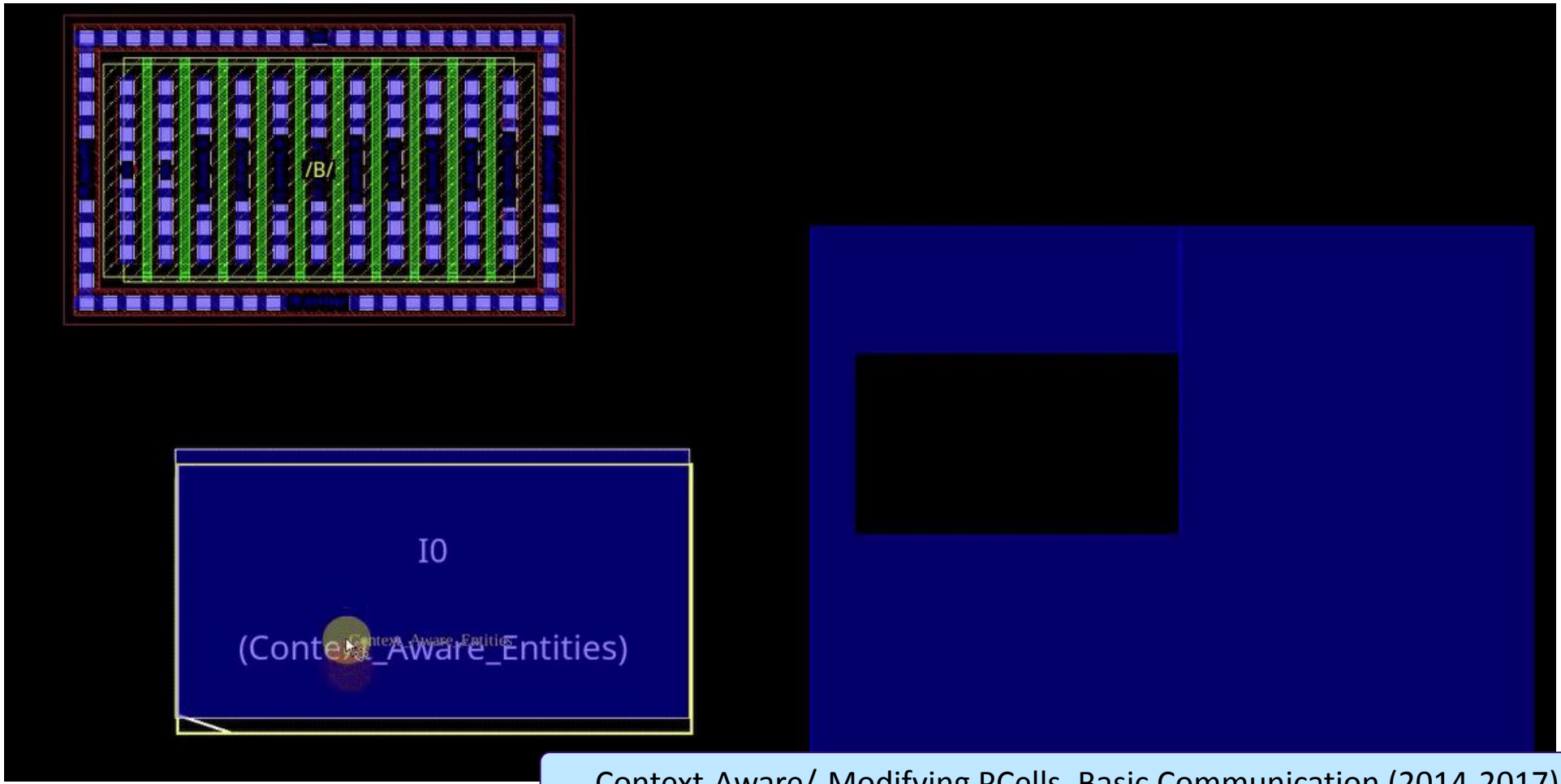
Functional and data flow programming approach (2013)

Directive	Parameter
bbox	
land	
bisect	south
bbox	
layer	[f: pMet1Layer]



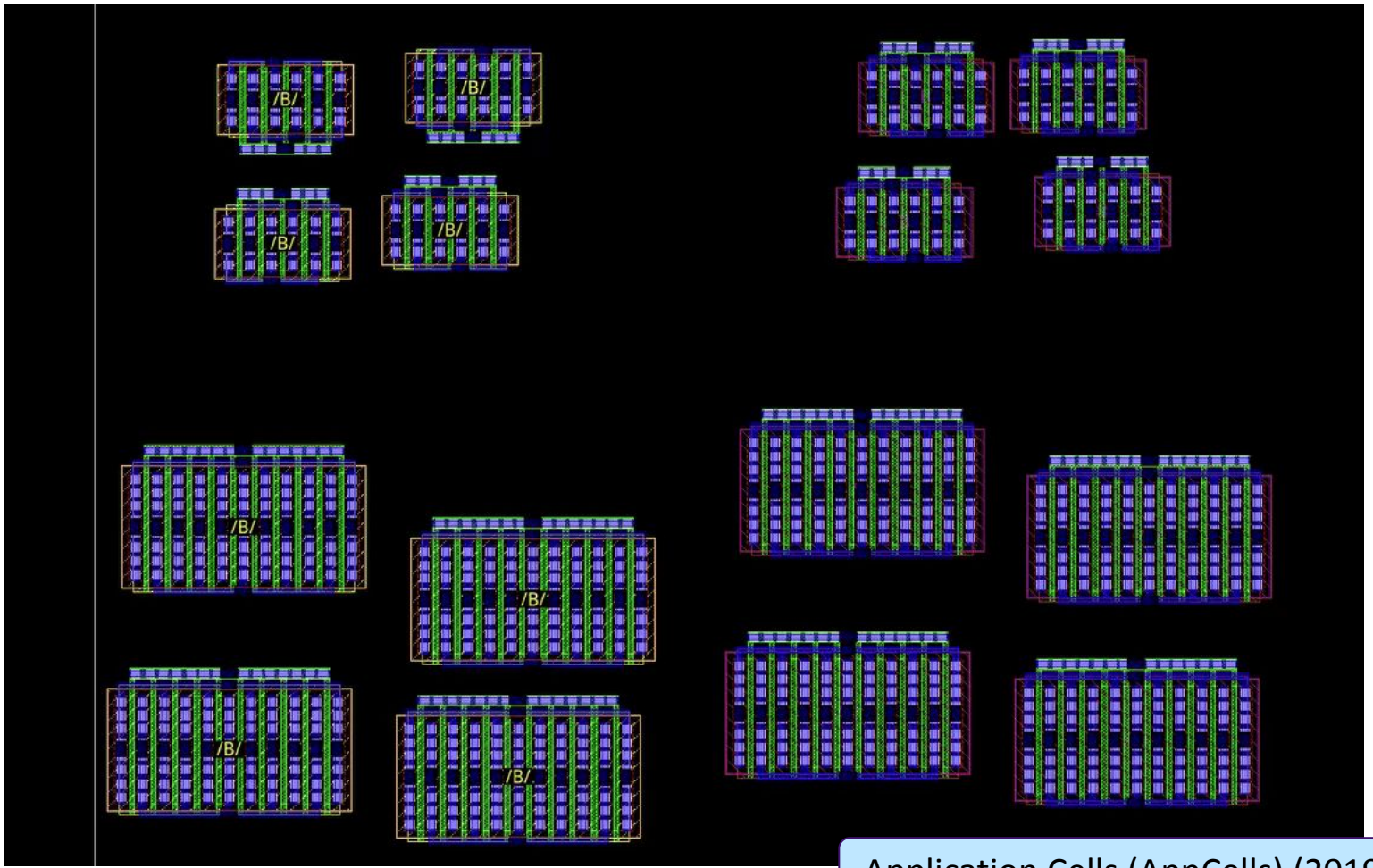
Complex queries and manipulation (2013)

Our Challenge – Our Motivation – Our Opportunity



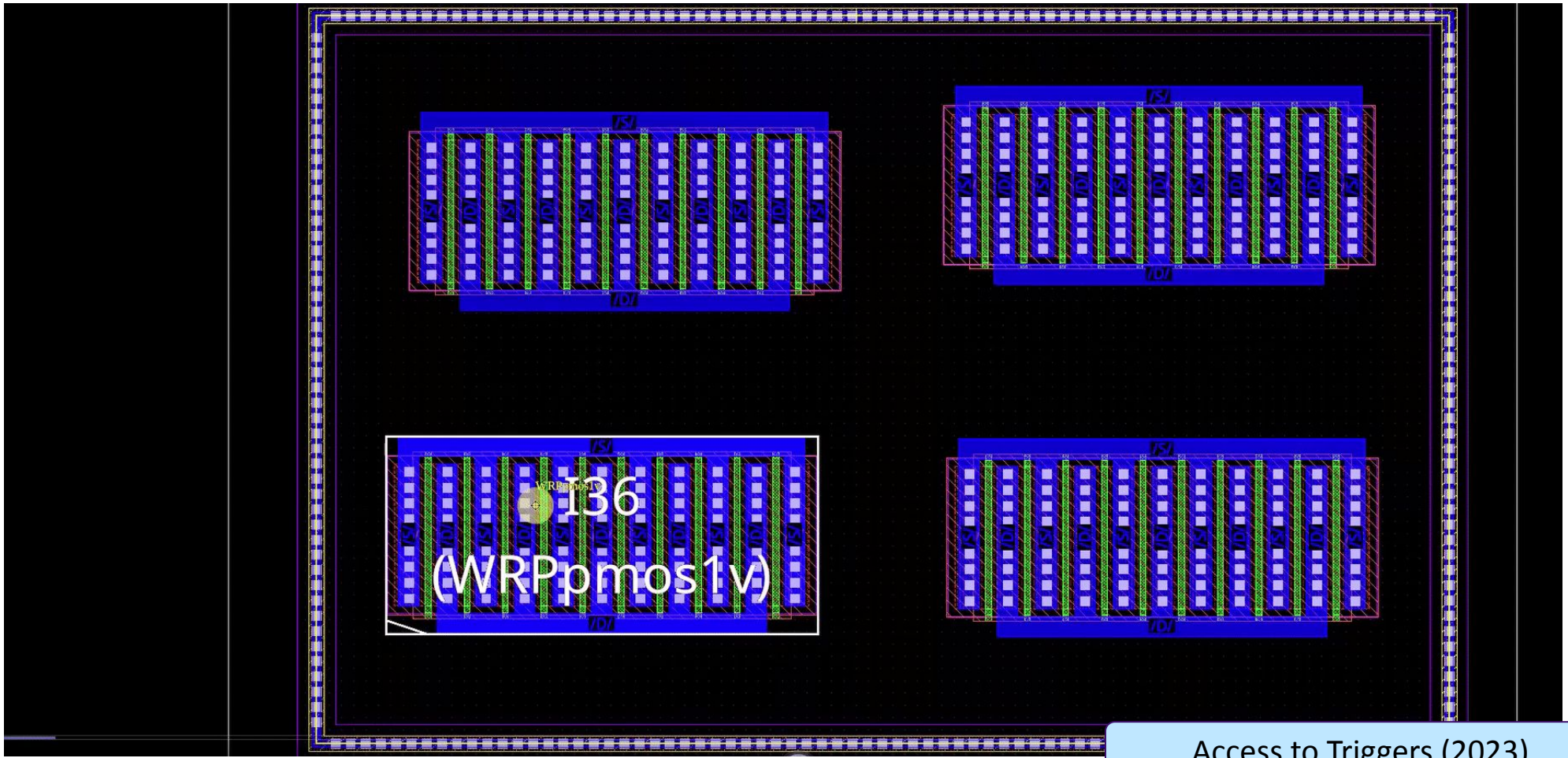
Context-Aware/-Modifying PCells, Basic Communication (2014-2017)

Our Challenge – Our Motivation – Our Opportunity



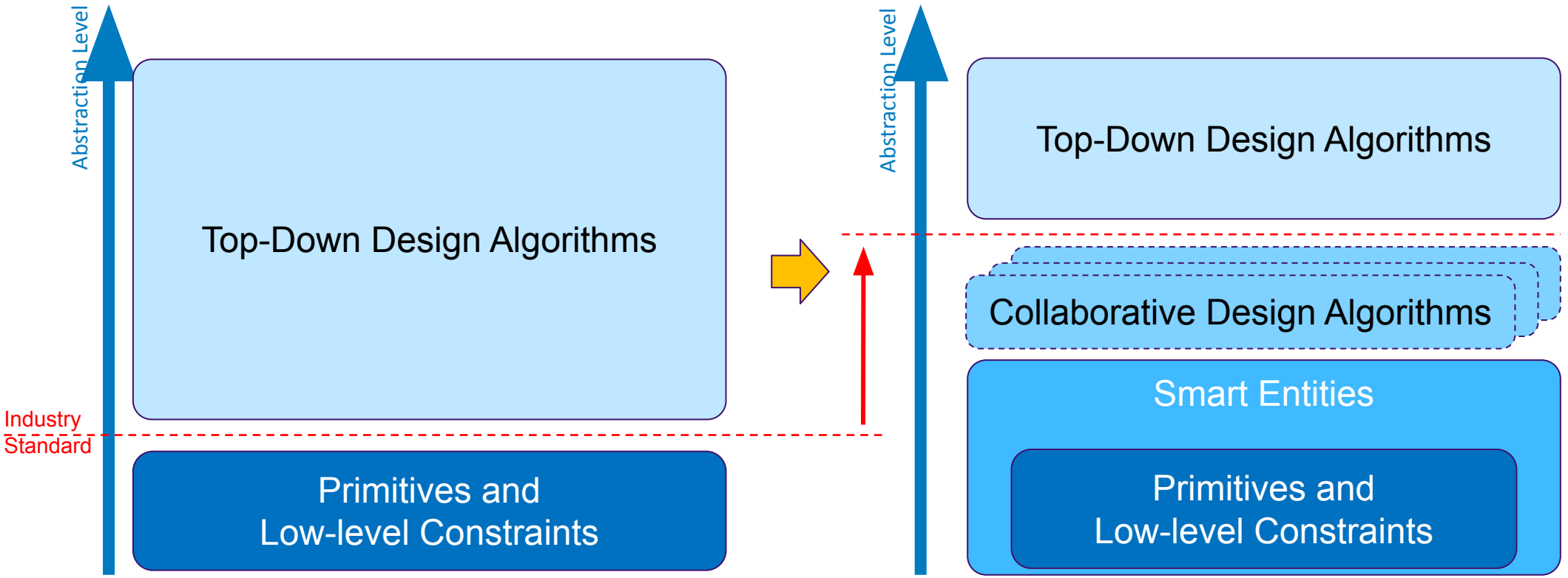
Application Cells (AppCells) (2019)

Our Challenge – Our Motivation – Our Opportunity



Access to Triggers (2023)

Our Challenge – Our Motivation – Our Opportunity



Custom Layout Automation Capability (CLAC)

Six interdependent factors that significantly impact CLAC during iterative layout planning, generation, and verification:

Problem Size

Number of design elements: device instances, nets, polygons, or constraints

Problem Complexity

Number, manifold nature, and inherent characteristics of relations between design elements

Formalization

Transforming a design problem into mathematical form for computer processing

Problem Difficulty

Inherent hardness of finding optimal solutions given requirements, goals, and constraints

Data Readiness

Quality, reliability, and availability of design data in data-driven design flows

Partitioning Ability

Effective design decomposition reduces size and space, focusing on relevant data only

A Hierarchical Concept Framework

Three interdependent concept levels to address long-standing custom automation challenges

Level 0 — Foundation Concepts

Five foundational concepts provide a theoretical base for improving custom layout automation

Partitioning

Divide large designs into manageable subproblems.

Abstraction

Simplified higher-level representations of layout, tasks and traits.

Reification

Transform passive structures into active, capable design objects.

Epistemic Uncertainty

Reflects what is unknown, but knowable. Quantify and reduce unknowns in design data and decisions.

Dialectic Motion

Iterative evolution driven by opposing forces and constraints.

Partitioning & Abstraction

Partitioning

Dividing a large design into smaller, more manageable physical regions, blocks, devices, or subdevices

Core Principles

Scoped Decomposition by logical function, physical proximity, or functional domains

Cut Optimization using metrics to minimize cross-boundary relations

Constraint Propagation carrying constraints across partitions

Iterative Refinement — coarsening or refining based on feedback

Abstraction

Simplified, higher-level representations enabling reasoning while hiding unnecessary detail

Three Types

Phenotype Abstraction — abstract view of layout cell with selectable accuracy

Task Abstraction — represent design activities at a higher, standardized level

Entity Abstraction — represent design objects for efficient modeling and integration

Example:

Placement algorithm provides relative direction instructions while delegating fine placement to capable context-considering device entities

Reification, Uncertainty & Dialectic Motion

Reification

Transforming passive structures into active, capable design objects

Core Principles

Entity-centric or function-centric mapping

Deterministic, rule-driven transformations

Traceability between abstract masters and concrete instantiations

Reversible mapping for backtracking

Epistemic Uncertainty

Inherent unpredictability from unknown requirements and evolving conditions

Core Principles

Active data gathering via capable design objects

Fast & frequent prototyping to understand parameter bounds

Explicit uncertainty modeling in chains of effect

Robust optimization over absolute optimization

Dialectic Motion

Dynamic, iterative evolution driven by opposing design forces

Core Principles

Force modeling with attractive/repulsive objectives

Objective weighting & annealing across iterations

Local-global interplay for balanced refinement

Phase-dependent strategies in floorplanning, placement, routing

Level 1 — Elementary Concepts

An entity groups geometry, connectivity, parameters, and metadata into a single manipulable design object

Realization

Instances of classes, layout primitives, or generative macro/template results

Parameterization

Configurable key-value pairs: direct, indirect, self-, remote-, active, passive

Context

Dynamic external information perceived during evaluation (context-awareness and modification)

Evaluation

Created during evaluation via generator code or reconstructed from hibernation

Representation

Expose processable info in structured way for algorithmic discovery and queries

Communication

Exchange entity parameters, layout data, metadata with other entities and EDA tools

Level 1 — Elementary Concepts

An entity groups geometry, connectivity, parameters, and metadata into a single manipulable design object

Action

Layout-related actions performed by entities during evaluation

Action Scope

Scope and Focus define partitioning of the design space, Permissions

Self-Awareness

Reason about itself based on available parameters, context, and metadata

Reactivity

Entity's ability to perform actions after receiving communication

Persistence

Transient, dormant (active/passive), or permanent entity states

Level 2 — Composite Concepts

Practical constructs that integrate elementary concepts into adaptive, application-level solutions

Proxy

Surrogate that partially or fully represents an entity, reducing data overhead and complexity

Wrapper

Encapsulates entities to enhance or reduce their representation without altering fundamentals

Evolving Entity

Uses feedback loops and context awareness to iteratively refine its layout representation

Context-Considering

Proactive agents adapting dynamically based on real-time environmental data

Application Cell

Functional objects implementing design algorithms in a visible, accessible, scoped context

Observer

Analyzes context, initiates communication; enables event-driven and watchdog paradigms

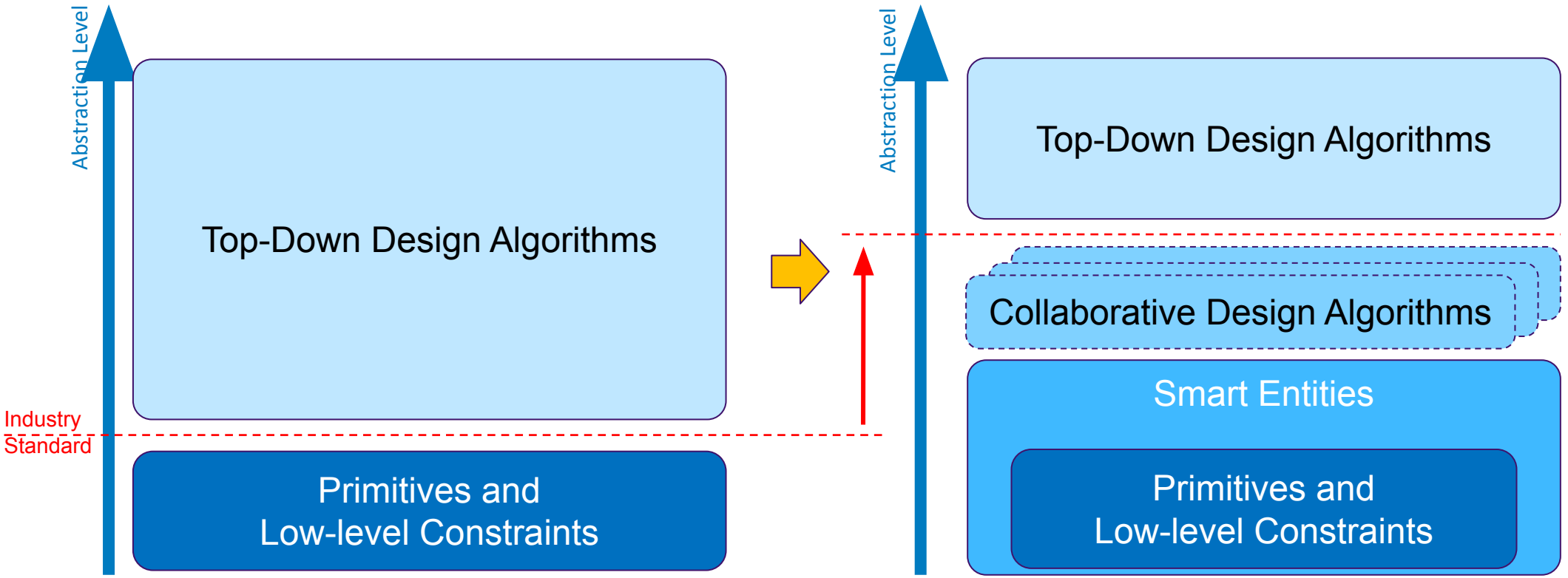
Parameterized Group

Set of entities whose layout is controlled by attached properties (dormant entity + metadata)

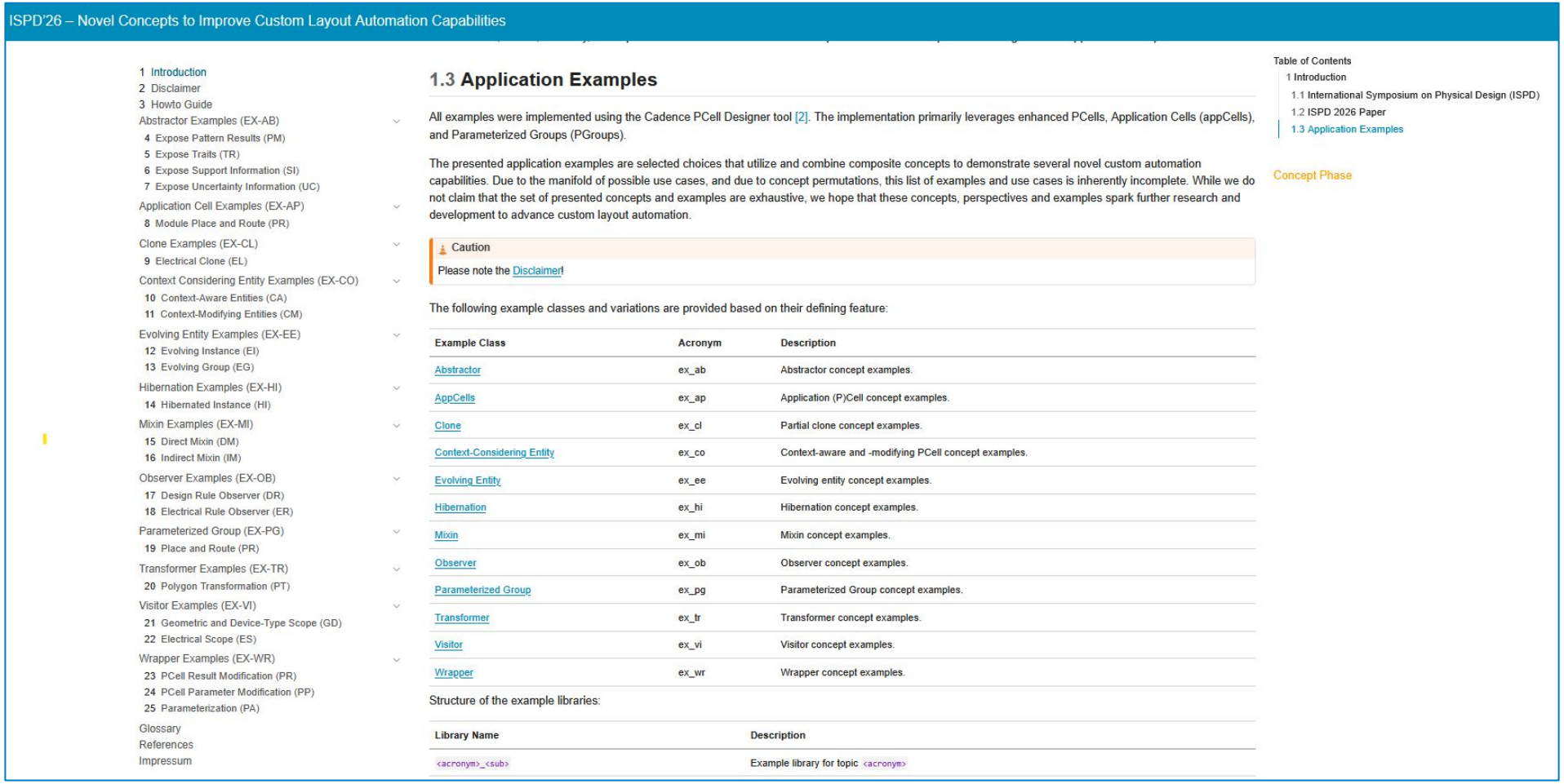
Orchestrator

Controls communication flow between entities to achieve a common design goal

Our Challenge – Our Motivation – Our Opportunity – Our Vision



Application Examples



ISPD'26 – Novel Concepts to Improve Custom Layout Automation Capabilities

1 Introduction
2 Disclaimer
3 Howto Guide
Abstractor Examples (EX-AB)
4 Expose Pattern Results (PM)
5 Expose Traits (TR)
6 Expose Support Information (SI)
7 Expose Uncertainty Information (UC)
Application Cell Examples (EX-AP)
8 Module Place and Route (PR)
Clone Examples (EX-CL)
9 Electrical Clone (EL)
Context Considering Entity Examples (EX-CO)
10 Context-Aware Entities (CA)
11 Context-Modifying Entities (CM)
Evolving Entity Examples (EX-EE)
12 Evolving Instance (EI)
13 Evolving Group (EG)
Hibernation Examples (EX-HI)
14 Hibernated Instance (HI)
Mixin Examples (EX-MI)
15 Direct Mixin (DM)
16 Indirect Mixin (IM)
Observer Examples (EX-OB)
17 Design Rule Observer (DR)
18 Electrical Rule Observer (ER)
Parameterized Group (EX-PG)
19 Place and Route (PR)
Transformer Examples (EX-TR)
20 Polygon Transformation (PT)
Visitor Examples (EX-VI)
21 Geometric and Device-Type Scope (GD)
22 Electrical Scope (ES)
Wrapper Examples (EX-WR)
23 PCell Result Modification (PR)
24 PCell Parameter Modification (PP)
25 Parameterization (PA)
Glossary
References
Impressum

1.3 Application Examples

All examples were implemented using the Cadence PCell Designer tool [2]. The implementation primarily leverages enhanced PCells, Application Cells (appCells), and Parameterized Groups (PGroups).

The presented application examples are selected choices that utilize and combine composite concepts to demonstrate several novel custom automation capabilities. Due to the manifold of possible use cases, and due to concept permutations, this list of examples and use cases is inherently incomplete. While we do not claim that the set of presented concepts and examples are exhaustive, we hope that these concepts, perspectives and examples spark further research and development to advance custom layout automation.

Caution

Please note the Disclaimer!

The following example classes and variations are provided based on their defining feature:

Example Class	Acronym	Description
Abstractor	ex_ab	Abstractor concept examples.
AppCells	ex_ap	Application (P)Cell concept examples.
Clone	ex_cl	Partial clone concept examples.
Context-Considering Entity	ex_co	Context-aware and -modifying PCell concept examples.
Evolving Entity	ex_ee	Evolving entity concept examples.
Hibernation	ex_hi	Hibernation concept examples.
Mixin	ex_mi	Mixin concept examples.
Observer	ex_ob	Observer concept examples.
Parameterized Group	ex_pg	Parameterized Group concept examples.
Transformer	ex_tr	Transformer concept examples.
Visitor	ex_vi	Visitor concept examples.
Wrapper	ex_wr	Wrapper concept examples.

Structure of the example libraries:

Library Name	Description
<acronym>_<sub>	Example library for topic <acronym>

Table of Contents
1 Introduction
1.1 International Symposium on Physical Design (ISPD)
1.2 ISPD 2026 Paper
1.3 Application Examples

Concept Phase

Examples and Implementation: Cadence PCell Designer Community Forum

Summary and Outlook

Key Contributions

- Structured conceptual framework with three interdependent levels to address custom automation complexities
- *Foundation concepts* offer a theoretical base for understanding challenges like uncertainty
- *Elementary concepts* introduce self- and context-aware design entities as first-class objects
- *Composite concepts* combine building blocks into practical, adaptive solutions
- Concepts are generic and transferable beyond physical design once adapted
- Framework enables flexible combination of bottom-up and top-down design approaches

Future Directions

- "Top-down design meets bottom-up design" paradigm evolution
- Leveraging generative AI for design algorithms
- Entity communication & action synchronization
- Managing large collections of interacting entities

Vision: From manual "polygon pushing" towards automated user-guided layout generation.

Thank You!

Paper: https://doi.org/10.1145/3764386.3779617

ISPD 2026 · Bonn, Germany · March 15–18, 2026