Automated Mixed-Signal PHY Generation of the AIB Die-to-Die Interface

Krishna Settaluri, Blue Cheetah Analog Design, Inc.

David Kehlet, Intel Corporation

This work is supported by the DARPA MTO office [DARPA CHIPS Contract Number: HR00111790020].

Legal Notices and Disclaimers

This presentation contains the general insights and opinions of Intel Corporation ("Intel"). The information in this presentation is provided for information only and is not to be relied upon for any other purpose than educational. Statements in this document that refer to Intel's plans and expectations for the quarter, the year, and the future, are forward-looking statements that involve a number of risks and uncertainties. A detailed discussion of the factors that could affect Intel's results and plans is included in Intel's SEC filings, including the annual report on Form 10-K.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at www.intel.com.

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

† Tests measure performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Agenda: Automated Mixed-Signal PHY Generation of AIB

- Motivations for Chiplets and Automated Analog Design
- Generator-Based Design Methodology and Application to AIB
- AIB Generator and Open Source Hardware
- AIB 2.0
- Conclusion

Why Are Chiplets Interesting Now?

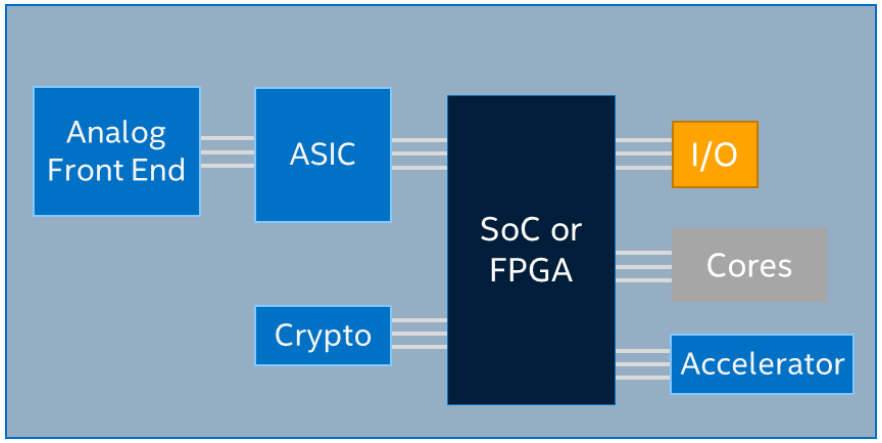
- Heterogeneous integration advancements
  - Mix foundries, process nodes, IP sources, etc.
- Time-to-market advantages
  - No need to port everything to same node
- Advanced packaging technology
  - Enables high die-to-die bandwidth
- Open source standards like AIB encourage chiplet ecosystem

"It may prove to be more economical to build large systems out of smaller functions, which are separately packaged and interconnected[1]."

Gordon E. Moore

1: 3rd Page of Moore's 1965 paper, "Cramming more components onto integrated circuits"

Image: Intel



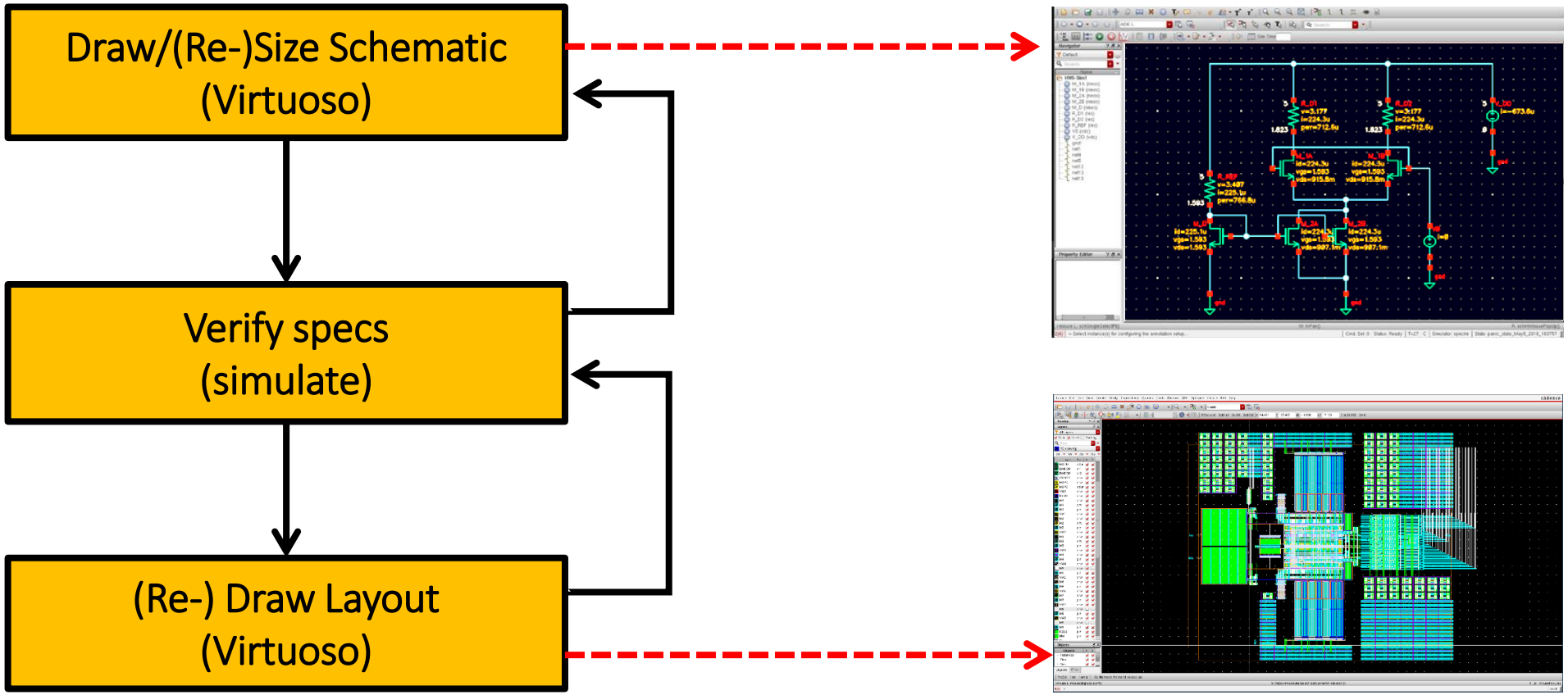
Barriers to Entry in Developing Custom Silicon

- Rapid generation of IP, across many technology nodes = catalyst for chiplet / AIB ecosystem
  - **Barriers** -- Labor, time-to-market, risk profile, broad expertise
  - **Focus** -- For many in industry, analog design is an exemplar of the above barriers
- Reuse of **instance** is a myth!
  - Features, performance, specifications, design attributes, process **change** every time.

Solution: Don't just develop instances – re-use designer knowledge and practices in generators!

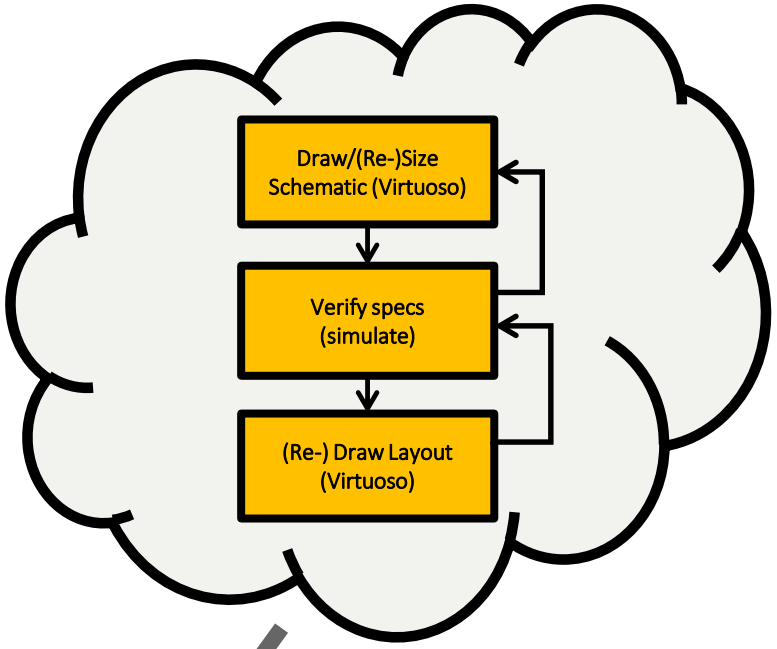
Why is Analog Design a Bottleneck?

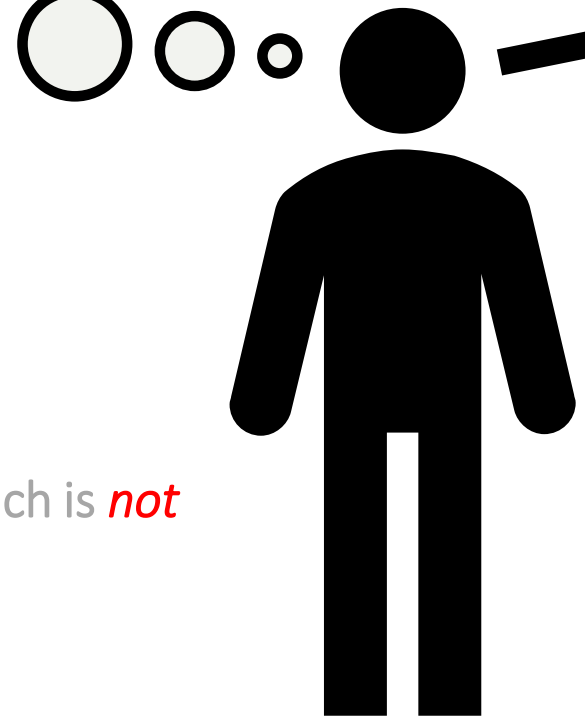
- Core design loop hasn't really changed for 30+ years



Images: Blue Cheetah

Generator-Based Design





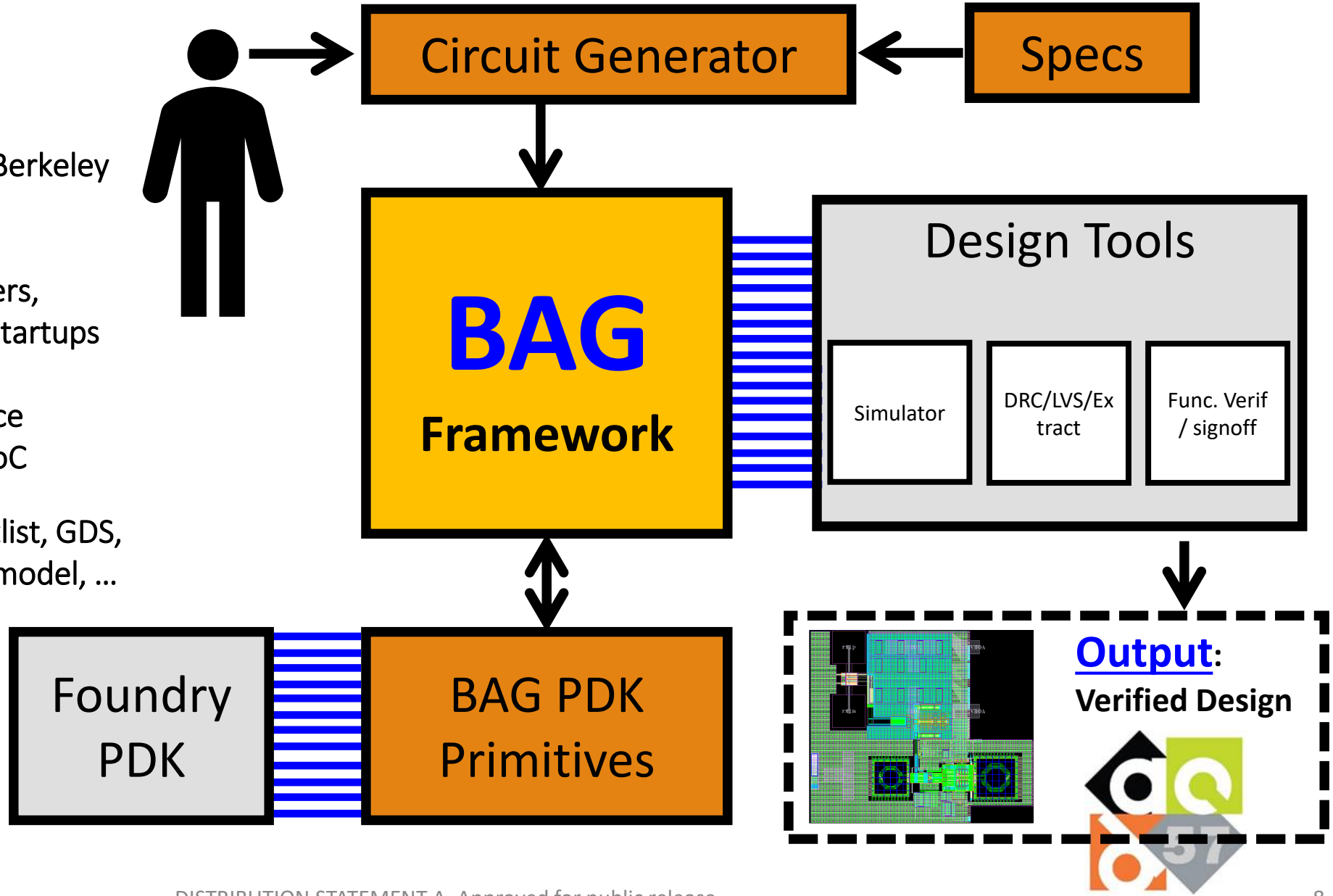
Capture knowledge in executable generator
Re-use design methodology

Enable parameterization, process migration, risk reduction

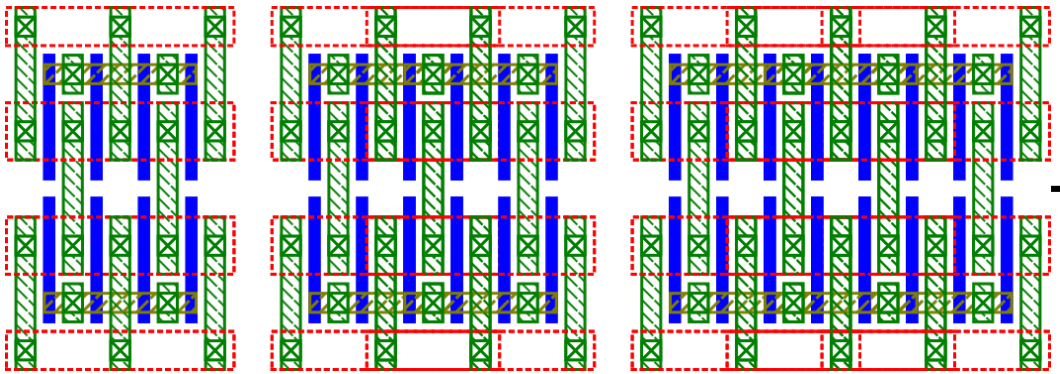
Blue Cheetah's approach is *not*
- Optimization engine
- Place-and-router
- GDS porter

Key Technology Enabler: **BAG** Framework

- Based on work done at UC Berkeley starting in 2012[1]
- Adopted by many researchers, universities, companies, & startups
- Powerful features to produce instances + integrate into SoC
- Full generators produce netlist, GDS, LEF file, LIB file, behavioral model, …
- Blue Cheetah is leader in Framework development and driving commercialization



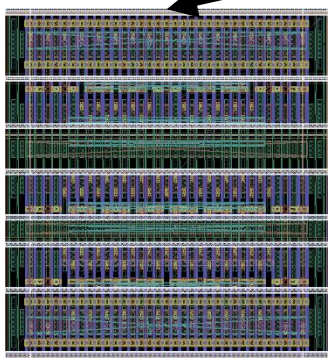
Capturing Floorplan Concept



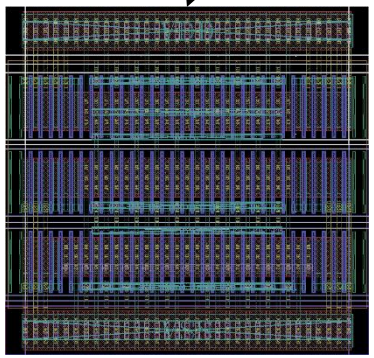
Floorplan Concept

```
# specify width/threshold of each row
nw_list, nth_list = [w_dict['amp']], [intent_dict['amp']]  # NMOS
pw_list, pth_list = [w_dict['load']], [intent_dict['load']]  # PMOS
# specify number of horizontal routing tracks for each row
ng_tracks, nds_tracks = [1], [1]  # NMOS gate and drain/source tracks
pg_tracks, pds_tracks = [1], [1]  # PMOS gate and drain/source tracks
# specify row orientations
n_orient = ['R0']  # gate connection on bottom
p_orient = ['MX']  # gate connection on top
# draw transistor rows
self.draw_base(lch, fg_tot, ptap_w, ntap_w, nw_list,
               nth_list, pw_list, pth_list,
               ng_tracks=ng_tracks, nds_tracks=nds_tracks,
               pg_tracks=pg_tracks, pds_tracks=pds_tracks,
               n_orientations=n_orient, p_orientations=p_orient,
               )
# ...
# create transistor connections
amp_ports = self.draw_mos_conn('nch', 0, amp_col, fg_amp, nsdir, nddir)
load_ports = self.draw_mos_conn('pch', 0, load_col, fg_load, 2, 0)
# ...
# connect transistors to routing tracks
vin = self.connect_to_tracks(amp_ports['g'], vin_tr)
vout = self.connect_to_tracks([amp_ports['d'], load_ports['d']], vout_tr)
vbias = self.connect_to_tracks(load_ports['g'], vbias_tr)
```

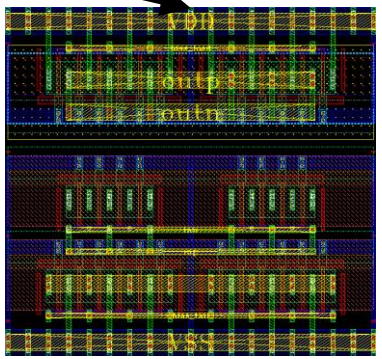
Layout Generator Script



FinFET



Bulk

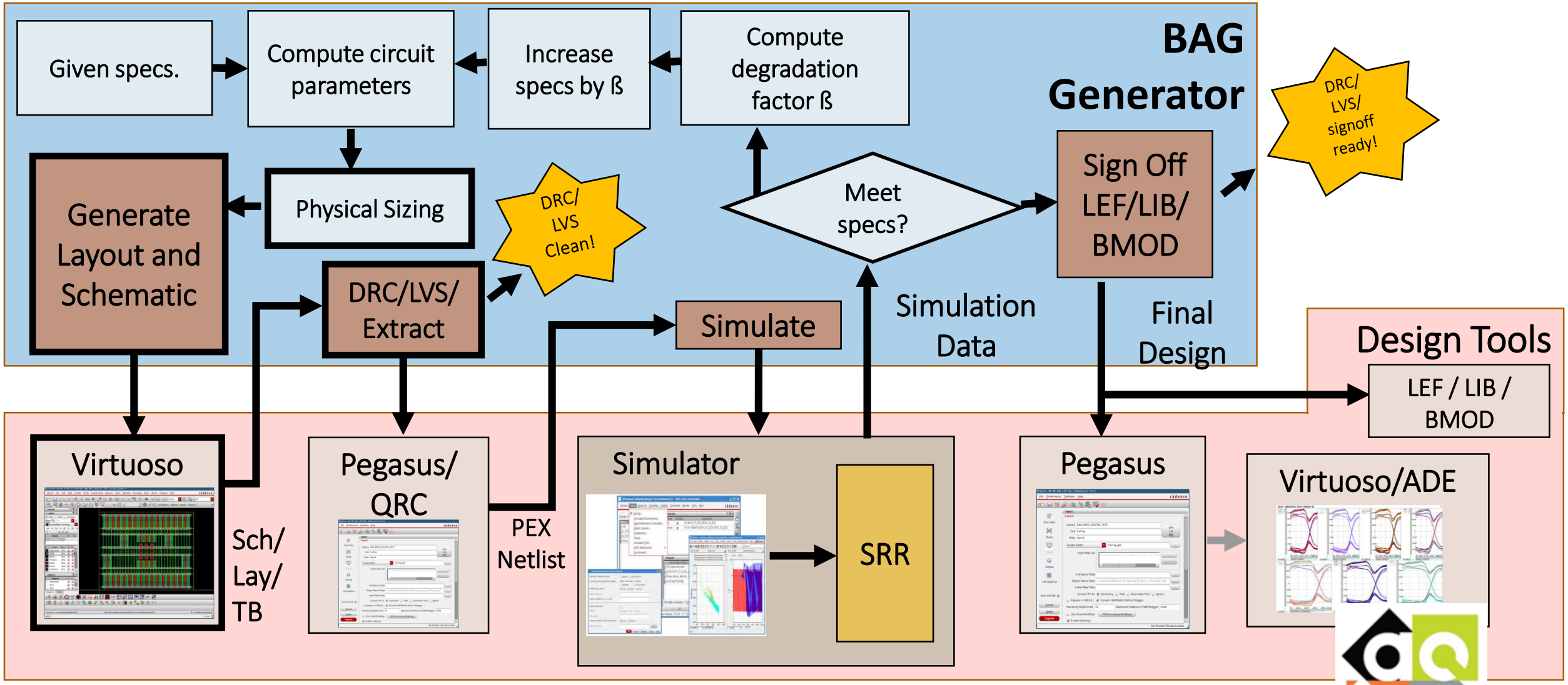


SOI

- Key: floorplan concept is process portable
- BAG APIs support allowable operations with given floorplan concept
  - Generator writer uses API to encode layout strategy

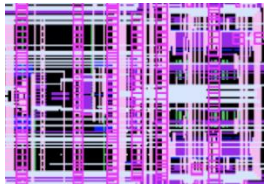
Chang, Eric, et al. "BAG2: A process-portable framework for generator-based AMS circuit design." *2018 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 2018.

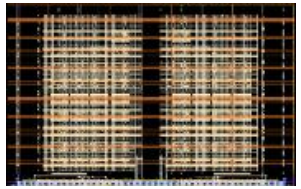
End-to-End Generator



Generator Instances & Hierarchy (UCB)

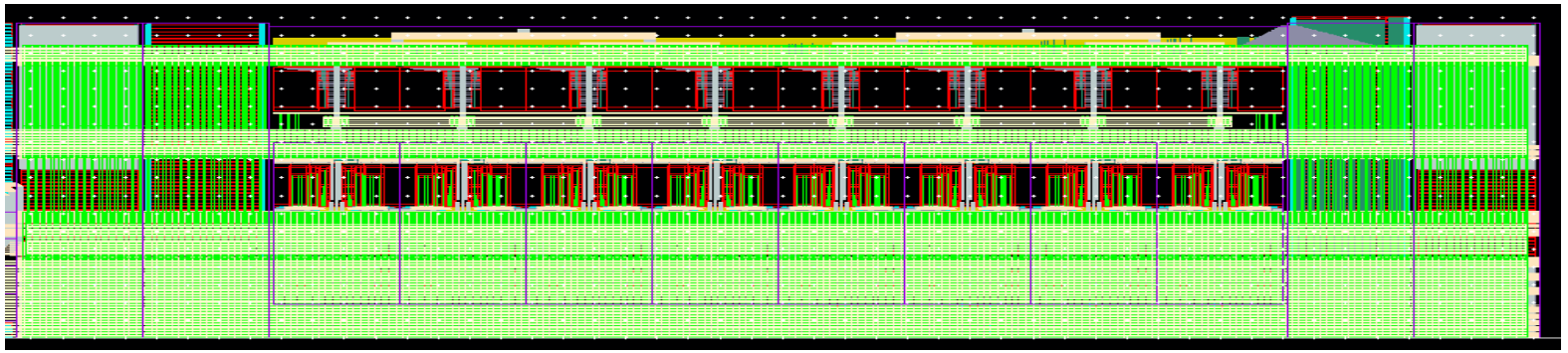
Active CTLE



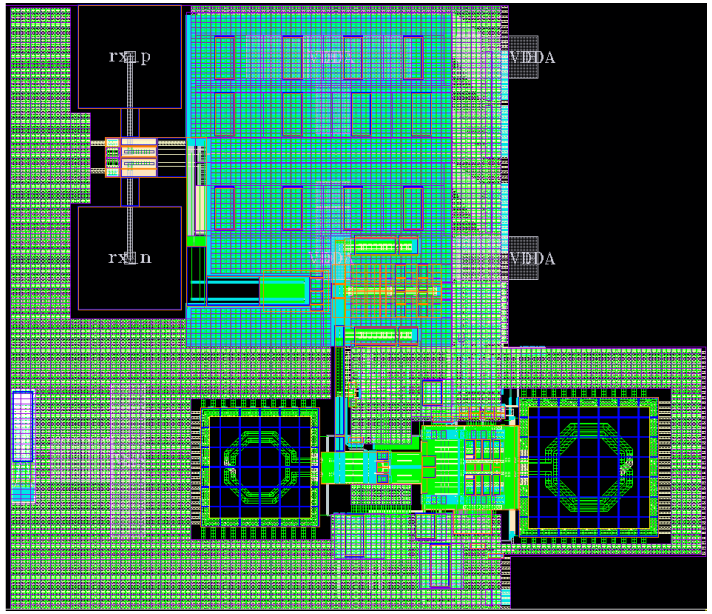


Switch-Cap DAC

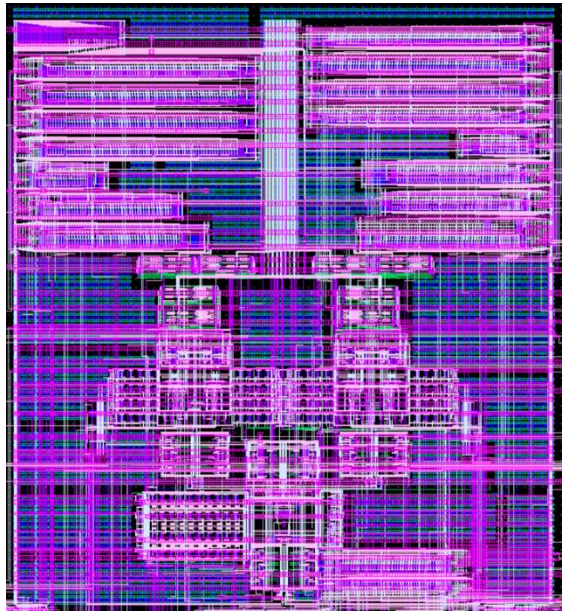
Comparator



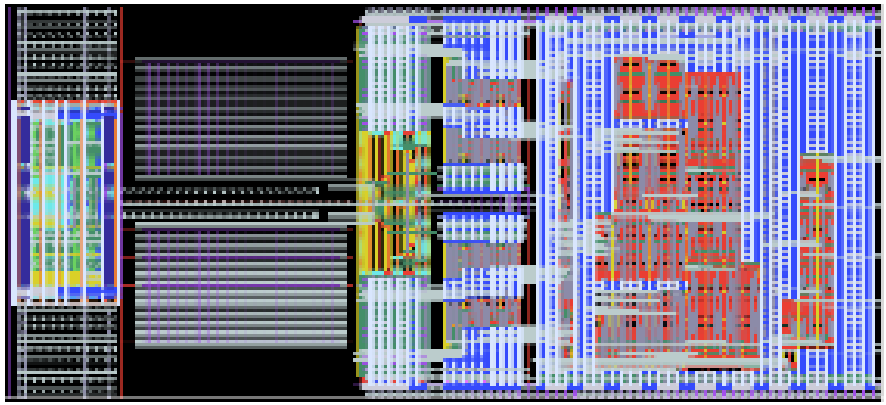
Time-Interleaved SAR ADC



Full SerDes RX



Photonic Receiver AFE

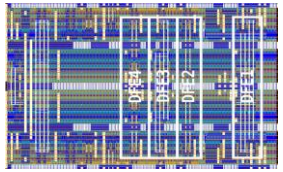


SAR ADC

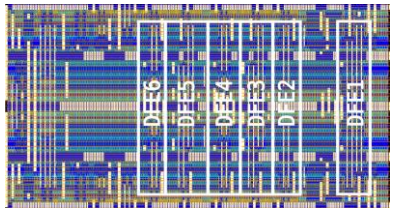
Chang, Eric, et al. "BAG2: A process-portable framework for generator-based AMS circuit design." *2018 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 2018.
Settaluri, K. T. (2018). *Photonics Links--From Theory to Automated Design* (Doctoral dissertation, UC Berkeley).

Power of Generators

Advanced FinFET SerDes RX



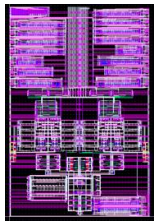
4-tap DFE

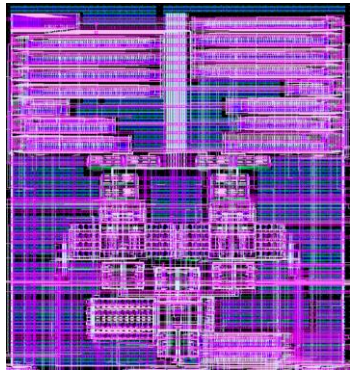


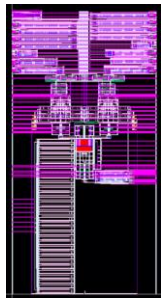
6-tap DFE

Parameterization:
Change # of taps and re-execute generator

PDSOI PAM4 RX







Parameterization:
Change bump pitch and re-execute generator

SerDes Rx Datapath Generator Script

With Process X Primitives Library →

With Process Y Primitives Library →

- Design effort of generator ~ design effort of instance #1
- Using Generator, instance #2 and beyond ~ factor of 10X+ reduction in labor and time

Chang, Eric, et al. "An Automated SerDes Frontend Generator Verified With a 16-nm Instance Achieving 15 Gb/s at 1.96 pJ/bit." *IEEE Solid-State Circuits Letters* 1.12 (2018): 245-248.
Chang, Eric, et al. "BAG2: A process-portable framework for generator-based AMS circuit design." *2018 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 2018.
Settaluri, K. T. (2018). *Photonics Links--From Theory to Automated Design* (Doctoral dissertation, UC Berkeley).

Blue Cheetah AIB PHY Generator

- **Generator for the AIB PHY**
  - Full generators for all custom blocks (e.g.. driver, phase interp., etc.)
  - 2-channel test chip taped out in Intel 22FFL Jan 2020
  - Blue Cheetah is already leveraging AIB generators in follow-on projects; generators released to open source

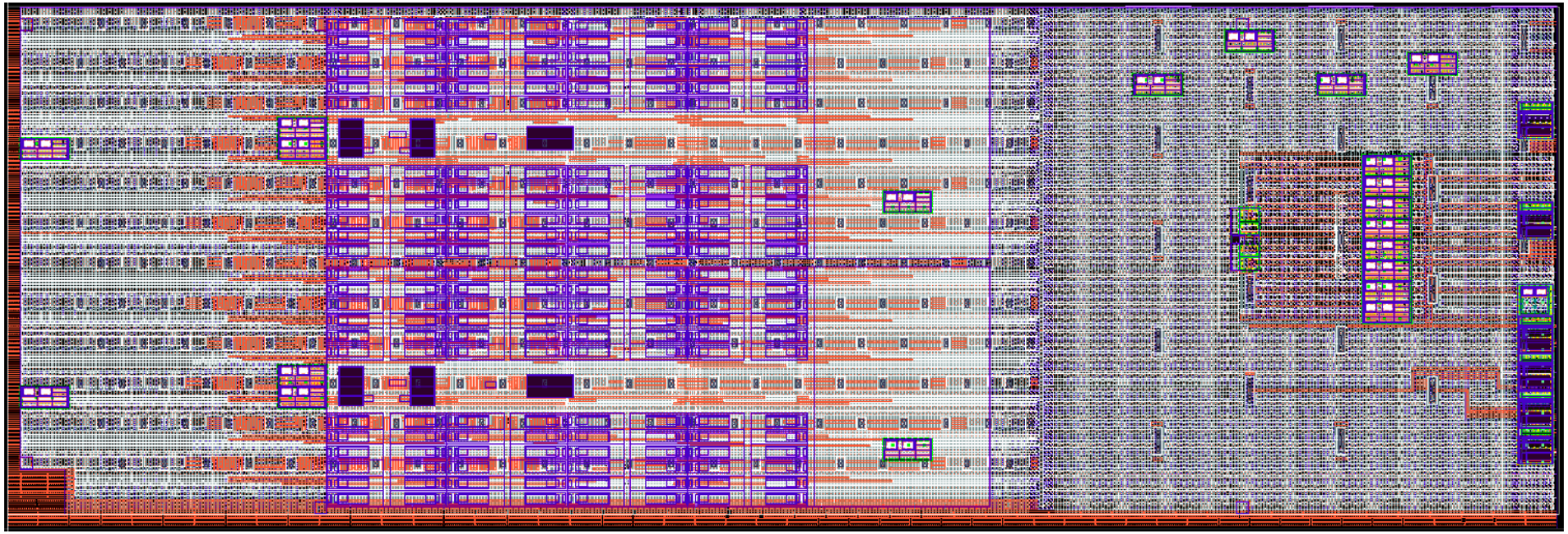
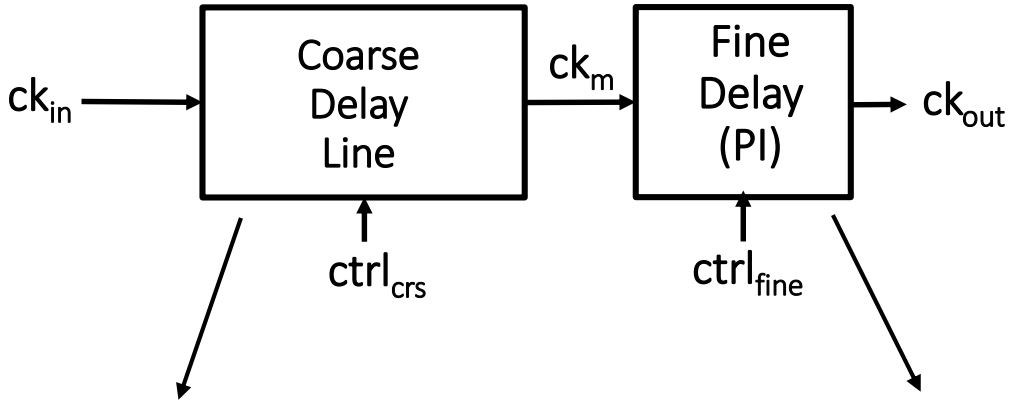
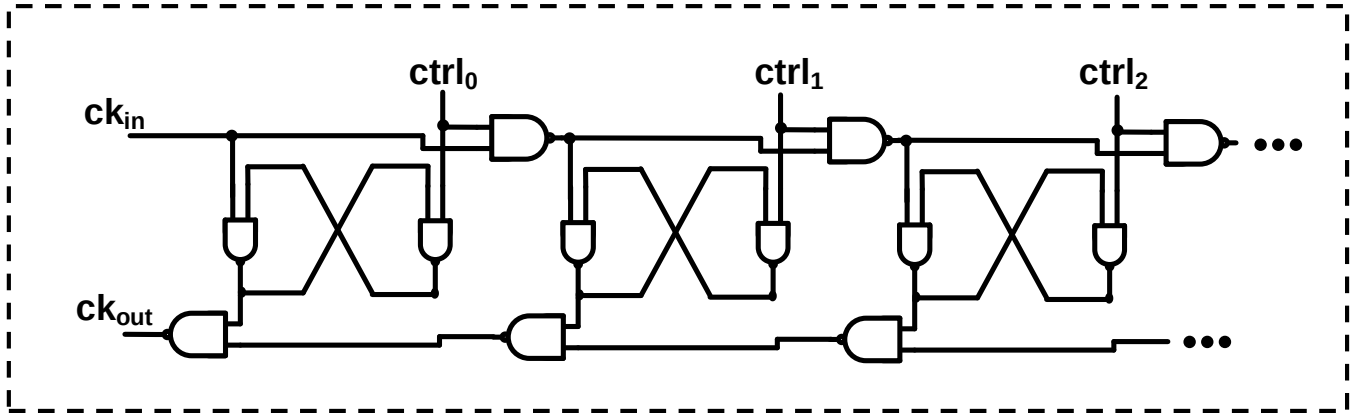


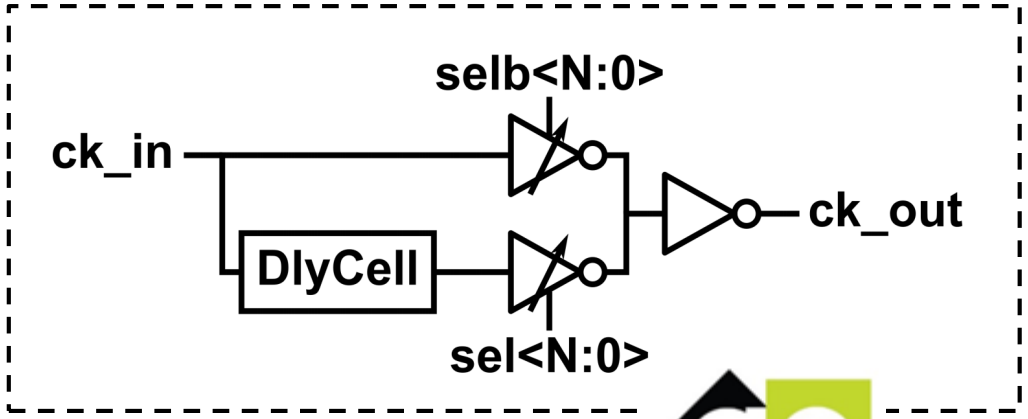
Image: Blue Cheetah

Example AIB Sub-block Generator: Delay Line

- Core of both DCC (for Tx) and DLL (for Rx)



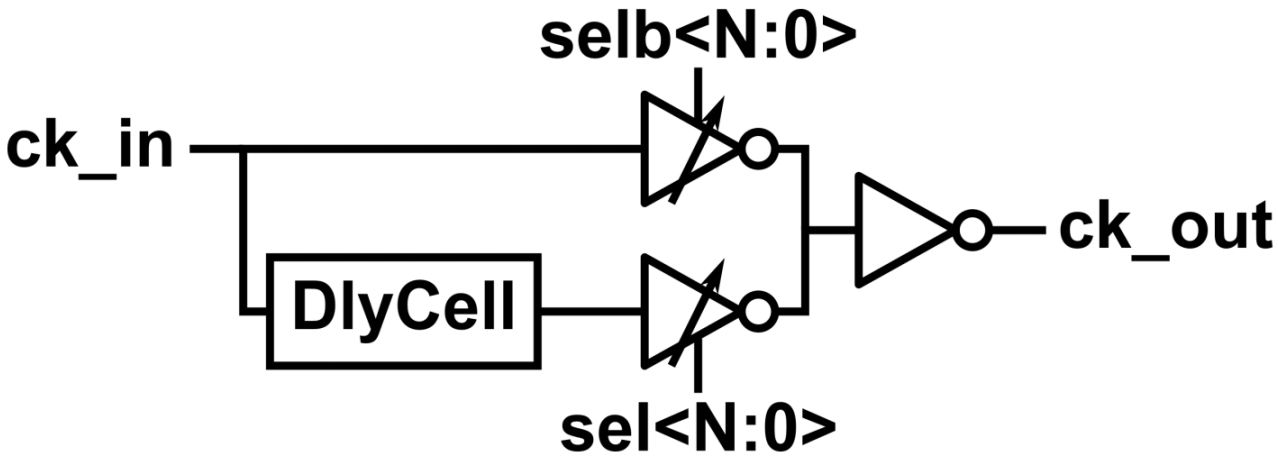




AIB Custom Block Generator Example: Phase Interpolator (Fine Delay)

- **Top-level specifications:**
  - $t_{d,step,min}$, $t_{d,step,max}$, $t_{p,max}$, $s_{td,step}$
  - # of delay steps

- **End-to-end generator: executable code produces netlist, GDS, LIB, LEF, and behavioral model**
  - All sizing decisions needed to achieve top-level specs made in code based on feedback from simulated (post-layout) results



Generated Phase Interpolator Instances

Input yaml

Generated Layout

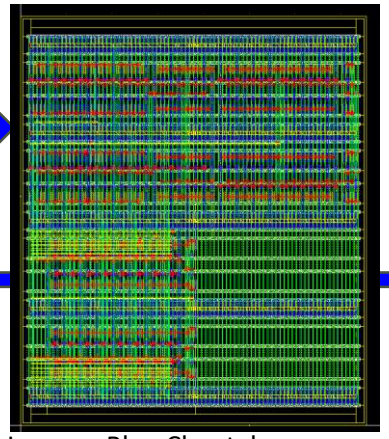
Simulated Results

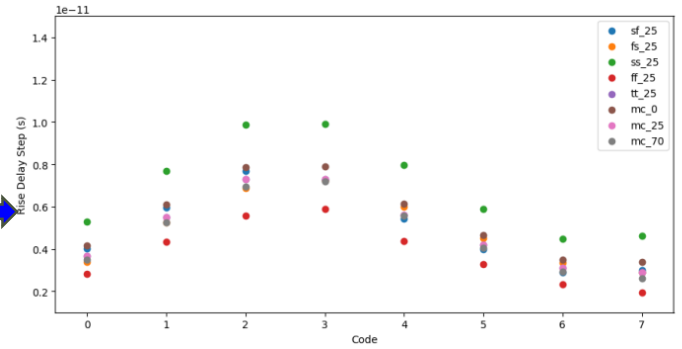
8 Steps

```yaml
dsn_params:
  target:
    td_min: 1.0e-12
    td_max: 15.0e-12
    td_sigma: 0.25e-12
    t_max: 150.0e-12

  tristate_seg: 2
  nbits: 7
  seg_buf_min_override: 2
  seg_buf_max_override: 50
  design_using_signoff: True

  rtol: 1.0e-3
  atol: 1.0e-3
  tbit: 1.0e-9
  trf: 20.0e-12
  cload: 5.0e-15
  plot_result: True
```





Generator Code

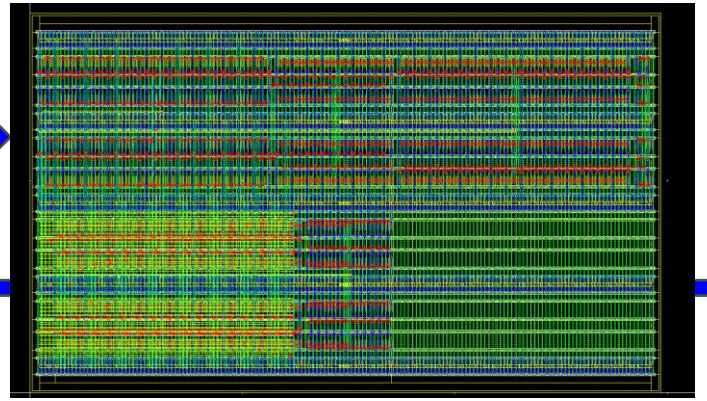
Images: Blue Cheetah

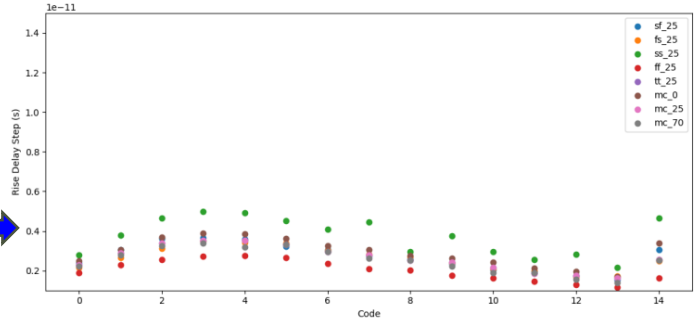
15 Steps

```yaml
dsn_params:
  target:
    td_min: 1.0e-12
    td_max: 15.0e-12
    td_sigma: 0.25e-12
    t_max: 250.0e-12

  tristate_seg: 2
  nbits: 14
  seg_buf_min_override: 2
  seg_buf_max_override: 50
  design_using_signoff: True

  rtol: 1.0e-3
  atol: 1.0e-3
  tbit: 1.0e-9
  trf: 20.0e-12
  cload: 5.0e-15
  plot_result: True
```

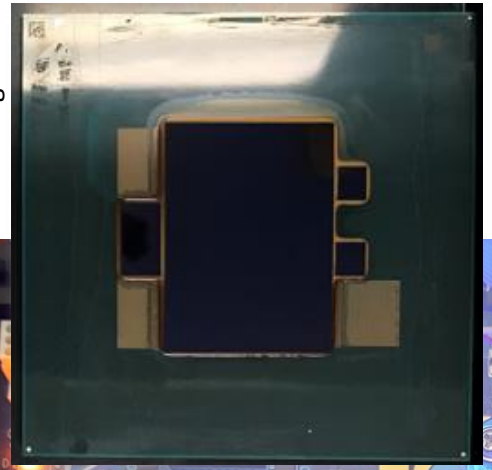




(designs shown on cds_ff_mpt)

AIB Generated Interface Hardware Success!

Image: Intel



```
RX status for channel 9
chan=9 lane= 0 RXdata=0x7EE2A299C5684CA76612C1629FDE4BB3 bits=  1,019,172,419,980 errs=          0 ber=<9.81e-13
chan=9 lane= 1 RXdata=0xE5646D83BD64512319F66FB0354455DB bits=  1,019,461,769,164 errs=          0 ber=<9.81e-13
chan=9 lane= 2 RXdata=0xCC61115889FD90A5A71825F5F03698F5 bits=  1,019,735,316,064 errs=          0 ber=<9.81e-13
chan=9 lane= 3 RXdata=0xB6035051449424264109 64D64B7D650D bits=  1,020,019,163,220 errs=          0 ber=<9.80e-13
chan=9 lane= 4 RXdata=0x745DD77F098AF0E2581701FBC90FFC78 bits=  1,020,306,084,048 errs=          0 ber=<9.80e-13
chan=9 lane= 5 RXdata=0x8D9ED4B7E7C50863B89936078110600E bits=  1,020,582,015,552 errs=          0 ber=<9.80e-13
chan=9 lane= 6 RXdata=0xD2D75C18B3D43FC86AF6AAC9E8F333EE bits=  1,020,860,844,920 errs=          0 ber=<9.80e-13
chan=9 lane= 7 RXdata=0x0CFC1EFDAC76E1C33F0C4ADDC6C08518 bits=  1,021,156,358,272 errs=          0 ber=<9.79e-13
chan=9 lane= 8 RXdata=0x78BF562D5A399FAF5F8355E2E36BD1FA bits=  1,021,422,326,504 errs=          0 ber=<9.79e-13
chan=9 lane= 9 RXdata=0xDF5676E51D0D4566E6C2D0D7289748B9 bits=  1,021,711,255,440 errs=          0 ber=<9.79e-13
chan=9 lane=10 RXdata=0x4F6ADC89ABA18811784A5B4B80843087 bits=  1,022,007,677,548 errs=          0 ber=<9.78e-13
chan=9 lane=11 RXdata=0x589862CD135BB3E6F9879561CA47D0D8 bits=  1,022,290,662,440 errs=          0 ber=<9.78e-13
chan=9 lane=12 RXdata=0x9BB24A535CFBE9943E31EEF6A94A8C61 bits=  1,022,568,045,872 errs=          0 ber=<9.78e-13
chan=9 lane=13 RXdata=0xF955E94C7D0A82C0E6CC976C453EF0C0 bits=  1,022,846,107,480 errs=          0 ber=<9.78e-13
chan=9 lane=14 RXdata=0xD74C4A3AF089A07AB7EE92A1473A684B bits=  1,023,144,137,220 errs=          0 ber=<9.77e-13
chan=9 lane=15 RXdata=0xE65CFA23BBC1CDF87891AE3637D856BB bits=  1,023,427,887,788 errs=          0 ber=<9.77e-13
chan=9 lane=16 RXdata=0x376BABD94615EAEF602E1739FFA90F83 bits=  1,023,710,087,304 errs=          0 ber=<9.77e-13
chan=9 lane=17 RXdata=0x1EB6C45FC5D7643898B9F6A58158F32E bits=  1,024,007,413,384 errs=          0 ber=<9.77e-13
chan=9 lane=18 RXdata=0x66F076D44549D74099A7464835F09F7F bits=  1,024,301,639,596 errs=          0 ber=<9.76e-13
chan=9 lane=19 RXdata=0xD4A136E8BE9061ECE100D8C1FDB75AD8 bits=    512,296,826,296 errs=          0 ber=<1.95e-12
```

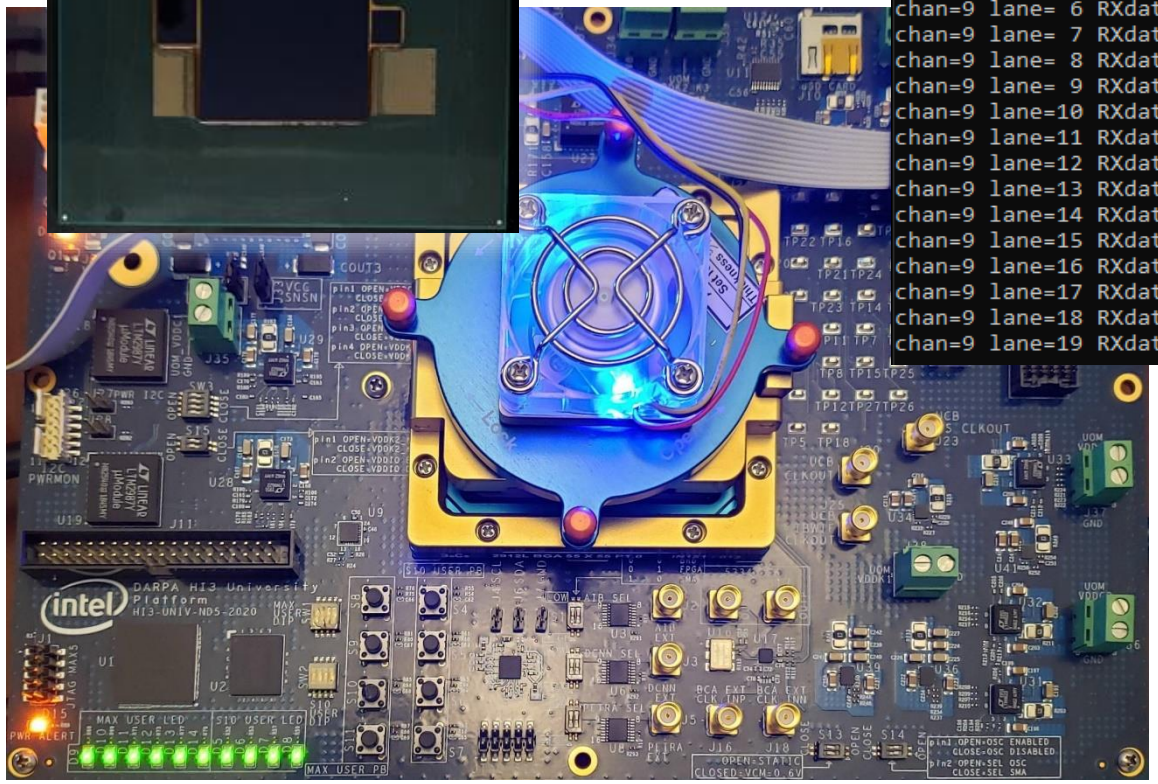
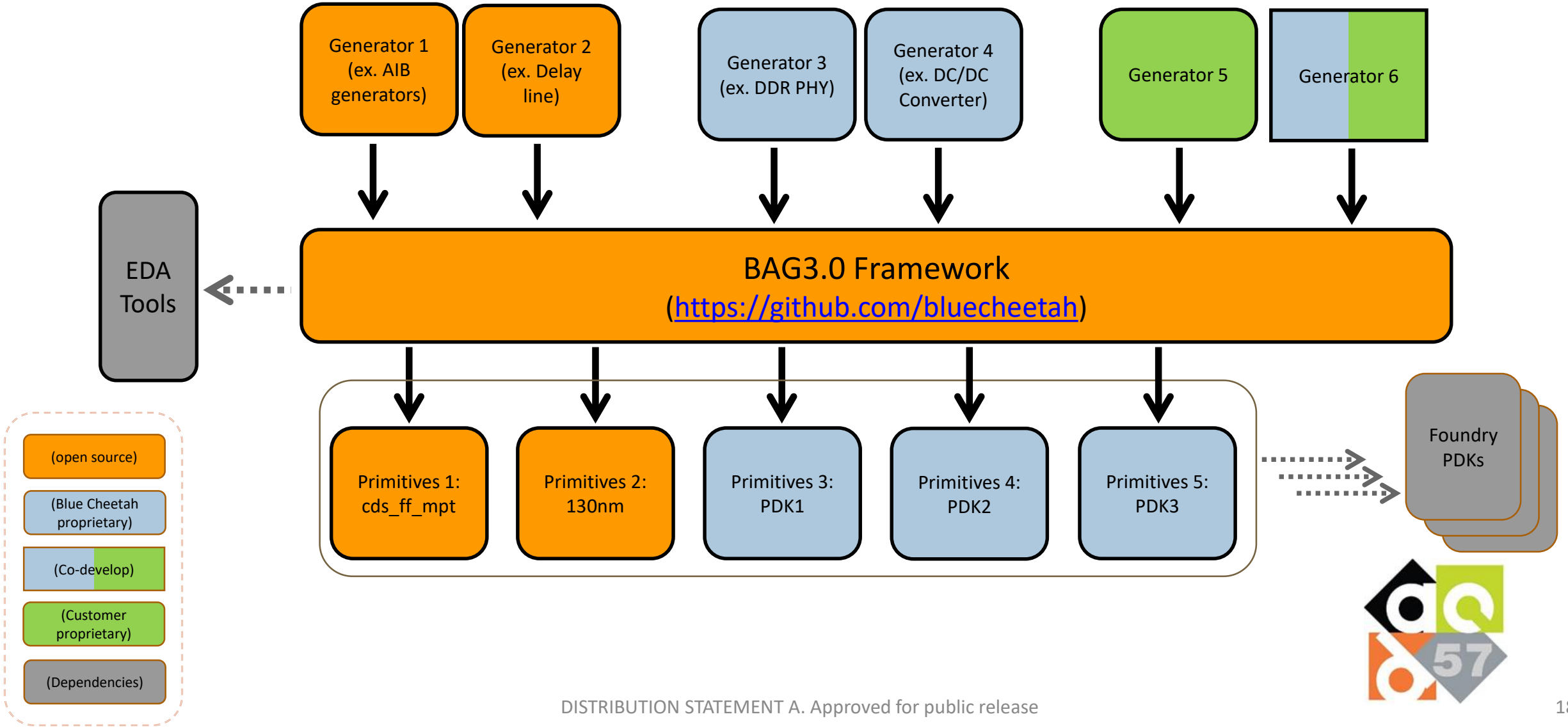


Image: Blue Cheetah

Multi-Chip Package and Board

- Zero error loopback results @ 2Gbps!
- Test Chip Tx → Stratix 10 Rx → Stratix 10 Tx → Test Chip Rx
- Simultaneous offset PRBS-31 patterns on each lane

Blue Cheetah Ecosystem



AIB Generator and Open Source Hardware

Open Source Hardware?

- Open source user's point of view, pro and con:
  - "[The CHIPS Alliance is addressing] the problem of high expenses of chipmaking and what can be done in the open source domain to reduce the cost." – *Zvonimir Bandic, Chair of the CHIPS Alliance, Sr. Director at Western Digital*[1]
  - "Replacing proprietary IP license costs with open-source IP licenses will only reduce these [SoC development] costs by ~1%-20%." – *Brucek Khailany, Director of Research at NVidia*[2]
  - **Reuse** to lower the cost of hardware development
- Open source provider's point of view:
  - Enable other companies' silicon to interoperate with my silicon
  - Address an expanded set of customer requirements

Intel FPGA with two Ayar Labs TeraPHY™ optical I/O chiplets

Image: Intel

Fully packaged FPGA with Ayar Labs optical I/O

Image: Ayar Labs, Inc. Used with permission.

1. "Five Minutes with Zvonimir Bandic," Embedded Computing Design, May 2019
2. Khailany, Brucek, "ISSCC 2020 Panel: Open Source Hardware," IEEE International Solid-State Circuits Conference, February 2020

AIB Generator Open Source Hardware

- AIB PHY Generator builds on the AIB PHY Hardware open source
  - https://github.com/chipsalliance/aib-phy-hardware
  - Verilog RTL, netlists and cell models
  - Easy cell to realize: **`c3lib_and2_svt_2x`** 
    - **`assign out = in0 & in1;`**
    - Map **`c3lib_and2_svt_2x`** to “and” gate in your target process’ cell library
    - Open source hardware for RTL: no problem
  - Less easy cell to realize: **`aibcr3_dll_interpolator`** 

    - Delays the input clock to achieve 90 degree offset for data sampling
    - Compensates for PVT through dynamic phase interpolation
    - Usually implemented as custom design in a target process
    - Cannot open source a design in a foundry’s process!

Introducing AIB Generator Open Source Release

https://github.com/chipsalliance/aib-phy-generator

- Design of analog macrocells is abstracted from realization in a proprietary semiconductor process
  - Generator is an analog hardware compiler (analog synthesis)
  - Analog macrocells are defined through electrical and physical specifications
- Creates portability across processes demonstrated with Intel 22FFL, Cadence Generic Process Design Kit (PDK)
- Generator is open source even when GDSII cannot be open source

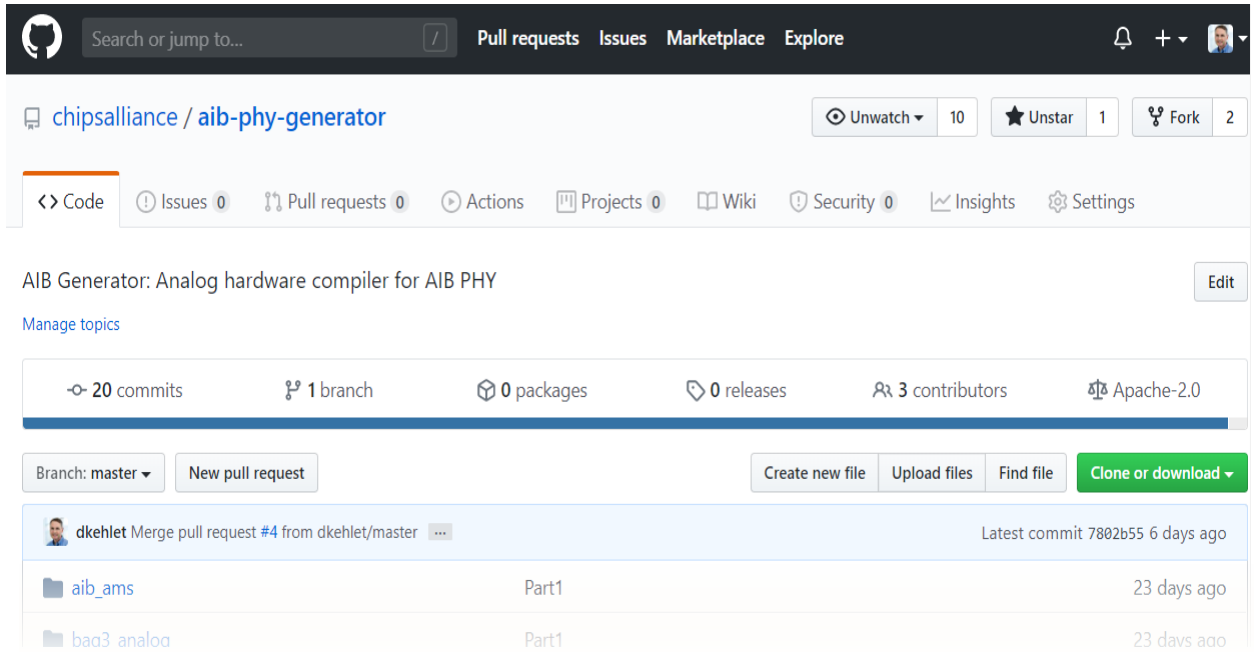


Image: Intel

AIB Generator Installation and Usage

- You need to have a silicon design ready computer
  - Red Hat Enterprise Linux 6, 64 bit
  - git tools with Internet connection
  - Cadence Virtuoso, Pegasus/PVS, Spectre (could be ported to other vendors)
- Download the Cadence Generic Process Design Kit, free to Cadence users
- **`git clone`** (download) the AIB PHY Generator

```
git clone https://github.com/chipsalliance/aib-phy-generator aib-phy-generator
```

- Tell the run scripts where to find your Cadence tools
- Design your AIB analog macrocells!

```
./run_bag.sh BAG_framework/run_scripts/dsn_cell.py
data/aib_ams/specs_dsn/dll_phase_interp.yaml
```

AIB Generator Operation

- Delay Locked Loop Phase Interpolator Design – one of the AIB analog macrocells
  - Iterates in the design space to meet the input specs
  - The Phase Interpolator script in addition can create a graph of delay vs. step code over process corners
  - .gds, .lef, .v, .lib stored in output directory

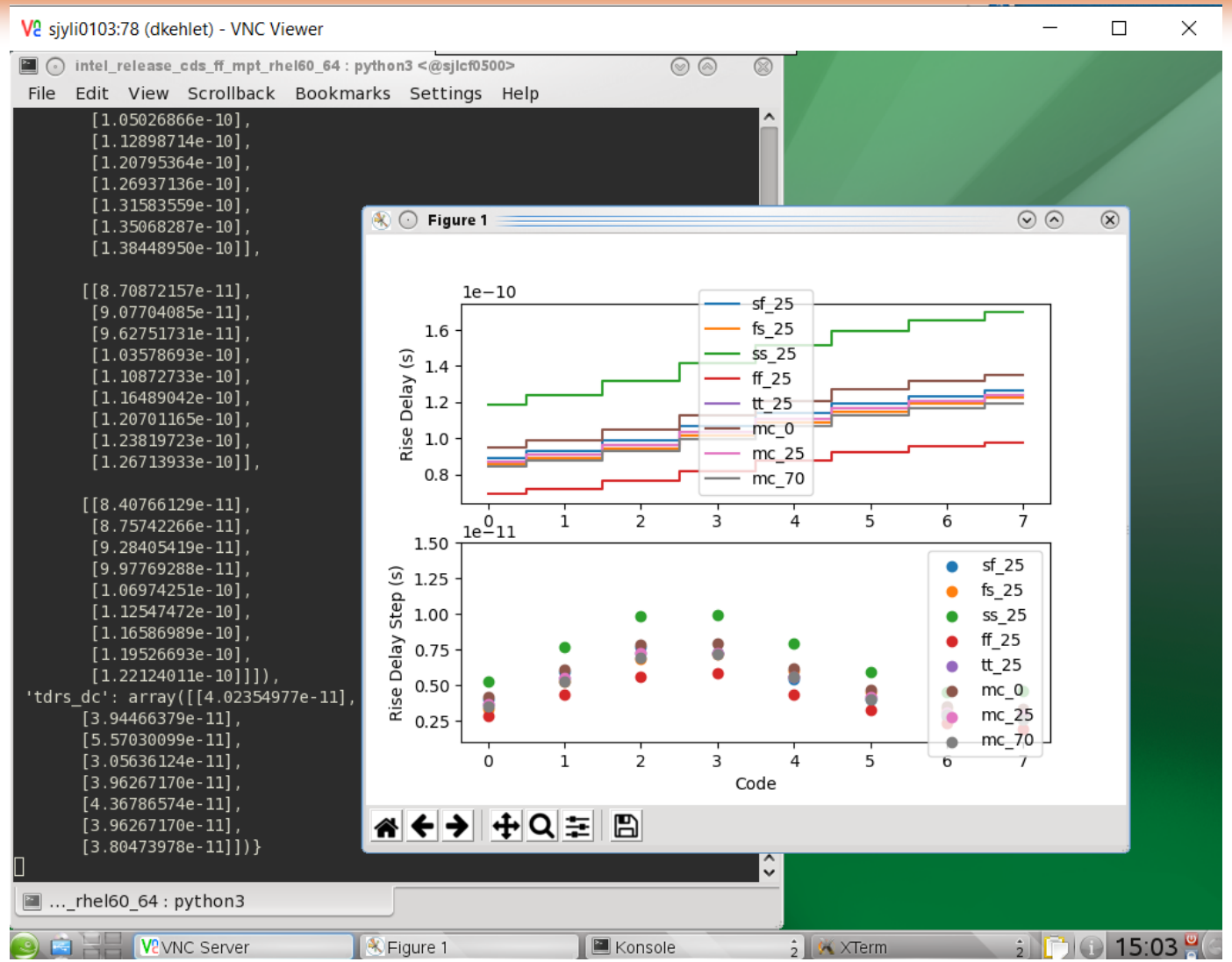


Image: Intel

AIB Generator Analog Blocks

DLL and Duty Cycle Corrector (DCC)

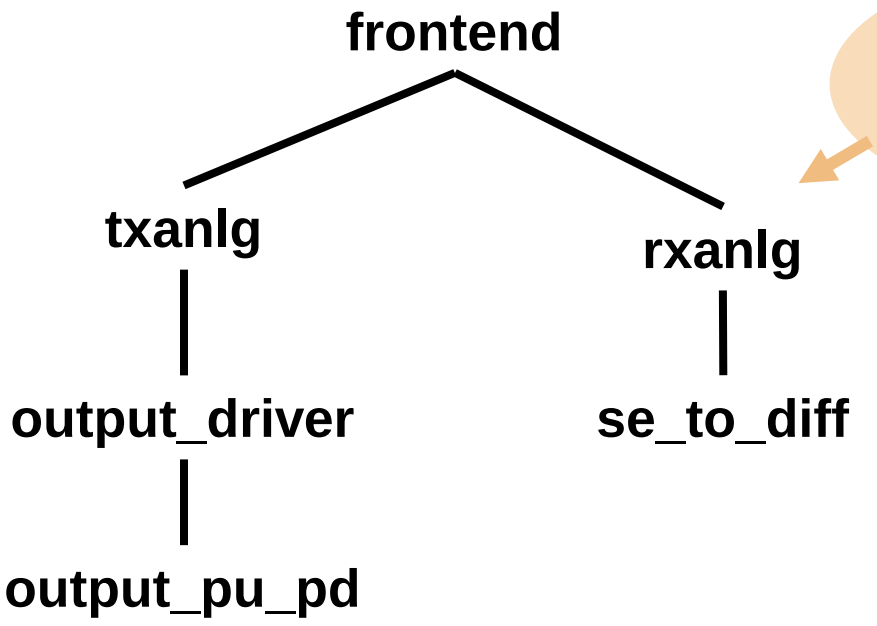
dcc/dll_phasedet

dcc/dll_interpolator

dcc/dll_delay_line

dcc_helper

Input / Output



frontend *invokes generators for its dependent blocks*

Next Steps: **AIB 2.0**

- CHIPS Alliance AIB 2.0 Specification
  → 2x the IOs, ½ the bump array size, 6.4x the edge bandwidth density!
- CHIPS Alliance AIB 2.0 Draft Specification available at
  - https://github.com/chipsalliance/AIB-specification
- AIB 2.0 tape out intended for early 2021

Feature	AIB 1.0	AIB 2.0
Bandwidth/wire (Gbps)	2	Up to 6.4
Common case* IOs/channel (TX/RX)	40 (20/20)	80 (40/40)
Bump density (um)	55	55/45/36
Bandwidth/mm shoreline (Gbps/mm)	256 (40 IOs/channel)	1638 (80 IOs/channel)
IO Voltage Output Swing (V)	0.90	0.90/0.40
Energy/bit (pJ/bit)	0.85	0.50
Backward Compatibility	n/a	1.0

* Common case is a high volume device from Intel. AIB specification and technology supports 20/20 to 80/80 IOs/channel.

Conclusion

- The chiplet era is here! And Generators are the catalyst!
- Powerful designer-centric generator methodology enables rapid parameterization and process migration
- Open source AIB Generator enables rapid generation of sign-off ready AIB custom A/MS blocks across many PDKs
  - https://github.com/chipsalliance/aib-phy-hardware RTL
  - https://github.com/chipsalliance/aib-phy-generator Analog
- Technologies such as optical networking are pushing bandwidth density – AIB 2.0 is coming

Questions