2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

MUNICH, GERMANY
DECEMBER 6 - 7, 2022

Automated Creation of Reusable Generators for Analog IC Design with the Intelligent IP Method

Uwe Eichler, Benjamin Prautsch, Torsten Reich

Fraunhofer IIS/EAS

SPONSORED BY THE Federal Ministry of Education and Research

Fraunhofer IIS

accellera SYSTEMS INITIATIVE

Agenda

- Motivation
- What is a generator?
- Layout templates
- Automatic creation of generators
- Generator and Simulation Results
- Summary / Outlook / References

Motivation (for analog layout automation)

- Digital IC design:
  - Circuit and layout synthesis based on an HDL description and standard cells is state of art
- Analog IC design:
  - Dominated by manual work, without programmatic entry, soft IP, or layout synthesis
  - Reduced efficiency and design safety
  - Few automation approaches, not well established

Analog IC Design Automation Approaches

- Synthesis
  - search for feasible / optimal P & R solution in very large parameter space
  - requires definition of many constraints due to complexity

  → handle complexity
- Generators
  - Programmatic circuit description, creates design data
  - Parameters to control topology, sizing, placement, routing
  - Circuit-class-specific, deterministic, can be organized in libraries

  → reduce complexity

Analog IC Design Automation Approaches

- Generators + Optimization
  - Generators for basic building blocks (array arrangements of unit devices, well scalable layouts, easy to implement as generator)
  - Need to arrange such blocks in hierarchical designs with optimized area and routing

  → use optimization to adjust generator parameters

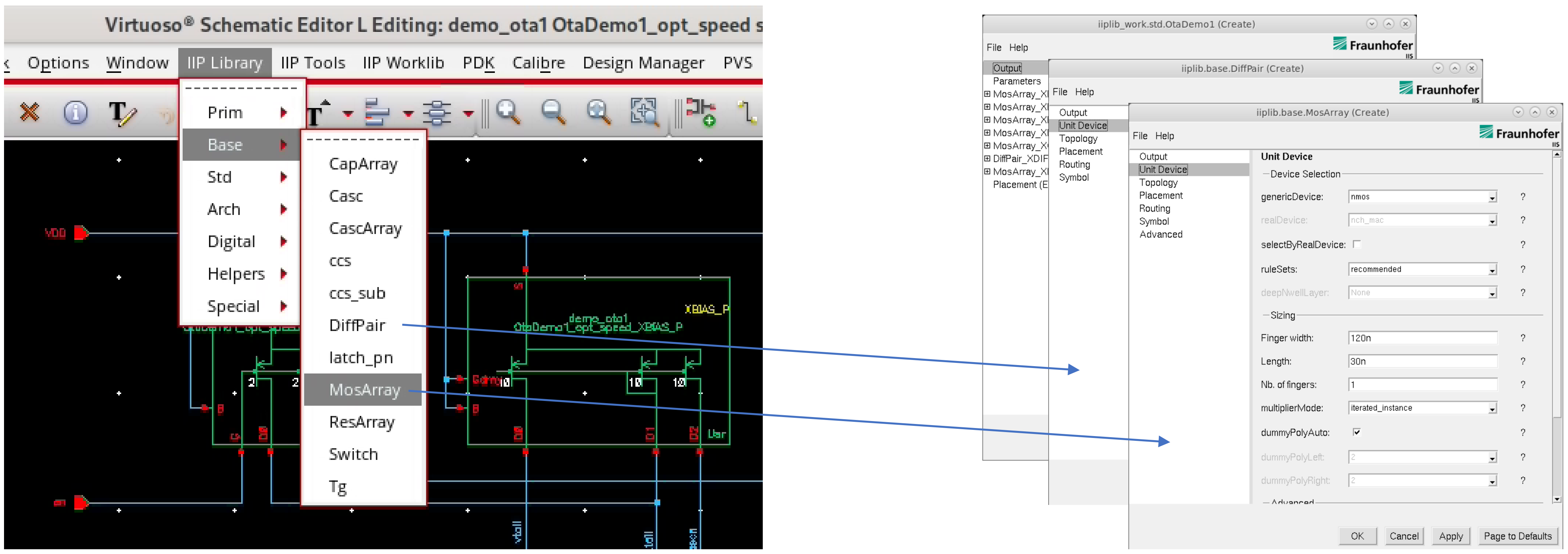
  → enables layout-aware sizing and optimization

What is a generator?

- *"Procedural generators are often proposed for analog IC design automation. They promise to encapsulate designer knowledge and intellectual property (IP) data in a deterministic and reusable way."* *[this paper]*
- Known generator approaches:
  - Cadence PCell / Synopsys PyCell [1, 2]
  - IPGen 1Stone (not available anymore) [3]
  - Fraunhofer IIS/EAS Intelligent IP ("IIP", this publication)
  - Berkeley Analog Generator [4]
  - Some more, mostly academic approaches

What is a generator?

- specific for a type of circuits
- also called an IIP (a kind of analog soft IP)
- integrates as a library of IIPs in the design environment



Unit Device

Device Selection

genericDevice:	nmos
realDevice:	nch_mac
selectByRealDevice:	☐
ruleSets:	recommended
deepNwellLayer:	None

Sizing

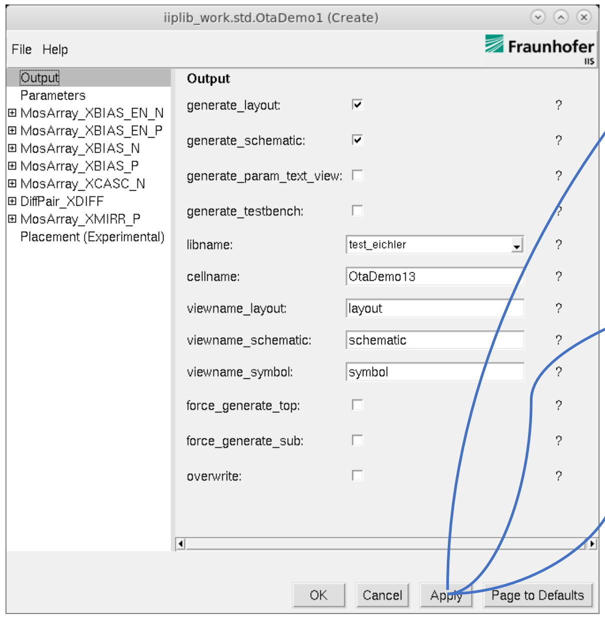
Finger width:	120n
Length:	30n
Nb. of fingers:	1
multiplierMode:	iterated_instance
dummyPolyAuto:	✓
dummyPolyLeft:	2
dummyPolyRight:	2

Advanced

What is a generator?

- Adjust parameters for e.g. topology, devices, sizing, placement, routing
- Generates cellviews in your design library (static, not as Pcell)

Generator "OtaDemo1"



iiplib_work.std.OtaDemo1 (Create)

File Help — Fraunhofer IIS

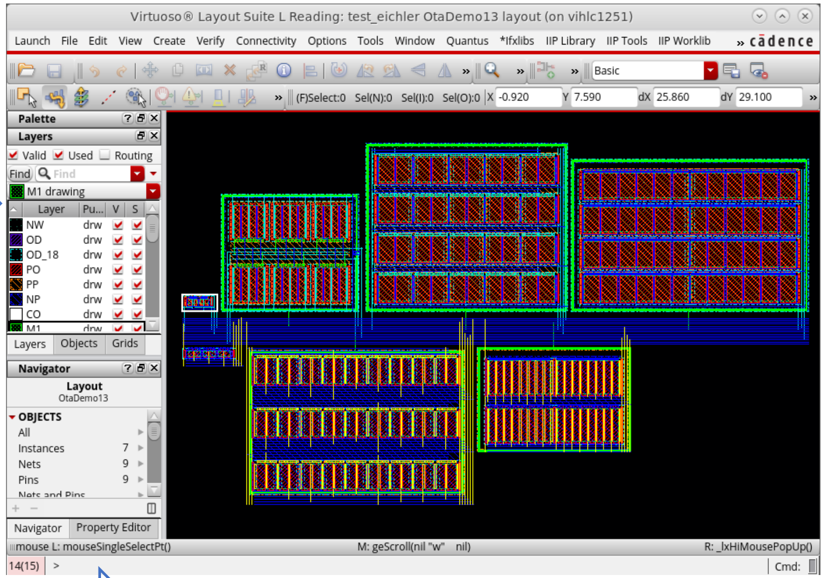
Output
Parameters
⊞ MosArray_XBIAS_EN_N
⊞ MosArray_XBIAS_EN_P
⊞ MosArray_XBIAS_N
⊞ MosArray_XBIAS_P
⊞ MosArray_XCASC_N
⊞ DiffPair_XDIFF
⊞ MosArray_XMIRR_P
Placement (Experimental)

Output

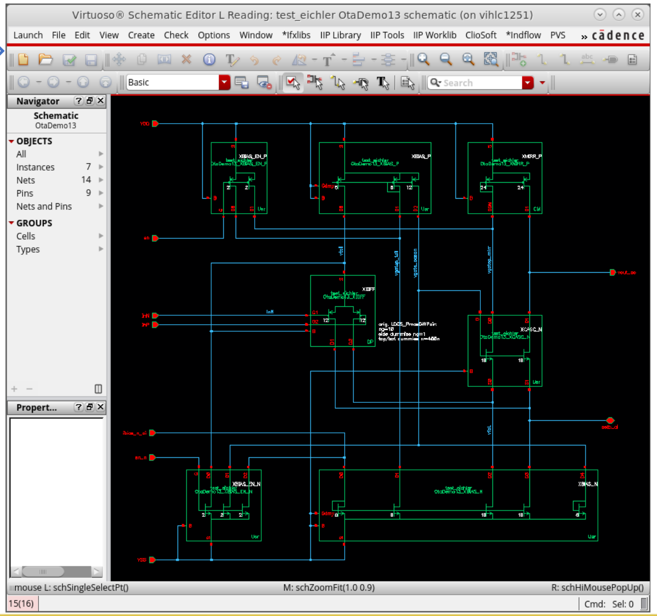
generate_layout:	☑	?
generate_schematic:	☑	?
generate_param_text_view:	☐	?
generate_testbench:	☐	?
libname:	test_eichler	?
cellname:	OtaDemo13	?
viewname_layout:	layout	?
viewname_schematic:	schematic	?
viewname_symbol:	symbol	?
force_generate_top:	☐	?
force_generate_sub:	☐	?
overwrite:	☐	?

OK Cancel Apply Page to Defaults

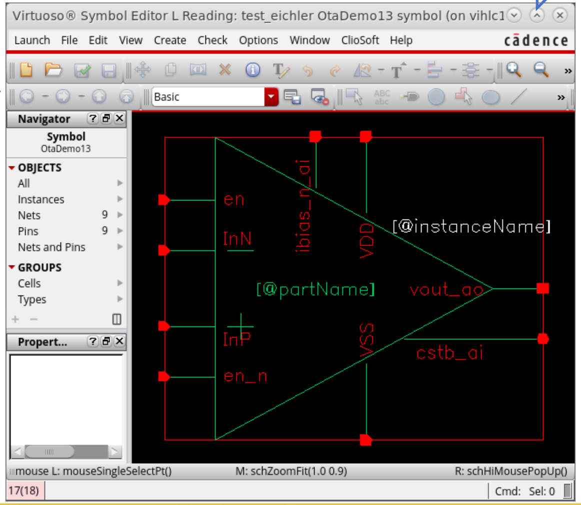
layout


Virtuoso® Layout Suite L Reading: test_eichler OtaDemo13 layout (on vihlc1251)
Launch File Edit View Create Verify Connectivity Options Tools Window Quantus *Ifxlibs IIP Library IIP Tools IIP Worklib
cadence
Palette
Layers
Valid Used Routing
M1 drawing
NW drw
OD drw
OD_18 drw
PO drw
PP drw
NP drw
CO drw
M1 drw
Layers Objects Grids
Navigator
Layout OtaDemo13
OBJECTS
All
Instances 7
Nets 9
Pins 9
Navigator Property Editor
mouse L: mouseSingleSelectPt() M: geScroll(nil "w" nil) R: _lxHiMousePopUp()

schematic

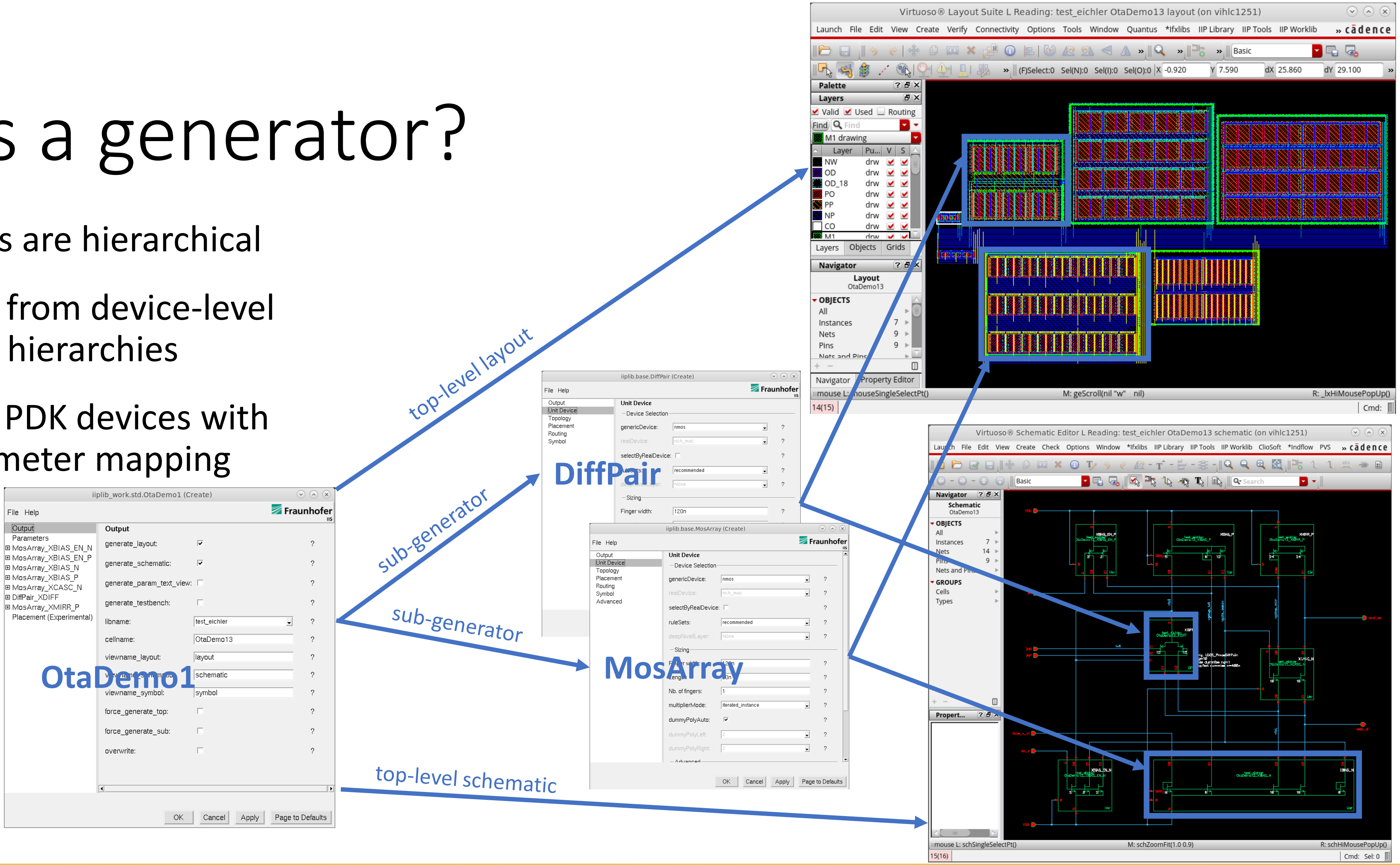

Virtuoso® Schematic Editor L Reading: test_eichler OtaDemo13 schematic (on vihlc1251)
Launch File Edit View Create Check Options Window *Ifxlibs IIP Library IIP Tools IIP Worklib ClioSoft *Indflow PVS
cadence
Navigator
Schematic OtaDemo13
OBJECTS
All
Instances 7
Nets 14
Pins 9
Nets and Pins
GROUPS
Cells
Types
mouse L: schSingleSelectPt() M: schZoomFit(1.0 0.9) R: schHiMousePopUp()

symbol


Virtuoso® Symbol Editor L Reading: test_eichler OtaDemo13 symbol (on vihlc1
Launch File Edit View Create Check Options Window ClioSoft Help
cadence
Navigator
Symbol OtaDemo13
OBJECTS
All
Instances
Nets 9
Pins 9
Nets and Pins
GROUPS
Cells
Types
en
InN
InP
en_n
ibias_n_ai
VDD
VSS
[@instanceName]
[@partName]
vout_ao
cstb_ai
mouse L: mouseSingleSelectPt() M: schZoomFit(1.0 0.9) R: schHiMousePopUp()

What is a generator?

- IIP generators are hierarchical
- IIPs available from device-level up to several hierarchies
- Direct use of PDK devices with generic parameter mapping



Virtuoso® Layout Suite L Reading: test_eichler OtaDemo13 layout (on vihlc1251)
Launch File Edit View Create Verify Connectivity Options Tools Window Quantus *Ifxlibs IIP Library IIP Tools IIP Worklib cadence
top-level layout
sub-generator
sub-generator
top-level schematic
DiffPair
MosArray
OtaDemo1
iiplib.base.DiffPair (Create)
iiplib.base.MosArray (Create)
Virtuoso® Schematic Editor L Reading: test_eichler OtaDemo13 schematic (on vihlc1251)
Fraunhofer

iiplib_work.std.OtaDemo1 (Create)

File Help

Output
Parameters
MosArray_XBIAS_EN_N
MosArray_XBIAS_EN_P
MosArray_XBIAS_N
MosArray_XBIAS_P
MosArray_XCASC_N
DiffPair_XDIFF
MosArray_XMIRR_P
Placement (Experimental)

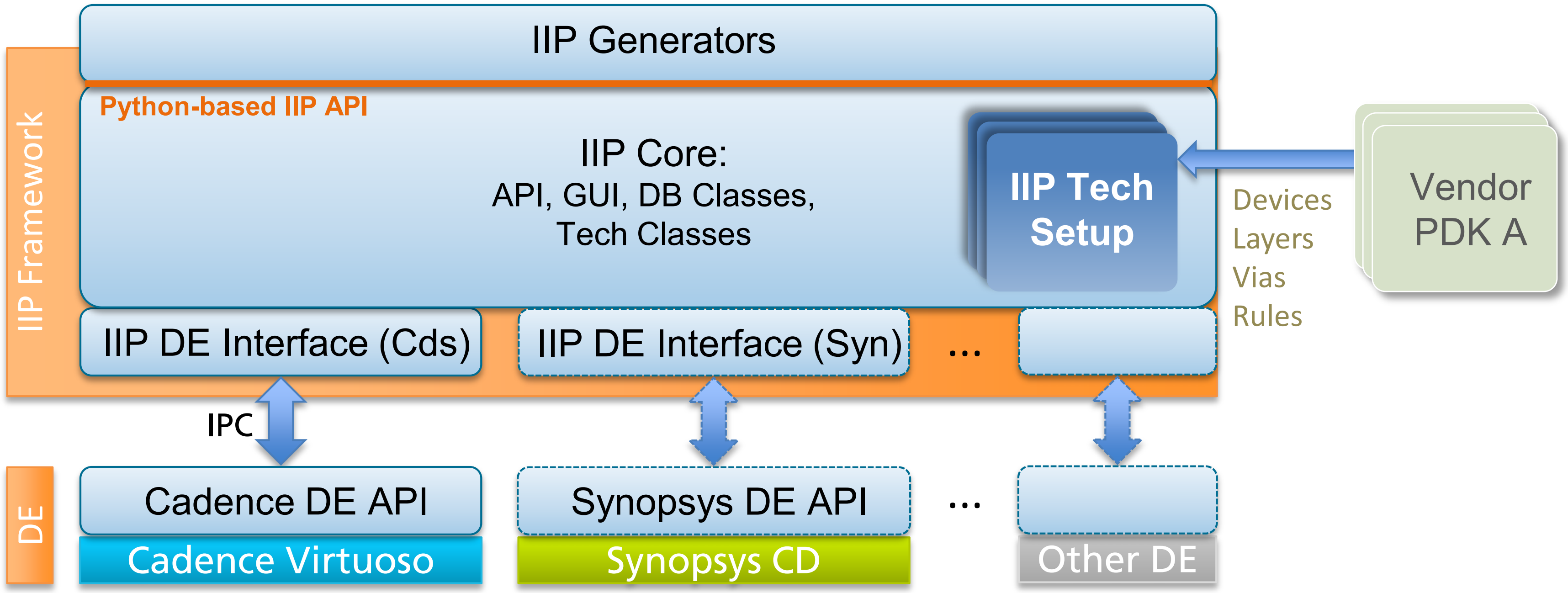
Output

generate_layout:	✓
generate_schematic:	✓
generate_param_text_view:	☐
generate_testbench:	☐
libname:	test_eichler
cellname:	OtaDemo13
viewname_layout:	layout
viewname_schematic:	schematic
viewname_symbol:	symbol
force_generate_top:	☐
force_generate_sub:	☐
overwrite:	☐

OK Cancel Apply Page to Defaults

What is a generator?

- implemented using a generic circuit description language based on Python



Motivation (for automated generator creation)

- How to create and integrate generators efficiently?
  - → Trade-off: implementation effort, reusability and acceptance
- Who should implement and maintain generator IP?
  - → Design team / CAD department / IP vendor / EDA vendor?
  - → Circuit IP and technology data often confidential
- Which interfaces and standards should be used?
  - → Programming language, Tool and PDK interfaces, Interoperability?

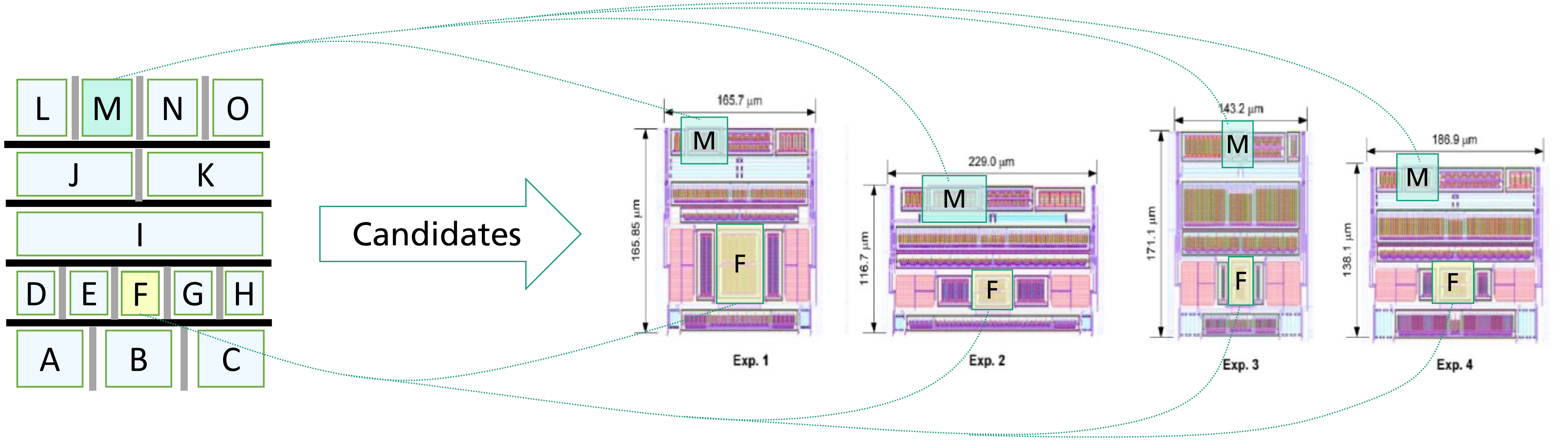
→ Automate also the generator implementation

Generator-based design of hierarchical circuits

- Use generators up to base-level only
  - → Already improves productivity
  - → Limited reuse due to manual effort for upper hierarchies
- Custom, circuit-specific generator implementation
  - → Large implementation effort for flexible, area-efficient layout
- Generator using abstract layout templates
  - → templates define relative positions of sub-block instances and interconnects
  - → usually regular arrangements, reusable across hierarchies
  - → fast generator implementation as P & R code encapsulated by template API

Template-based layout generation

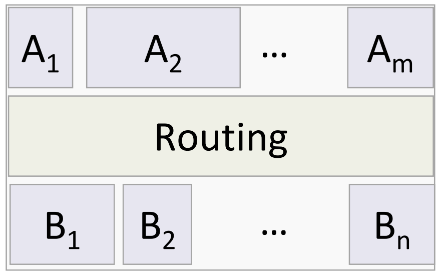
- Not a new approach: [5]



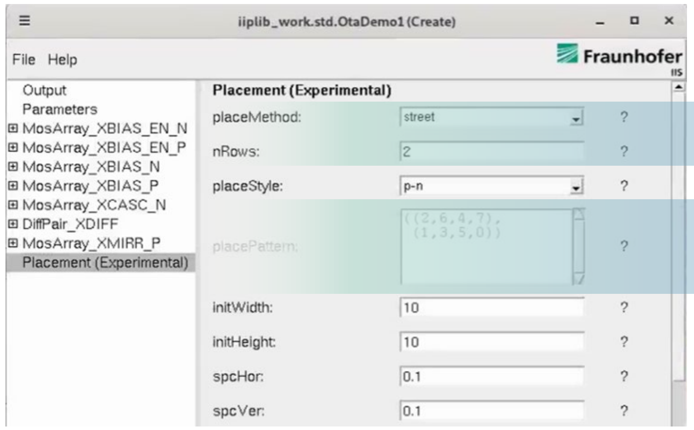
[R. Castro López et al., 2008]

Template-based layout generation

- Template P&R algorithms were embedded into IIP
- Template properties accessible through user interface



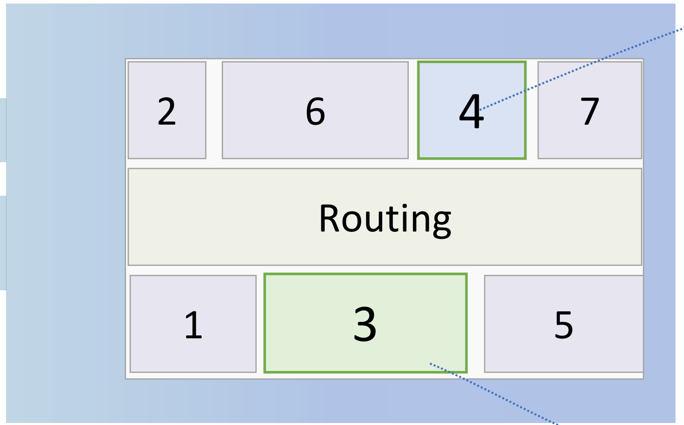
Template „Street“



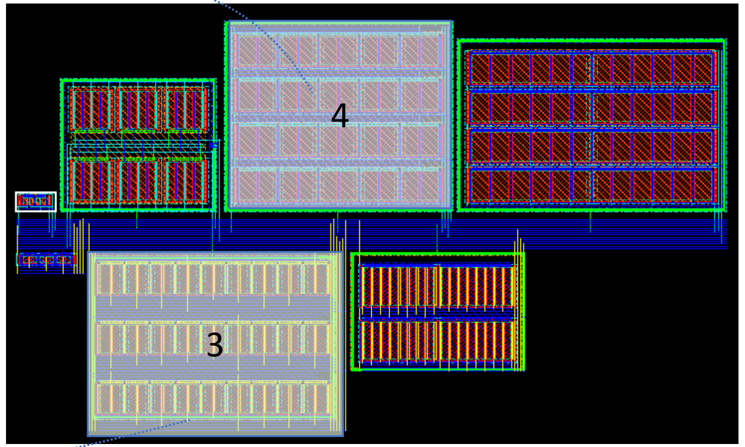
Placement (Experimental)

placeMethod:	street
nRows:	2
placeStyle:	p-n
placePattern:	((2,6,4,7), (1,3,5,0))
initWidth:	10
initHeight:	10
spcHor:	0.1
spcVer:	0.1

Generator GUI

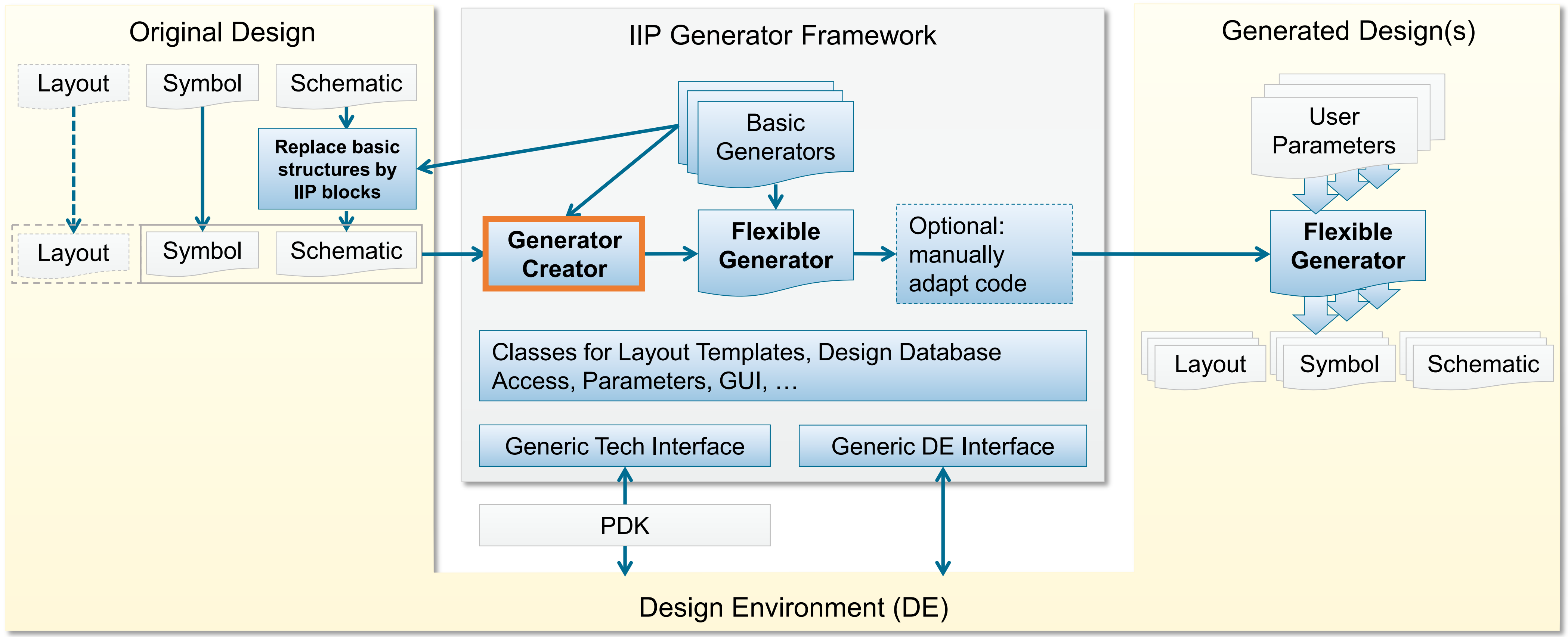


Template

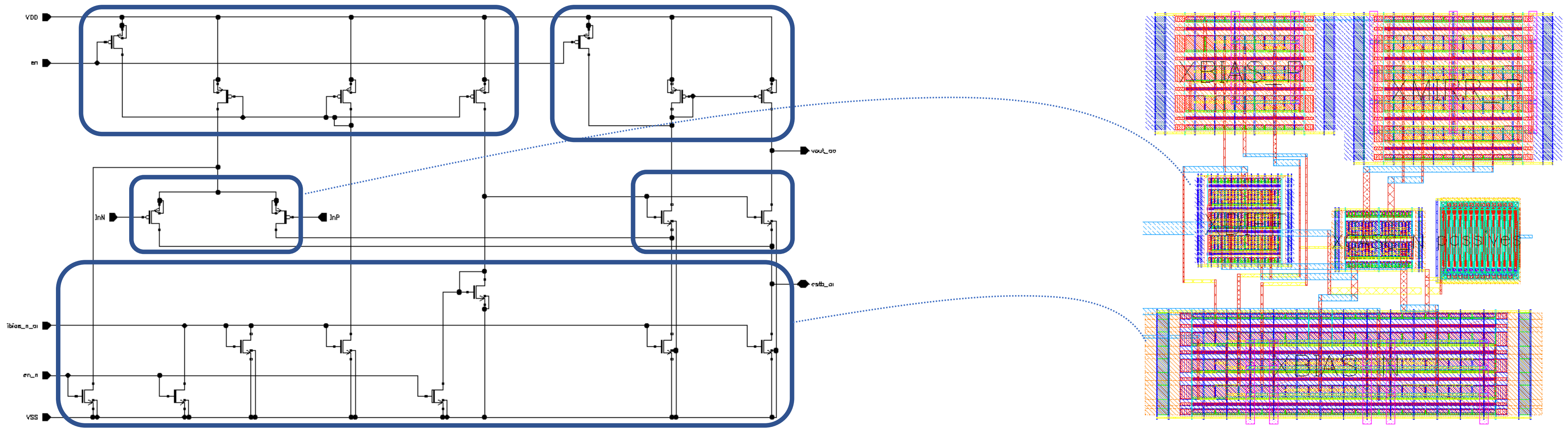


Generated layout

Automatic creation of generators (1)



Automatic creation of generators (2)

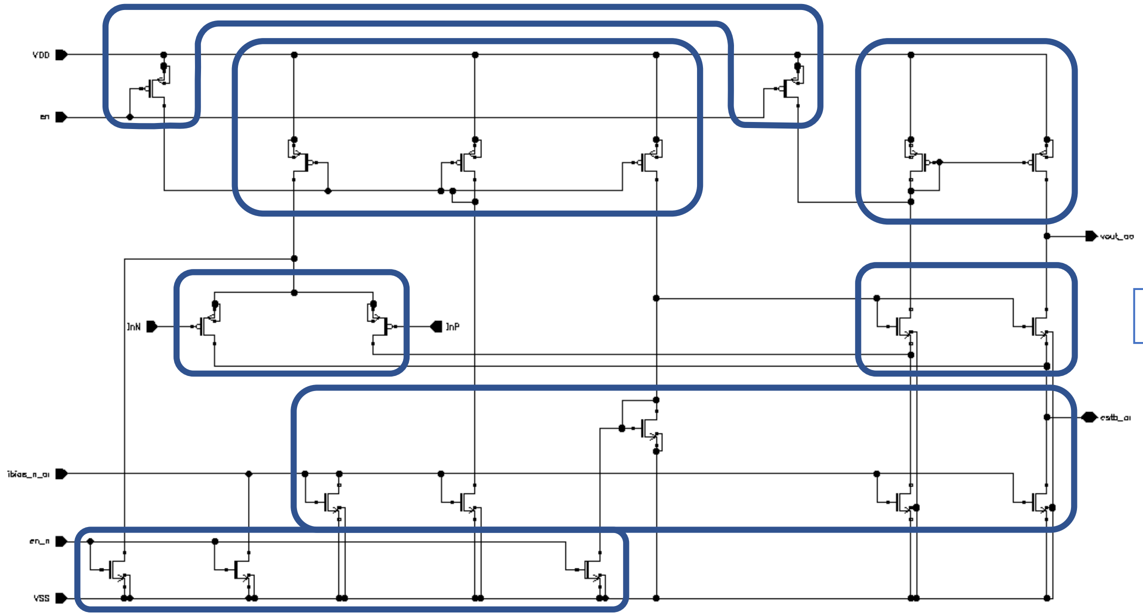


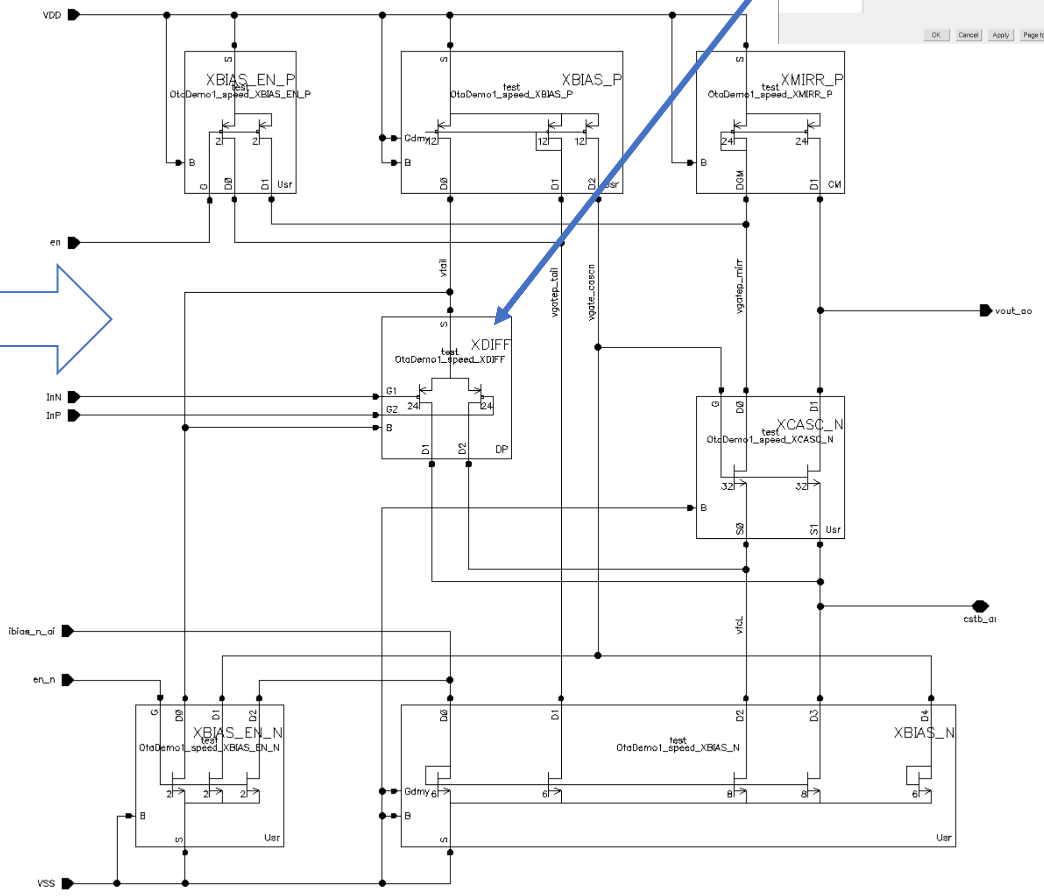
flat schematic

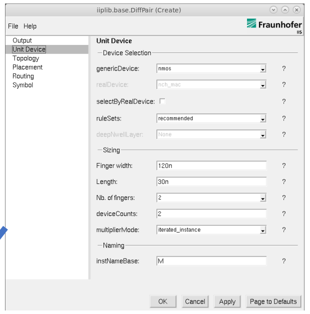
hier. layout
(generated by [4])

Orig. OTA design example by Infineon [6]

Automatic creation of generators (3)






iiplib.base.DiffPair (Create)

File Help

Output
Unit Device
Topology
Placement
Routing
Symbol

Unit Device

Device Selection

genericDevice: nmos
realDevice: nch_mac
selectByRealDevice: ☐
ruleSets: recommended
deepNwellLayer: None

Sizing

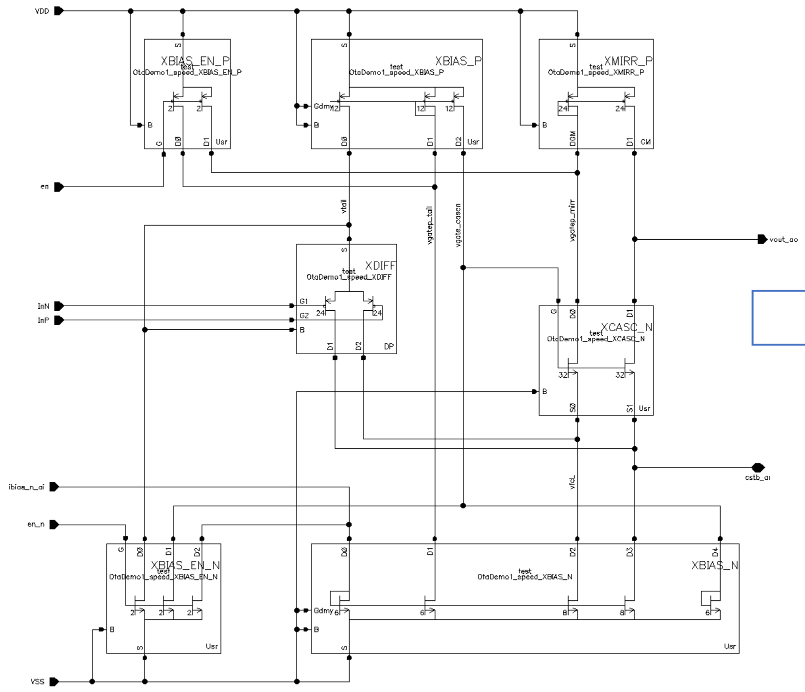
Finger width: 120n
Length: 30n
Nb. of fingers: 2
deviceCounts: 2
multiplierMode: iterated_instance

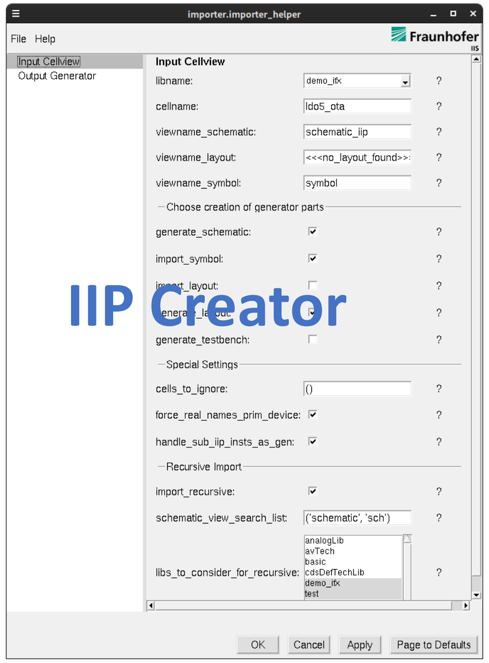
Naming

instNameBase: M

Replace basic structures by IIP blocks

Automatic creation of generators (4)




importer.importer_helper

File Help

Input Cellview
Output Generator

Input Cellview

libname: demo_ifx
cellname: ldo5_ota
viewname_schematic: schematic_iip
viewname_layout: <<<no_layout_found>>:
viewname_symbol: symbol

Choose creation of generator parts

generate_schematic: ✓
import_symbol: ✓
import_layout: ☐
generate_layout: ✓
generate_testbench: ☐

Special Settings

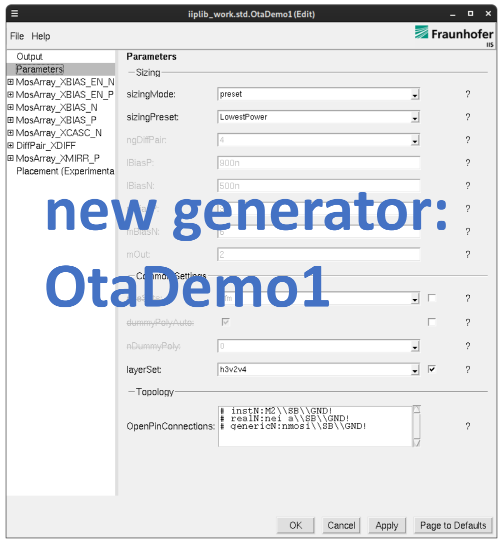
cells_to_ignore: ()
force_real_names_prim_device: ✓
handle_sub_iip_insts_as_gen: ✓

Recursive Import

import_recursive: ✓
schematic_view_search_list: ('schematic', 'sch')
libs_to_consider_for_recursive: analogLib, avTech, basic, cdsDefTechLib, demo_ifx, test

OK | Cancel | Apply | Page to Defaults

IIP Creator


iiplib_work.std.OtaDemo1 (Edit)

File Help

Output
Parameters
MosArray_XBIAS_EN_N
MosArray_XBIAS_EN_P
MosArray_XBIAS_N
MosArray_XBIAS_P
MosArray_XCASC_N
DiffPair_XDIFF
MosArray_XMIRR_P
Placement (Experimenta

Parameters

Sizing

sizingMode: preset
sizingPreset: LowestPower
ngDiffPair: 4
IBiasP: 900n
IBiasN: 500n
mBiasN: 6
mOut: 2

Common Settings

dummyPolyAuto: ✓
nDummyPoly: 0
layerSet: h3v2v4 ✓

Topology

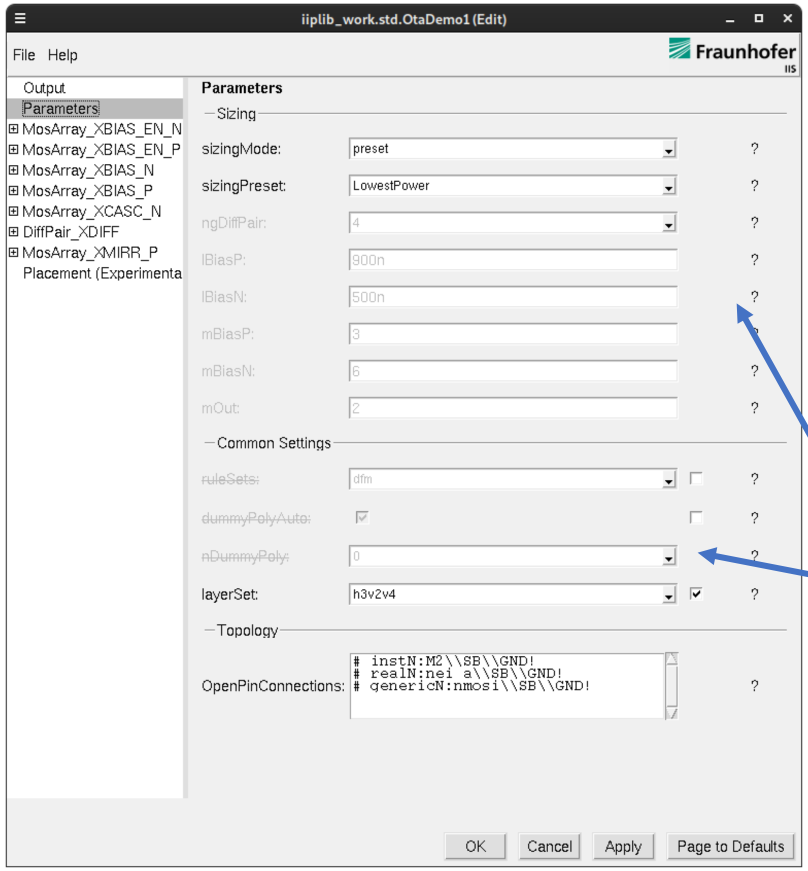
OpenPinConnections:
```
# instN:M2\\SB\\GND!
# realN:nei a\\SB\\GND!
# genericN:nmosi\\SB\\GND!
```

OK | Cancel | Apply | Page to Defaults

new generator: OtaDemo1

Use IIP Creator to translate available schematic data into a new generator

Automatic creation of generators (5)



iiplib_work.std.OtaDemo1 (Edit)

File Help

Output
Parameters
⊞ MosArray_XBIAS_EN_N
⊞ MosArray_XBIAS_EN_P
⊞ MosArray_XBIAS_N
⊞ MosArray_XBIAS_P
⊞ MosArray_XCASC_N
⊞ DiffPair_XDIFF
⊞ MosArray_XMIRR_P
Placement (Experimenta

Parameters

Sizing

sizingMode:	preset	
sizingPreset:	LowestPower	
ngDiffPair:	4	
IBiasP:	900n	
IBiasN:	500n	
mBiasP:	3	
mBiasN:	6	
mOut:	2	

Common Settings

~~ruleSets:~~	dfm	☐
~~dummyPolyAuto:~~	☑	☐
~~nDummyPoly:~~	0	
layerSet:	h3v2v4	☑

Topology

OpenPinConnections:

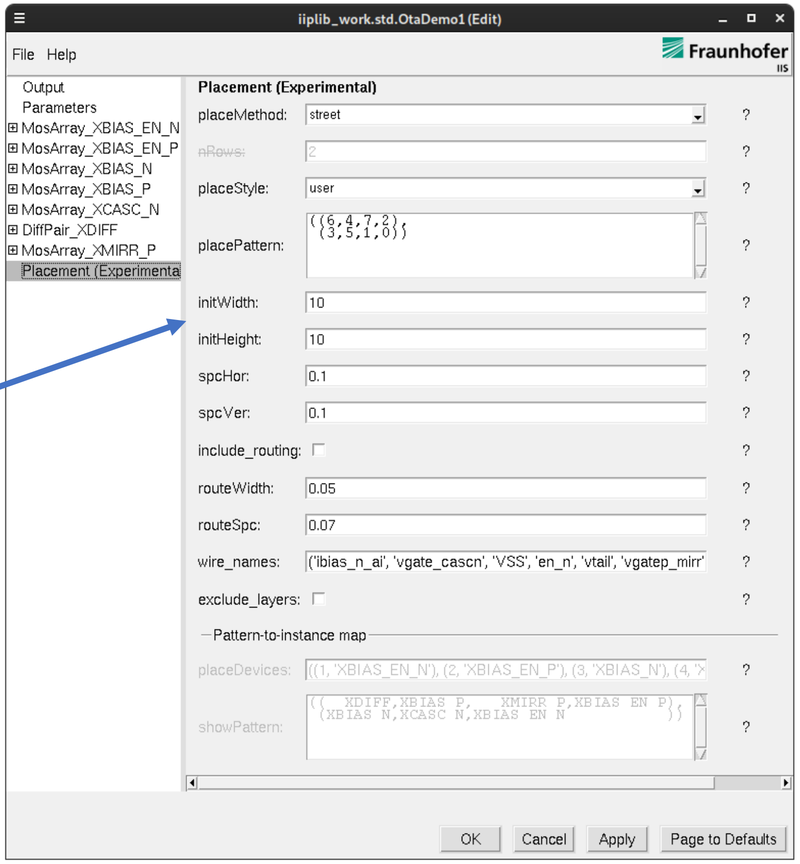
```
# instN:M2\\SB\\GND!
# realN:nei a\\SB\\GND!
# genericN:nmosi\\SB\\GND!
```

Auto-generated:

- inherited parameters from all sub-blocks
- Template P&R parameters

Manually added (optional):

- Predefined parameter sets for usability



iiplib_work.std.OtaDemo1 (Edit)

File Help

Output
Parameters
⊞ MosArray_XBIAS_EN_N
⊞ MosArray_XBIAS_EN_P
⊞ MosArray_XBIAS_N
⊞ MosArray_XBIAS_P
⊞ MosArray_XCASC_N
⊞ DiffPair_XDIFF
⊞ MosArray_XMIRR_P
Placement (Experimenta

Placement (Experimental)

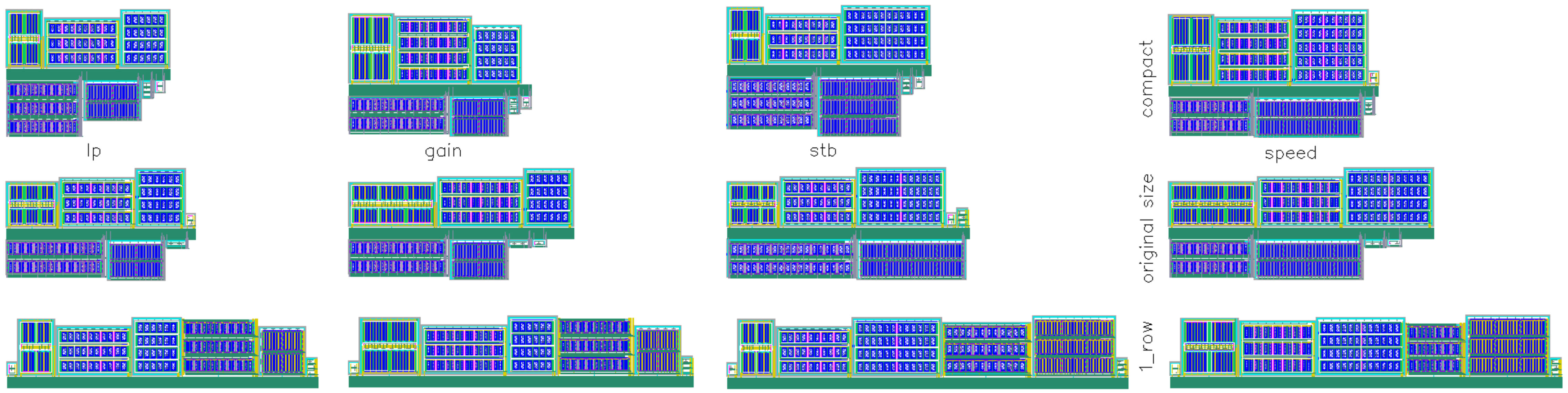
placeMethod:	street
~~nRows:~~	2
placeStyle:	user
placePattern:	((6,4,7,2), (3,5,1,0))
initWidth:	10
initHeight:	10
spcHor:	0.1
spcVer:	0.1
include_routing:	☐
routeWidth:	0.05
routeSpc:	0.07
wire_names:	('ibias_n_ai', 'vgate_cascn', 'VSS', 'en_n', 'vtail', 'vgatep_mirr'
exclude_layers:	☐

Pattern-to-instance map

placeDevices:	((1, 'XBIAS_EN_N'), (2, 'XBIAS_EN_P'), (3, 'XBIAS_N'), (4, 'X
showPattern:	((XDIFF,XBIAS P, XMIRR P,XBIAS EN P), (XBIAS N,XCASC N,XBIAS EN N))

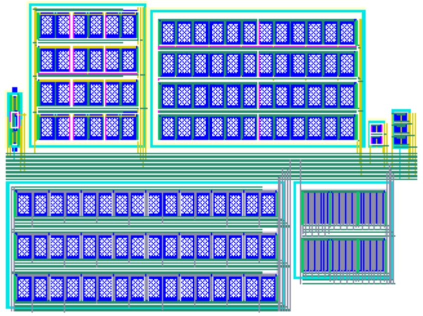
Generated layouts

- Original paper: 4 sizing variants, 3 layout variants

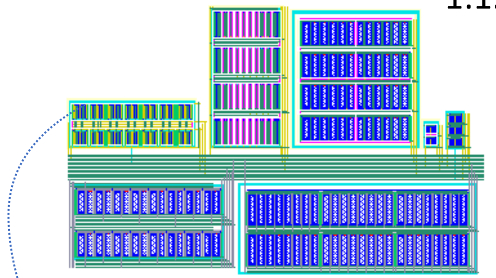


Generated layouts

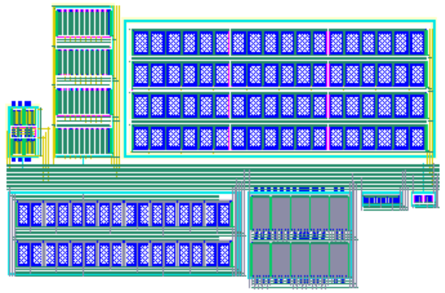
- New: updated testbench, 3 sizing variants, 2 layout variants



1:1.3 604um²



1:1.6 528um²

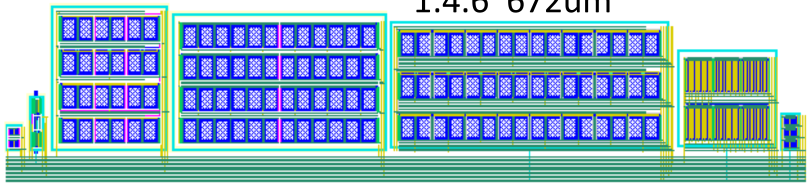


1:1.5 581um²

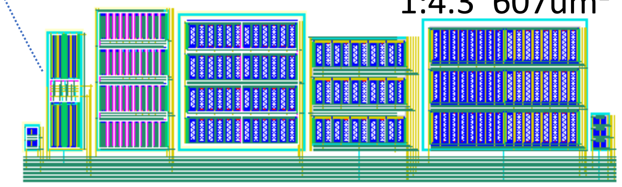
Low power

High gain

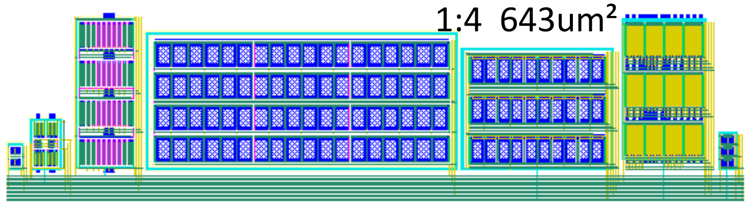
High speed



1:4.6 672um²



1:4.3 607um²

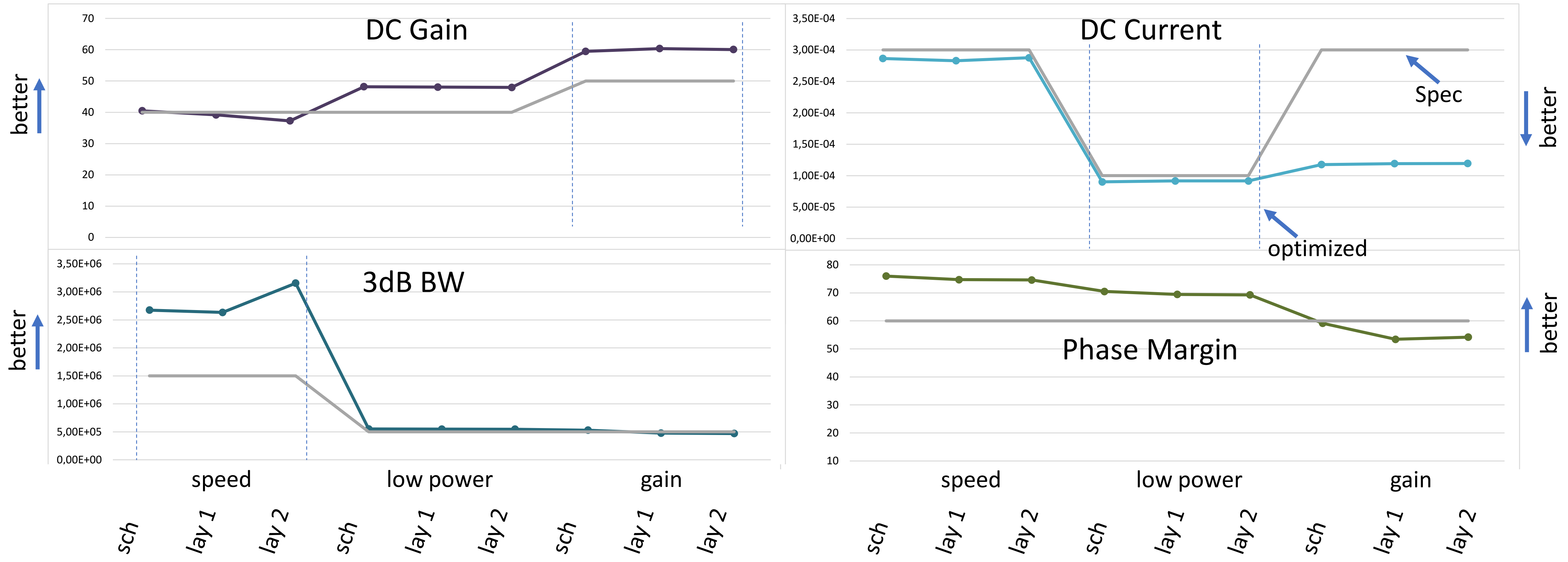


1:4 643um²

Selected performance results

- Generator runtime: ≈ 30 sec (dep. on PDK)
- DRC, LVS clean
- Aspect ratio: 1:1.3 .. 1:4.6, influence on bbox area: max. 15 %
- max. parasitic influence across all 6 layout variants:
  - 3dB BW: -11 .. +18 %
  - DC gain: -8 .. +2 %
  - DC current: -1 .. +2 %
  - Phase Margin: -10%

Selected performance results



Summary

- Template-based layouts are feasible for this (and more) types of circuits
- automatic creation of template-based generators may close the gap between generators for basic building blocks and designer-driven layout reuse and synthesis for more complex circuits
- IIP Creator approach allows fast transfer of existing design IP into a much more generic and reusable format

Outlook

- Parameter abstraction for aspect ratio, place pattern, routing, sizing
  → early investigation of layout effects, even before simulation
  → layout-aware sizing/optimization
- Better integration into simulation / optimization tool flows
  - In progress: [7] [8]
  - Parameter interfaces to Cadence ADE, MunEDA WiCkeD, Intento ID-Xplore, …
- Standardization of a common generator description language
  → interoperability across generator suppliers
- further development driven by public funded projects: AnastASICA, InnoStar, HoLoDEC

References

Thanks to Husni Habal et al. from Infineon Technologies for providing the original design and many discussions!

This work was funded by the German Federal Ministry of Education and Research BMBF within project InnoStar.

[1] Cadence PCell Designer: https://www.cadence.com

[2] Synopsys PyCell Studio: https://www.synopsys.com

[3] IPGen 1Stone: A. Graupner, R. Jancke und R. Wittmann, “Generator Based Approach for Analog Circuit and Layout Design and Optimization,“ DATE 2011

[4] Berkeley Analog Generator: E. Chang et al., “BAG2: A Process-portable Framework for Generator-based AMS Circuit Design,“ CICC 2018

[5] R. Castro-López et al., "An Integrated Layout-Synthesis Approach for Analog ICs," IEEE TCAD, 2008

[6] F. Passerini et al., “ANAGEN: A Methodology for ANAlog Circuit GENeration”, ESSCIRC/ESSDERC, 2021

[7] B. Prautsch, R. Iskander et al., "Automatic Generation of Multiple GF-22FDX OpAmp Variants and Layouts in the Virtuoso® Suite by Incorporating ID-Xplore™ of Intento Design with IIP of Fraunhofer IIS/EAS", CadenceLIVE, 2022

[8] B. Prautsch et al., "A Multi-level Analog IC Design Flow for Fast Performance Estimation Using Template-based Layout Generators and Structural Models," SMACD 2022

Thank you

Questions?