Design Flow Parameter Optimization with Multi-Phase Positive Nondeterministic Tuning

Matthew M. Ziegler, Lakshmi N. Reddy, Robert L. Franch

IBM T. J. Watson Research Center
Yorktown Heights, NY, 10598

contact: zieglerm@us.ibm.com

IBM®

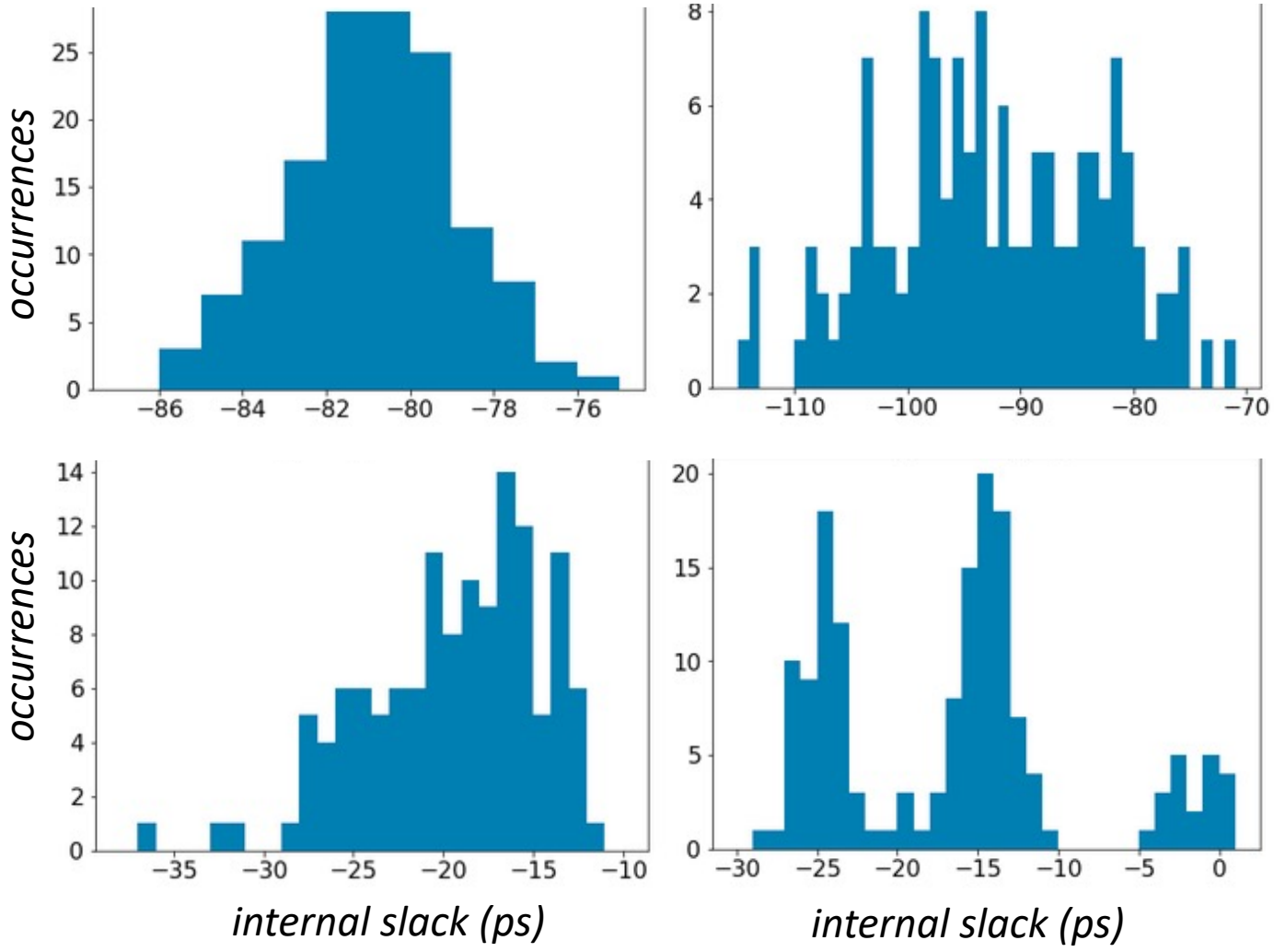
Design Flow Tuning in a Nondeterministic Tool Setting

- Aggressive PPA (performance, power, area) targets & tight product schedules often push CAD tools to the limits
  - Skilled designers are tasked with achieving better PPA than the tools natively deliver
- *Design Flow Tuning* is a key for optimizing Synthesis and P&R flows
  - But, CAD tools can have 100-1000's of parameters -> highly complex design space…
  - Automated Design Flow Tuning is emerging as a productive tuning solution
- *CAD tool nondeterminism* can further complicate design closure & design flow tuning
  - i.e., some CAD tools return different results, given identical inputs
  - Typically, CAD tool nondeterminism is an unwanted behavior
- *We propose "Positive Nondeterministic Tuning" (PNT) to harness nondeterminism for outperforming prior design flow tuning approaches*

Challenges of Nondeterminism

- *Nondeterminism*: identical input, different results, sometimes significantly different
  - Similar to *Chaos* [1]: slightly perturbed inputs, significantly different results
- What is the source of a "good" results given nondeterminism or chaos?
  - *Skill* – a skillful modification of tool parameters
  - *Luck* – nondeterminism led to a better results

Examples of Nondeterminism: internal slack (ps) for 4 designs 150 Physical Synthesis runs



[1] K. Jeong and A. B. Kahng, "Methodology from chaos in IC implementation," ISQED'10.

Design Flow Tuning Background

- CAD tool & design flow tuning is gaining interest in both academia & industry
  - Publications appearing in numerous domains, e.g.,
    - logic synthesis, physical design (placement, CTS, …), FPGA flows, ASIC flows, etc.
- Two main approaches: *Online* & *Offline* tuning [2]
- *Online* tuning is our target in this work
  - PNT concepts can be extended to Offline or hybrid approaches

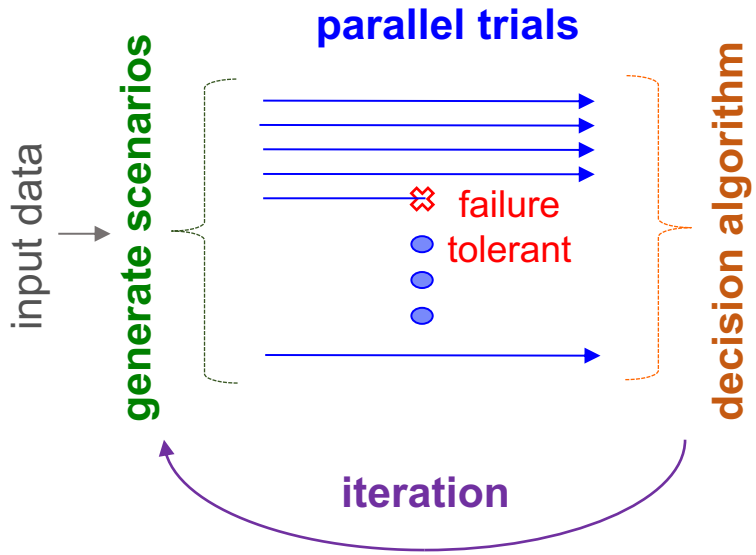
Approach	Description, Examples, Challenges
Online	*Learning without a prior dataset, as data becomes available, typically iterative*
	Examples: Bayesian Optimization, Active Learning, Reinforcement Learning
	Keys Challenges: compute cost of parallel trials, iterative runtime latency
Offline	*Batch Learning, i.e., training from an existing dataset, typically followed by inference*
	Examples: Neural Networks, Recommender Systems, Transfer Learning
	Keys Challenges: dataset curation, training runtime

[2] M. Ziegler, J. Kwon, H.-Y. Liu, and L. Carloni. "Online and Offline Machine Learning for Industrial Design Flow Tuning," ICCAD'21.

Online Design Flow Tuning Considerations

- **Logic & Physical Synthesis design flow is our target application**
  - *Similar implications for any application with similar compute characteristics*

Characteristic	Implication
Large Number of Parameters	→ Iterative Refinement
Long Runtimes	→ Parallel Trials
Overall Tuning Latency Limitation	→ Few Iterations
Distributed Compute Environment	→ Tolerant of Trial Failure



generic design flow tuning framework

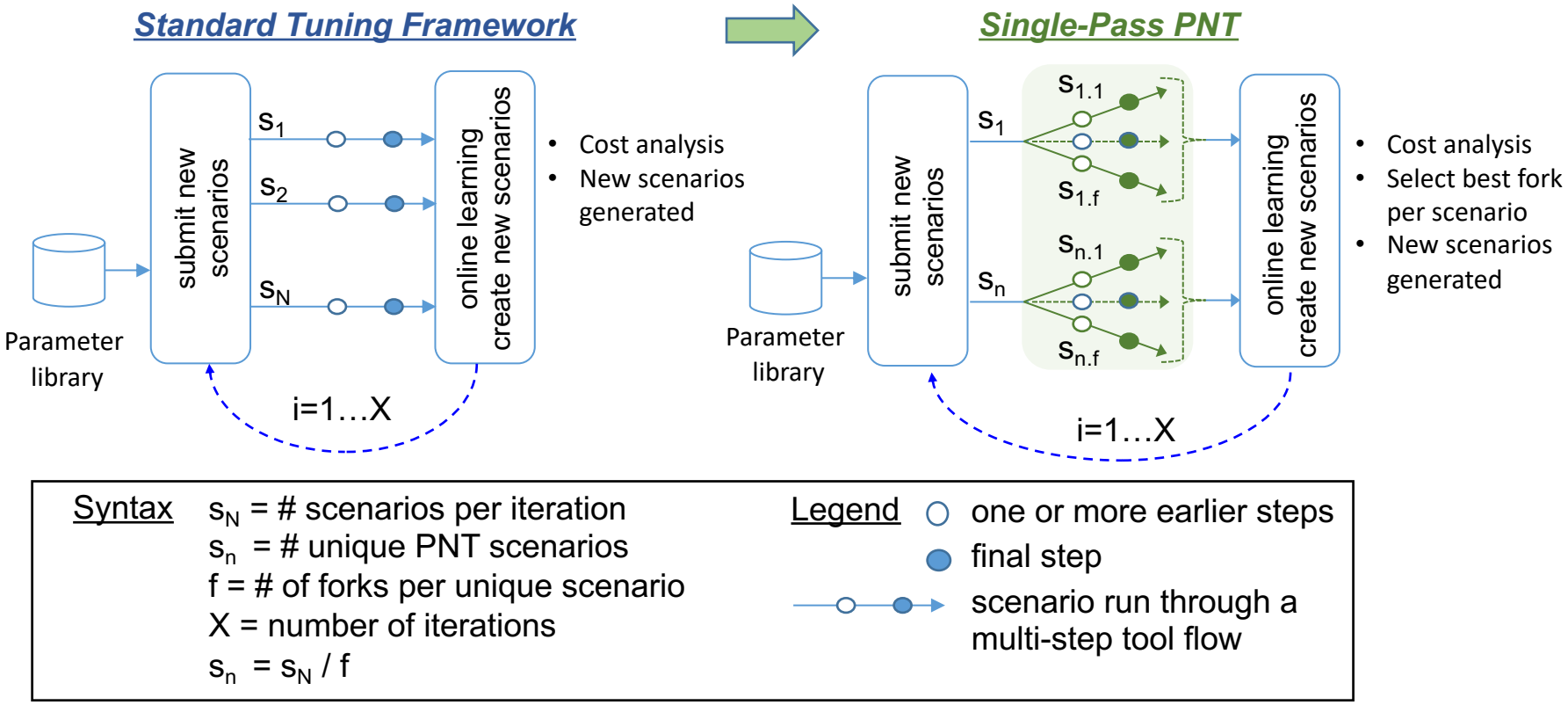
- **CAD Tool Application Specific Considerations**
  - Many parameters are categorical typed
  - Well intentioned default parameter settings
  - Parameters may be intended to affect specific metrics

Positive Nondeterministic Tuning (PNT) Concepts

- PNT introduces 2 main approaches that can enhance a "generic design flow tuning compute framework"
- *Forking*
  - Multiple identical trials (same parameter settings) are run in parallel
  - Challenge: limited compute resources
    - Tradeoffs between parameter exploration (unique trials) vs. exploitation of nondeterminism (identical trials)
  - Chaotic systems can inject minor perturbations into forked trials
- *Multi-Phase Tuning*
  - Save promising tuning trials (or intermediate results) as seeds and perform additional tuning passes using seeds
  - Challenge: reduce latency of additional tuning passes
- *Exploit nondeterminism (forking) to elicit positive outliers, use them as seeds to elicit further positive outliers (multi-phase tuning)*

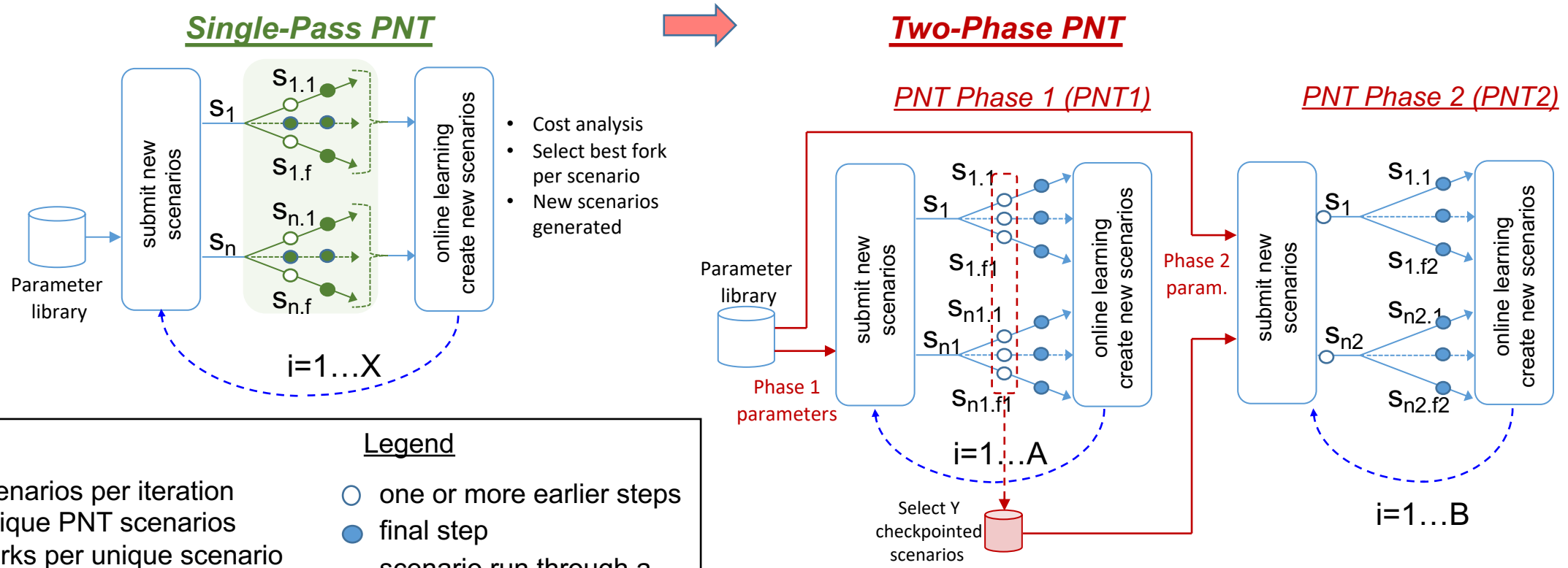
PNT Forking

- A standard parallel & iterative tuning framework an be modified to include forking without modifying the underlying ML algorithm
  - Trial submission are intercepted, forked, and a single (best) trial is returned
  - Forking can be integrated into the underlying algorithm, with more complexity



Multi-Phase Tuning

- Two or more running phases allows saving information from top trials in earlier phases as seeds
  - Tuning parameters can be modified between phases to focus on specific flow steps
  - Multiple seed mechanisms can tradeoff latency and flexibility for future design changes



Syntax

s_N = # scenarios per iteration
s_n = # unique PNT scenarios
f = # of forks per unique scenario
X = number of iterations
$s_n = s_N / f$

Legend

one or more earlier steps
final step
scenario run through a multi-step tool flow

Proposed Multi-Phase Seed Mechanisms

- *Checkpoint (CP)* – saves complete placement & sizing (database) at an intermediate step, e.g., after CTS (clock tree synthesis)
- *Preplace (PP)* – saves placement & sizing of CTS network, registers & clock buffering
- *Attractions (ATT)* – saves placement attractions for registers only, attractions guide the tool, but the tool some ability to move registers

Mode	**Checkpoint (CP)**	**Preplace (PP)**	**Attractions (ATT)**
Flexibility	Least	Medium	Most
Allows Design changes	No	Comb. logic	Yes
Restart flow from first step	No	Yes	Yes
Run time savings	Most	Medium	Medium

Experimental Case Study

- 5 macros (designs/blocks) selected from an industrial processor remap
  - Relative macro footprint reduced
  - Lowest Vt withheld for power reduction
  - All selected macros timing challenged

Macro Name	Logic function	Logic gates	Runtime (hrs)
IDEC	Instruction decode	191K	22
INTR	Interrupt handler	27K	4.3
DAQ	Data/addr queue	36K	10.3
FP	Floating point pipe	63K	11
DIV	Divider algorithm	17K	3

- Tuning metrics weighted in terms of importance & design closure targets

Metric	Tuning Weight	Closure Target
R2R : Worst Negative Internal Slack (ps)	2	>=0
WNS : Worst Negative Slack (ps)	1	NA
TNS : Total Negative Slack (ps)	1	NA
Congestion (a.u.)	3	<=90
Total Power (mW)	4	minimize

- Ranking criteria is “normalized weighted sum” (min-max scaling)

$$Cost = \sum_{i=1}^{m} W_i \cdot Norm(M_i)$$

Comparisons Flows

Flow	Description
Default x 150	running the default flow parameter settings 150 times
Stable	running the tool in a deterministic mode – incurs a higher runtime
STS	SynTunSys – IBM's in-house design flow tuner*
PNT2-CP	2-Phase PNT using the checkpoint seed, fork count set to 3
PNT2-PP	2-Phase PNT using the preplace seed, fork count set to 3
PNT2-ATT	2-Phase PNT using the attractions seed, fork count set to 3

Comments:

- PNT flows modify the STS ML algorithm, which is an in-house Bayesian inspired algorithm described in [3]
- All flows run a similar number of trials, with the exception of Stable
- * SynTunSys (STS) [3] has been employed to optimize for all IBM Server processors for over 10 years

[3] M. Ziegler, H.-Y. Liu, L. Carloni, "Scalable auto- tuning of synthesis parameters for optimizing high-performance processors," ISPLED'16

Overall QoR Comparison

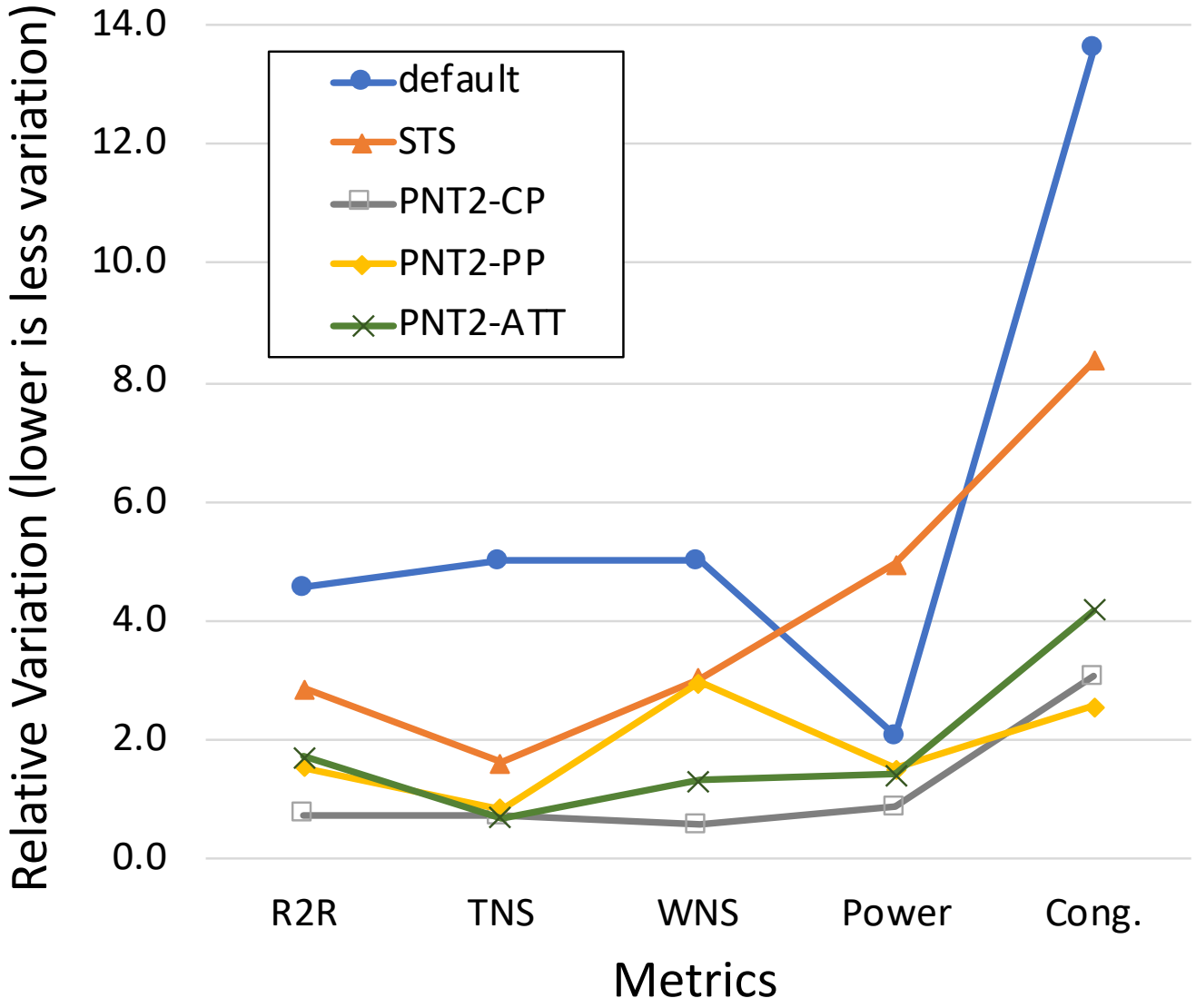
- Overall, all 3 PNT flows outperform prior flows
- PNT2-ATT performed the best considering weighted cost of metrics of interest
- PNT2-PP had the best congestion result
- PNT2-CP has an inherent runtime advantage (>20% faster than other PNT2 flows)

Design	R2R : Worst Internal Slack (ps)						Congestion (a.u.)						Total Power (mW)						Utilization (%)					
	stable	default x 150	STS	PNT2-CP	PNT2-PP	PNT2-ATT	stable	default x 150	STS	PNT2-CP	PNT2-PP	PNT2-ATT	stable	default x 150	STS	PNT2-CP	PNT2-PP	PNT2-ATT	stable	default x 150	STS	PNT2-CP	PNT2-PP	PNT2-ATT
IDEC	-82	-71	-18	-18	-19	-12	101	105	101	94	94	94	122	117	107	106	109	106	62	60	52	50	51	50
INTR	-16	-11	-12	-3	-3	0	91	91	89	90	89	89	19	19	18	18	18	18	81	83	77	76	76	76
DAQ	-86	-57	-8	0	-6	2	91	92	92	92	89	91	22	22	21	21	21	20	82	83	71	70	70	70
FP	-53	-39	-11	-5	-6	-5	91	91	88	88	89	88	38	39	36	36	36	36	76	77	65	65	65	65
DIV	-97	-78	-16	-8	-5	-2	91	92	88	89	89	91	8	8	7	7	7	7	71	72	64	59	60	58
SUM	**-333**	**-257**	**-64**	**-34**	**-39**	**-16**	**465**	**471**	**459**	**452**	**450**	**454**	**209**	**206**	**189**	**188**	**190**	**187**	**372**	**375**	**329**	**320**	**322**	**319**

Variability Comparison

- The best scenario from PNT2 flows are also less susceptible to run-to-run variation
  - Seeds introduce some amount of repeatability
- Additional results in the paper:
  - Runtime/Latency comparison of the proposed flows
  - PNT3 – adding a 3rd phase for particularly difficult designs
  - ePNT – early PNT, latency reducing technique
  - Variable forking – customization for count per design

50 identical runs of the best scenario from each flow



Summary

- Design flow tuning complexity & nondeterministic CAD tools create challenges in achieving aggressive PPA targets in modern VLSI design
- We have presented *"Positive Nondeterministic Tuning" (PNT)*, an approach that leverages nondeterminism within a design flow tuning framework
- Experimental results show PNT outperforms an existing industrial design flow tuning approach
- *So, if you can't avoid nondeterminism, leverage it for a better design*
  - *Maybe even add more nondeterminism to elicit even better outliers…*