Construction of Realistic Gate Sizing Benchmarks With Known Optimal Solutions

Andrew B. Kahng, Seokhyeong Kang

VLSI CAD LABORATORY, UC San Diego

International Symposium on Physical Design

March 27th, 2012

Outline

- **Background and Motivation**
- Benchmark Generation
- Experimental Framework and Results
- Conclusions and Ongoing Work

Gate Sizing in VLSI Design

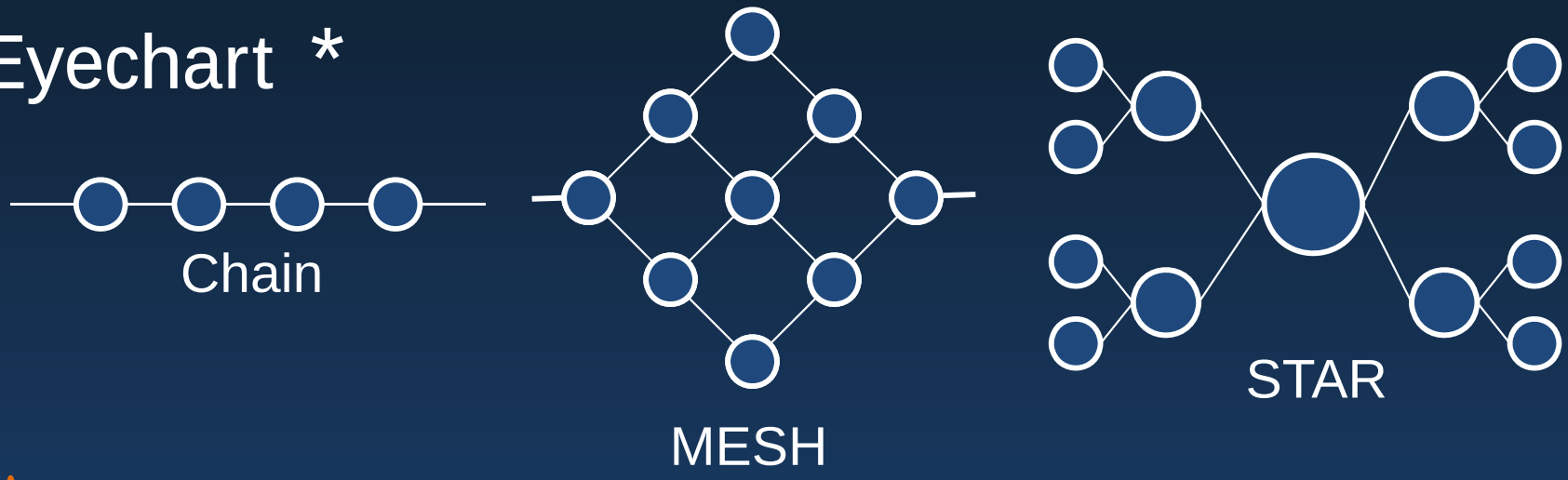
- Gate sizing
  - Essential for power, delay and area optimization
  - Tunable parameters: gate-width, gate-length and threshold voltage
  - Sizing problem seen in all phases of RTL-to-GDS flow
- Common heuristics/algorithms
  - LP, Lagrangian relaxation, convex optimization, DP, sensitivity-based gradient descent, …

1. Which heuristic is better?
2. How suboptimal a given sizing solution is?

→ systematic and quantitative comparison is required

Suboptimality of Sizing Heuristics

- Eyechart *

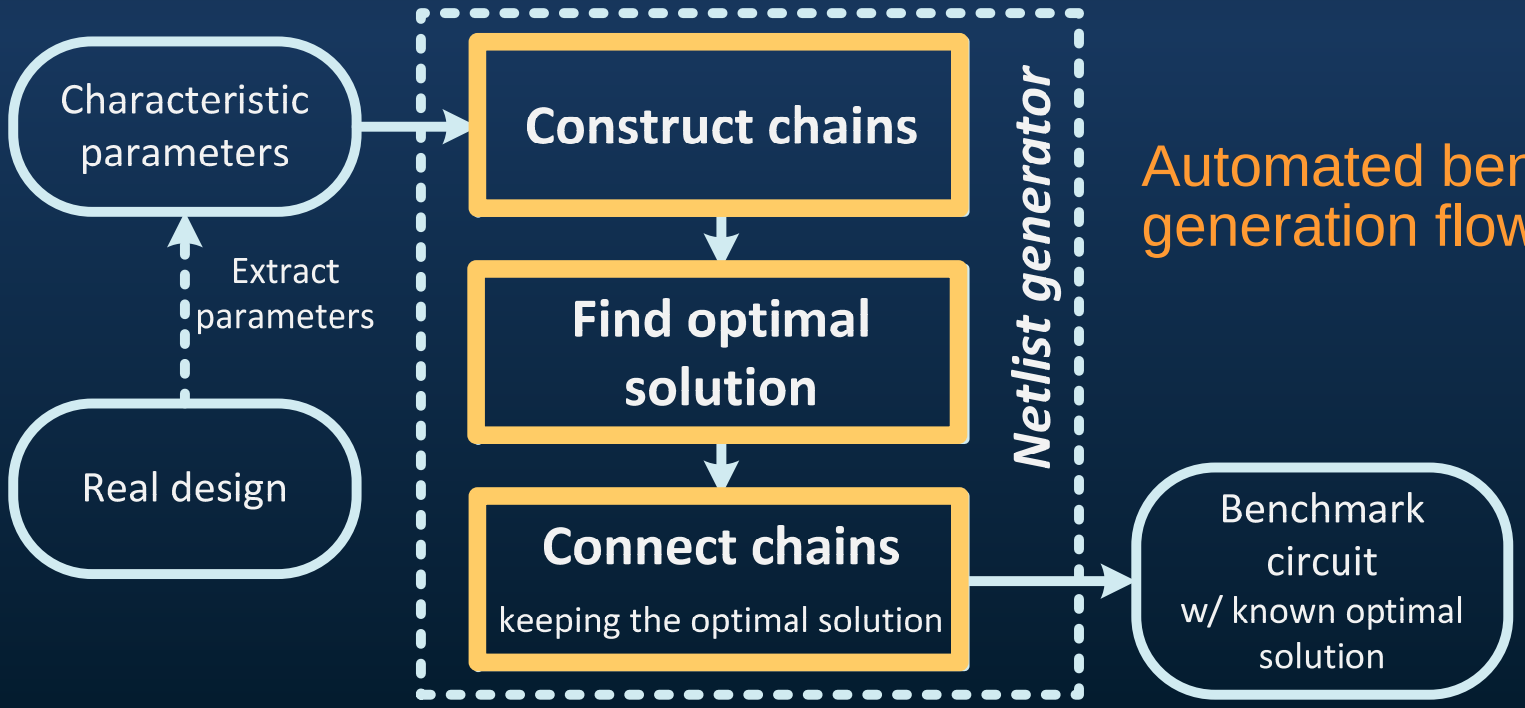


  - Built from three basic topologies, optimally sized with DP – allow suboptimalities to be evaluated
  - Non-realistic: Eyechart circuits have different topology from real design – large depth (650 stages) and small Rent parameter (0.17)
- More realistic benchmarks are required along w/ automated generation flow

*Gupta et al., “Eyecharts: Constructive Benchmarking of Gate Sizing Heuristics”, DAC 2010.

Our Work: Realistic Benchmark Generation w/ Known Optimal Solution

1. Propose benchmark circuits with known optimal solutions
2. The benchmarks resemble real designs
   - Gate count, path depth, Rent parameter and net degree
3. Assess suboptimality of standard gate sizing approaches



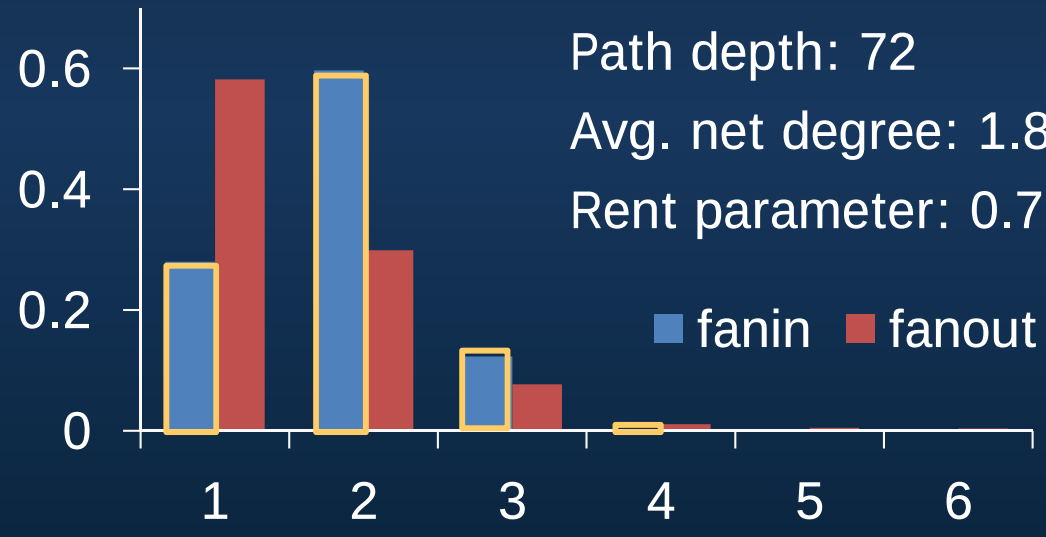
Automated benchmark generation flow

Outline

- Background and Motivation
- **Benchmark Considerations and Generation**
- Experimental Framework and Results
- Conclusions and Ongoing Work

Benchmark Considerations

- Realism vs. Tractability to Analysis – opposing goals
- To construct realistic benchmark: use design characteristic parameters
  - # primary ports, path depth, fanin/fanout distribution



Path depth: 72

Avg. net degree: 1.84

Rent parameter: 0.72

design:
JPEG Encoder

Fanin distirbution
25%: 1-input
60%: 2-input
15%: >3-input

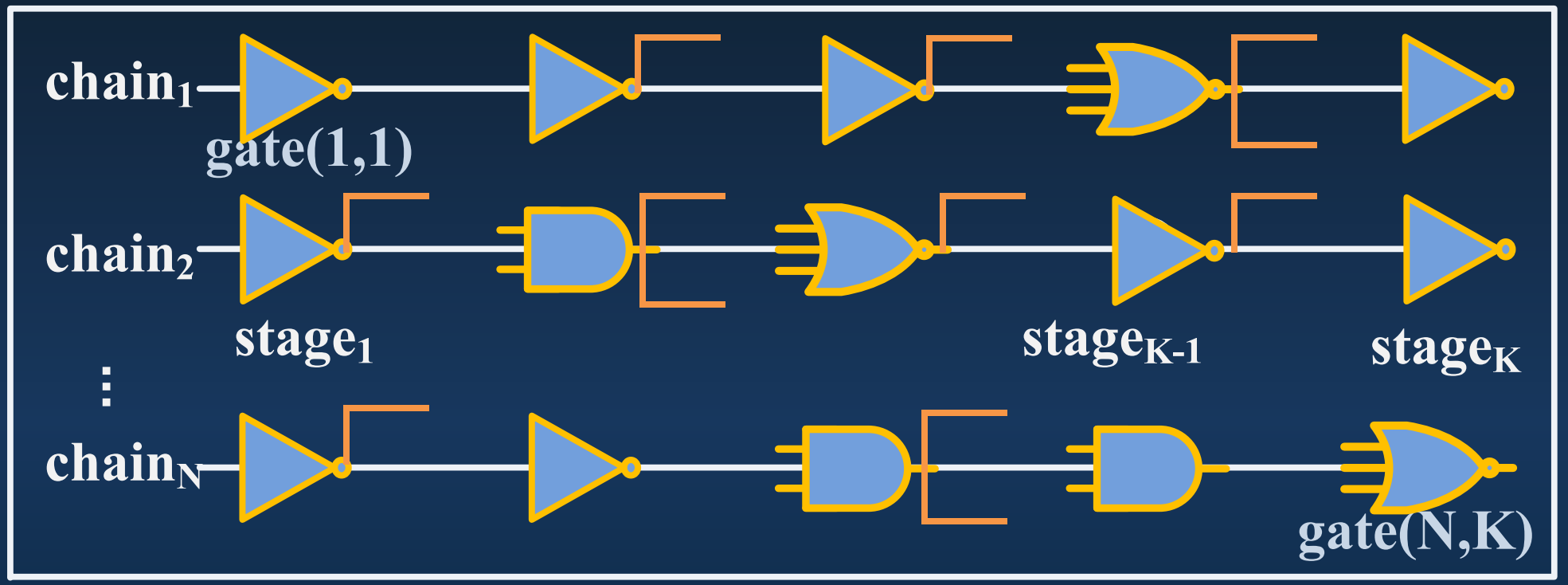
- To enable known optimal solutions
  - Library simplification as in Gupta et al. 2010: slew-independent library

Benchmark Generation

- Input parameters
  1. timing budget ***T***
  2. depth of data path ***K***
  3. number of primary ports ***N***
  4. fanin, fanout distribution ***fid(i), fod(j)***
- Constraints
  - ***T*** should be larger than min. delay of ***K***-stage chain
  - $\sum_{i=1}^{I} i \cdot fid(i) = \sum_{o=1}^{O} o \cdot fod(o)$
- Generation flow
  1. construct ***N*** chains with depth ***K***
  2. attach connection cells (***C***)
  3. connect chains

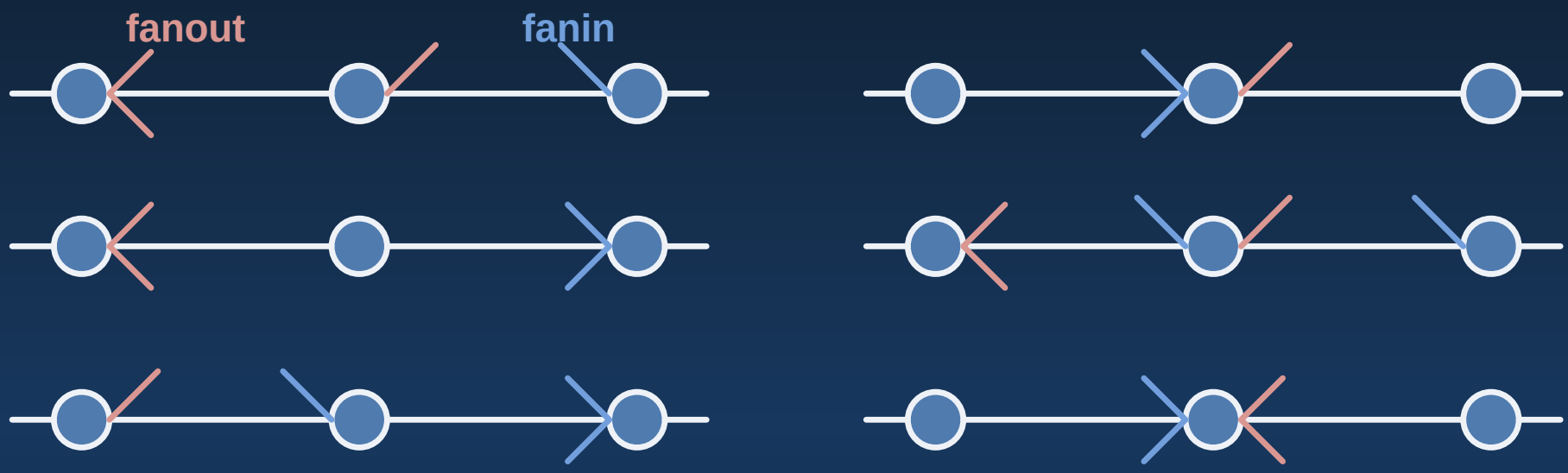
  ➔ netlist with ***N*K + C*** cells

Benchmark Generation: Construct Chains



- Construct *N* chains each with depth *k* (*N*k* cells)
- Assign gate instance according to *fid(i)*
- Assign # fanouts to output ports according to *fod(o)*
- Assignment strategy: arranged and random

Benchmark Generation: Construct Chains



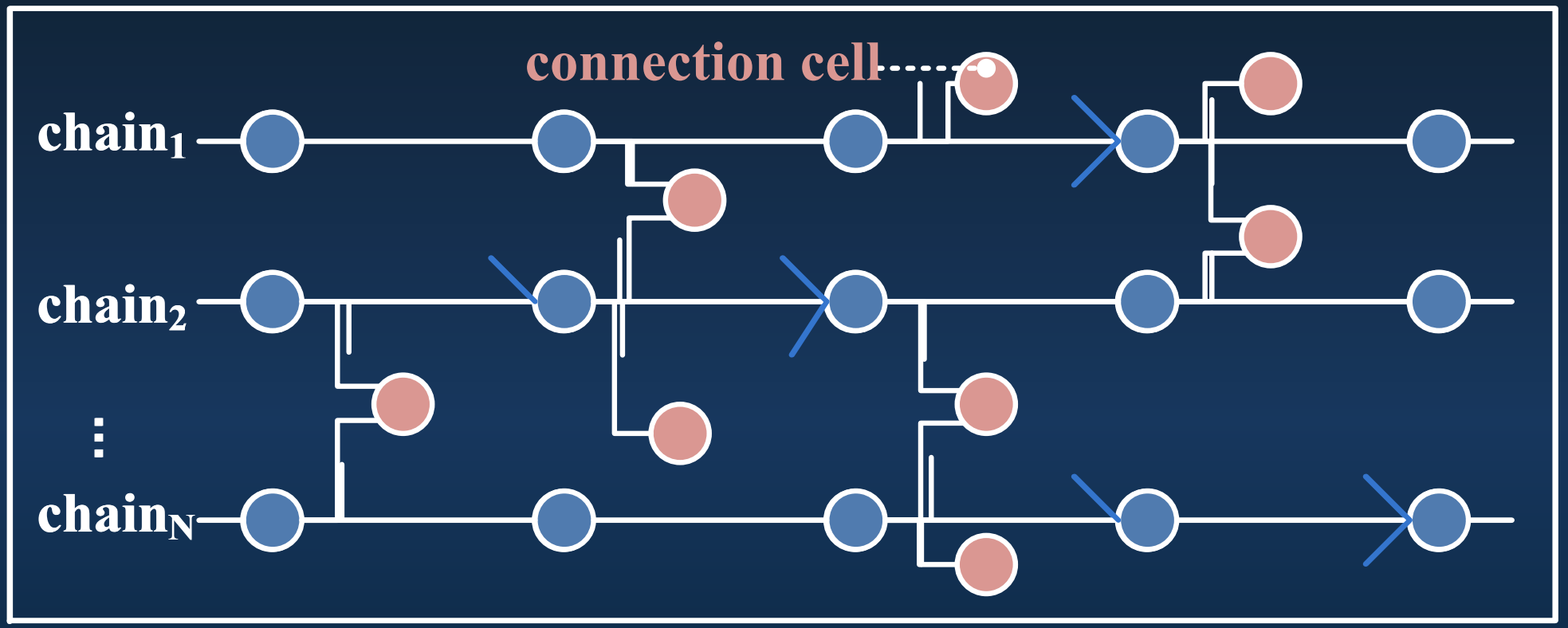
Arranged assignment

Random assignment

1. Construct *N* chains each with depth *k* (*N*k* cells)
2. Assign gate instance according to *fid(i)*
3. Assign # fanouts to output ports according to *fod(o)*

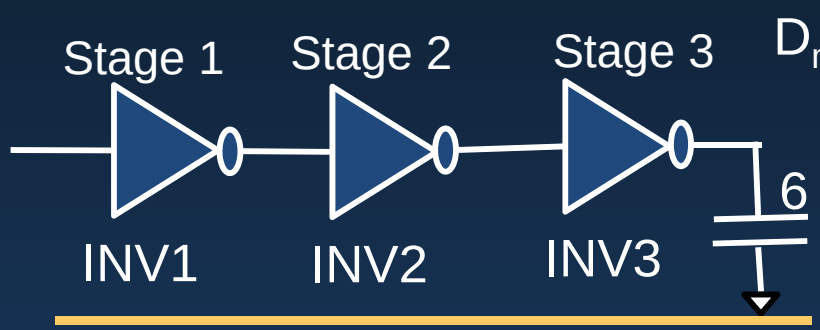
Assignment strategy: arranged and random

Benchmark Generation: Find Optimal Solution with DP



1. Attach connection cells to all open fanouts
   - to connect chains keeping optimal solution
2. Perform dynamic programming with timing budget *T*
   - optimal solution is achievable w/ slew-independent lib.

Benchmark Generation: Solving a Chain Optimally (Example)



$D_{max} = 8$

size	input cap	leakage power	delay	
			load 3	load 6
Size 1	3	5	3	4
Size 2	6	10	1	2

Stage 1

Budget	Power	Size	
1	10	2	Load = 3
2	10	2	
3	5	1	
4	5	1	
5	5	1	
6	5	1	
7	5	1	
8	5	1	
2	10	2	Load = 6
3	10	2	
4	5	1	
5	5	1	
6	5	1	
7	5	1	
8	5	1	

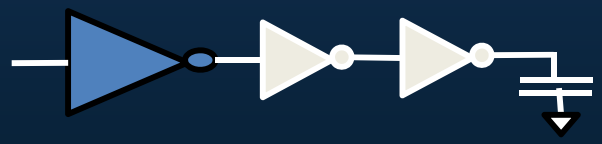
Stage 2

Budget	Power	Size	
3	20	2	Load = 3
4	15	1	
5	15	2	
6	10	1	
7	10	1	
8	10	1	
4	20	2	Load = 6
5	15	1	
6	15	2	
7	10	1	
8	10	1	

Stage 3

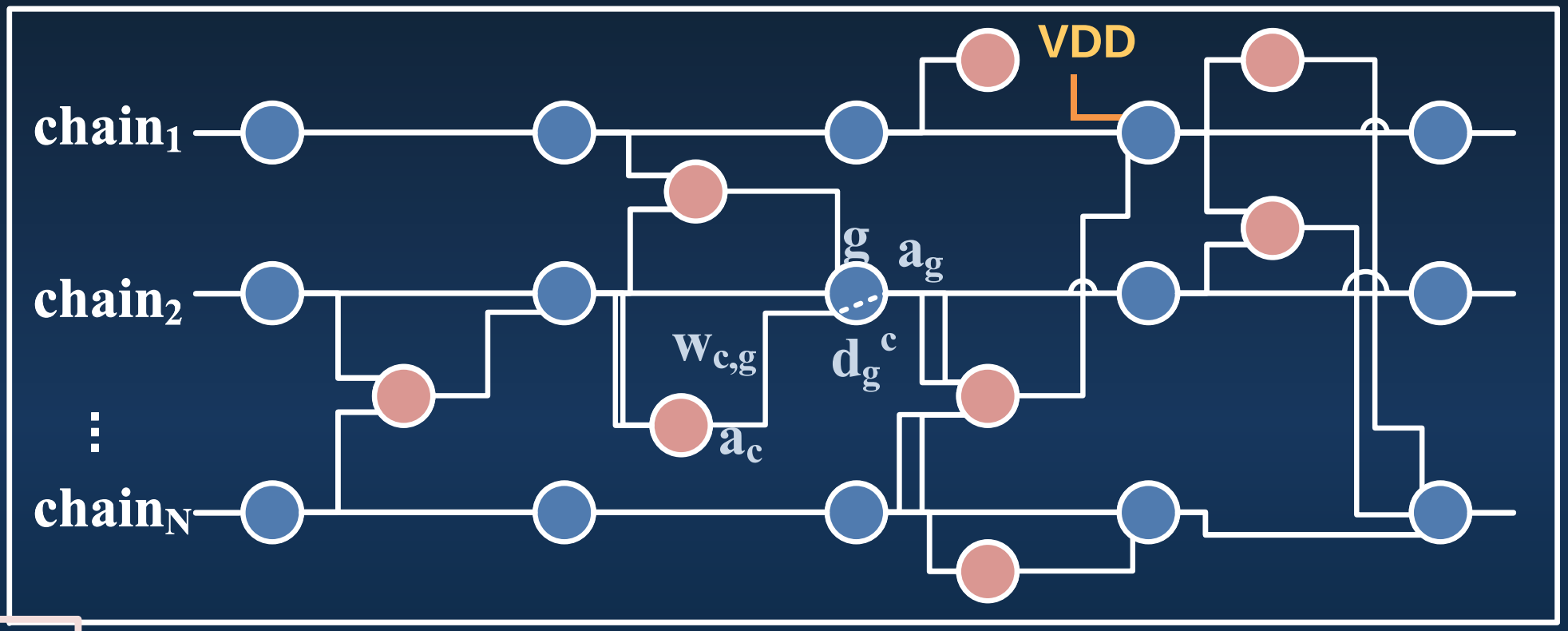
Budget	Power	Size
8	20	1
8	25	2

OPTIMIZED CHAIN



size 2 size 1 size 1

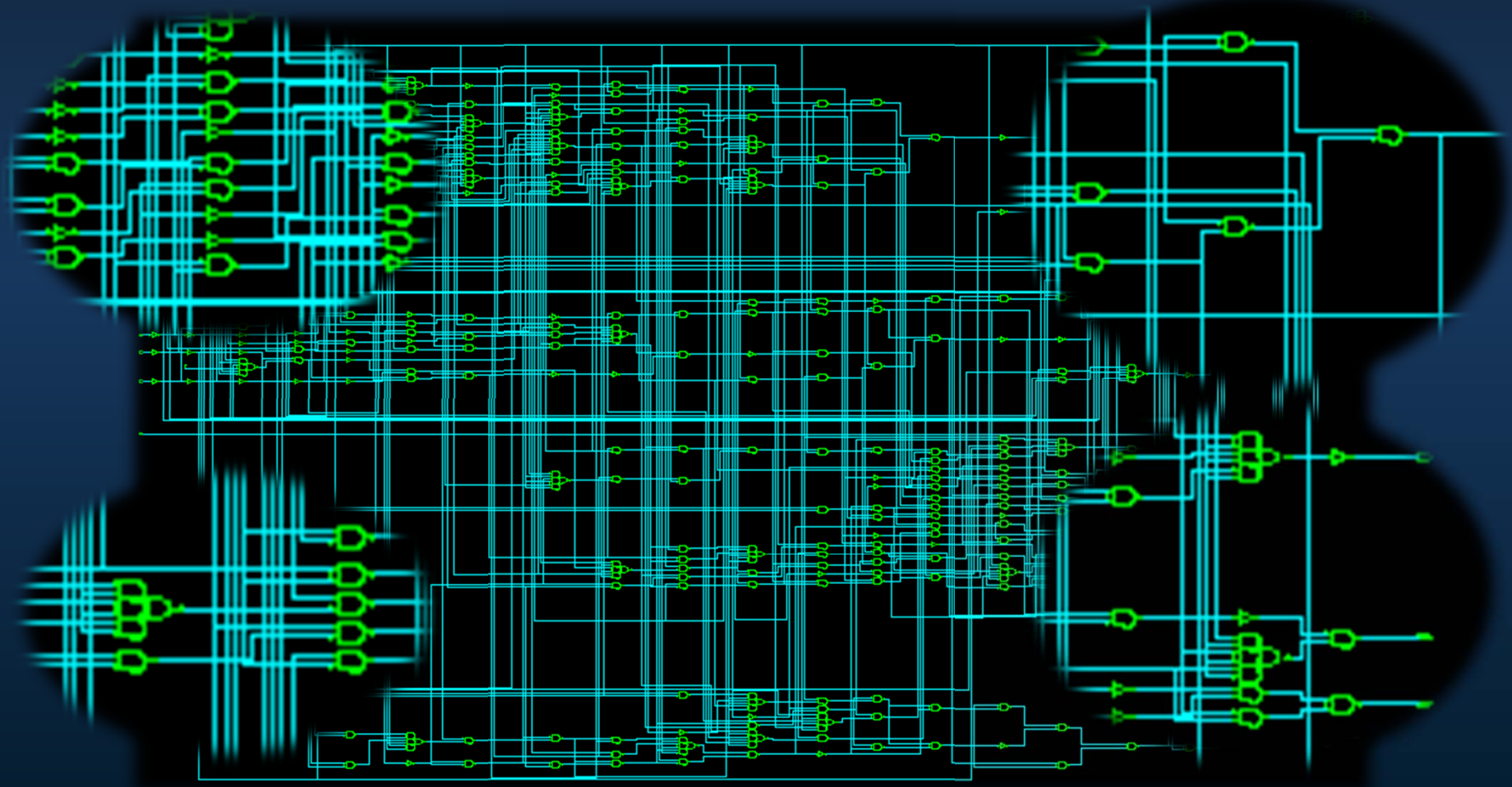
Benchmark Generation: Connect Chains



1. Run STA and find arrival time for each gate
2. Connect each connection cell to open fanin port
   - connect only if timing constraints are satisfied
   - connection cells do not change the optimal chain solution
3. Tie unconnected ports to logic high or low

Benchmark Generation: Generated Netlist

- Generated output:
  - benchmark circuit of $N*K + C$ cells w/ optimal solution



Schematic of generated netlist (N=10, K=20)

Chains are connected to each other (→ various topologies

Outline

- Background and Motivation
- Benchmark Generation
- **Experimental Framework and Results**
- Conclusions and Ongoing Work

Experimental Setup

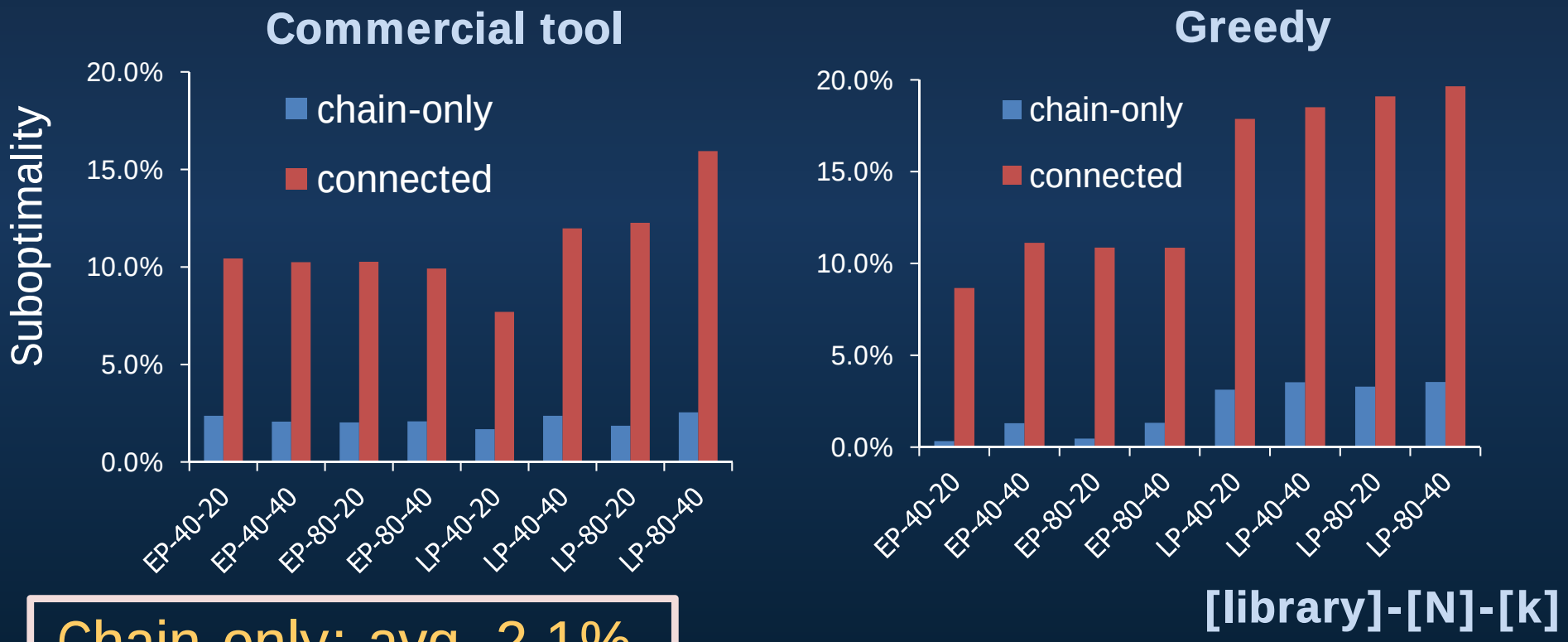
- Delay and Power model (library)
  - ***LP:*** linear increase in power – gate sizing context
  - ***EP:*** exponential increase in power – Vt or gate-length
- Heuristics compared
  - Two commercial tools (BlazeMO, Cadence Encounter)
  - UCLA sizing tool
  - UCSD sensitivity-based leakage optimizer
- Realistic benchmarks: six open-source designs
- Suboptimality calculation

$$Suboptimality = \frac{power_{heuristic} - power_{opt}}{power_{opt}}$$

Generated Benchmark - Complexity

- Complexity (suboptimality) of generated benchmark

Chain-only vs. connected-chain topologies



Chain-only: avg. 2.1%

Connected-chain: avg. 12.8%

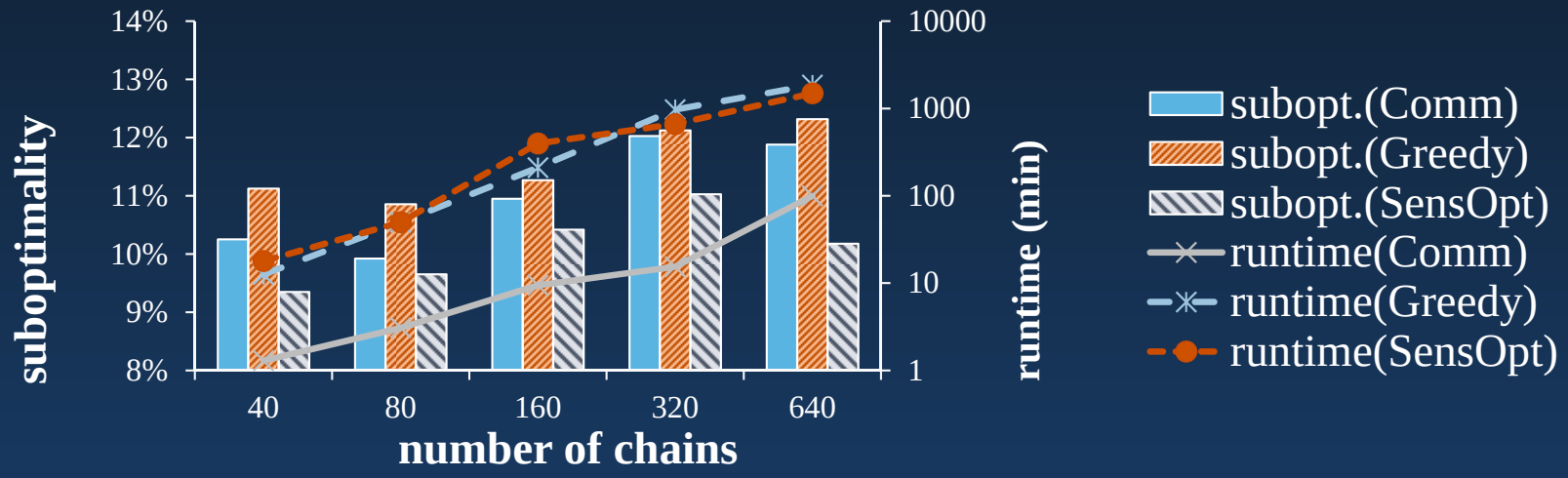
Generated Benchmark - Connectivity

- Problem complexity and circuit connectivity
  1. Arranged assignment: improve connectivity
     (larger fanin – later stage, larger fanout – earlier stage)
  2. Random assignment: improve diversity of topology

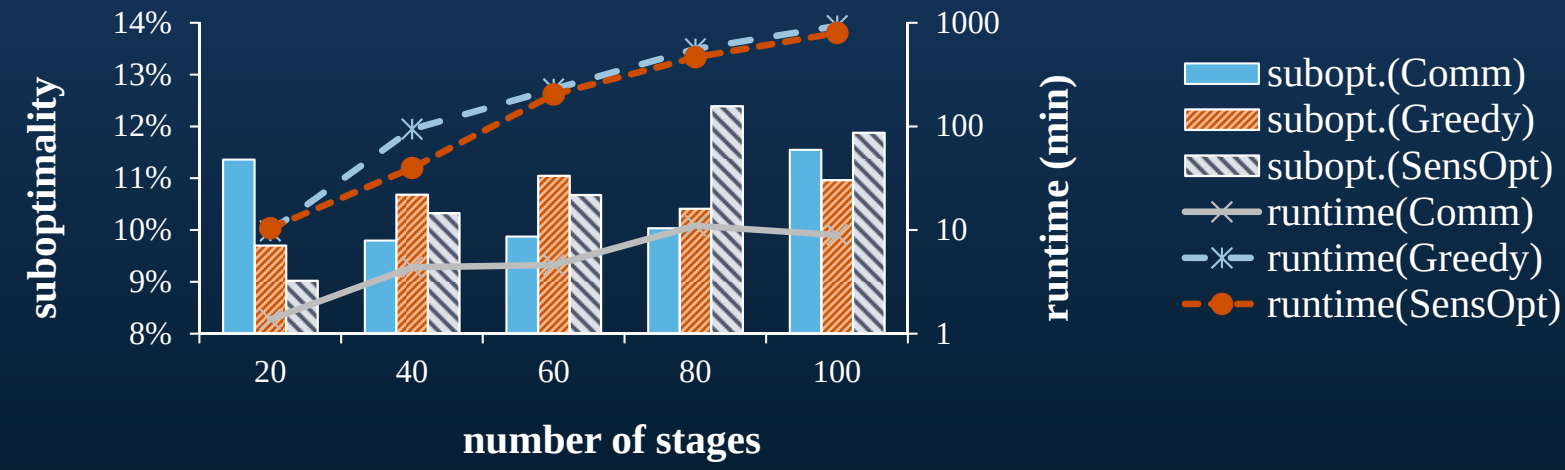
arranged	random	unconnected	Subopt.
100%	0%	0.00%	2.60%
75%	25%	0.00%	6.80%
50%	50%	0.25%	10.30%
25%	75%	0.75%	11.20%
0%	100%	17.00%	7.70%

Suboptimality w.r.t. Parameters

- For different number of chains



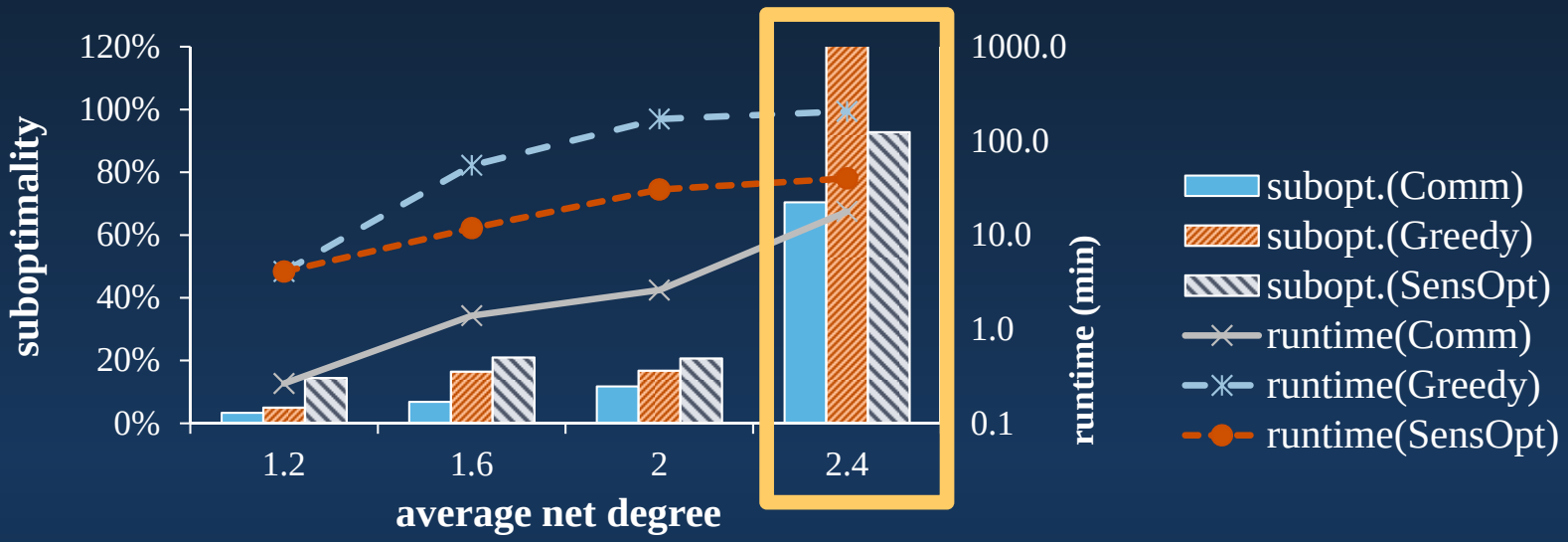
- For different number of stages



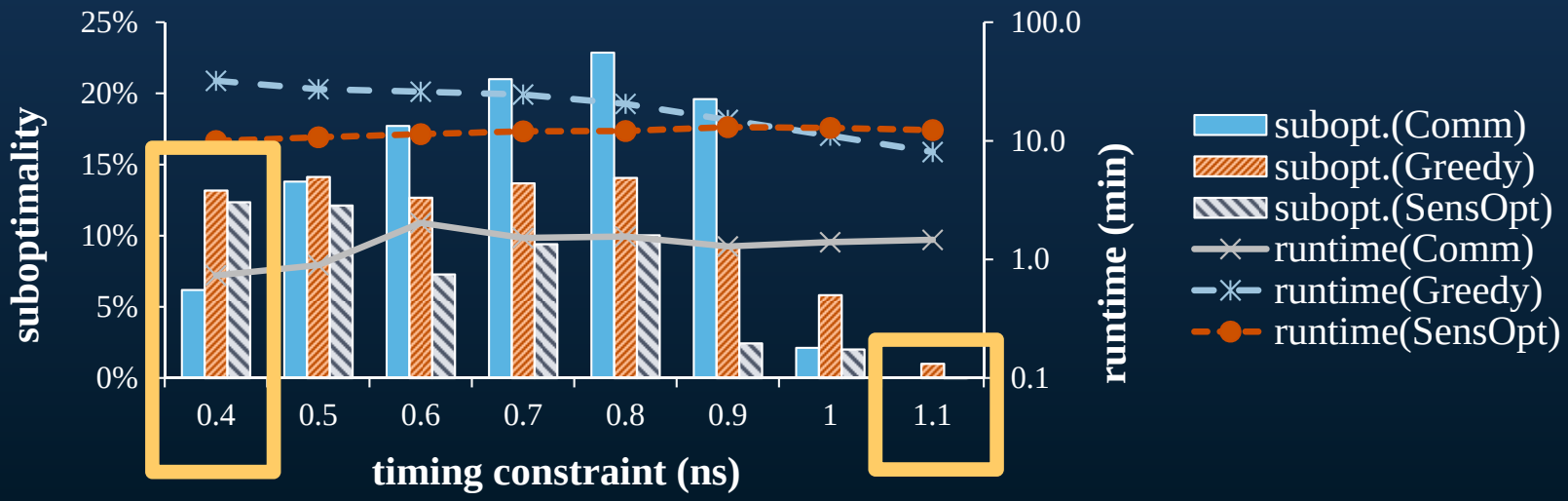
Total # paths increase significantly w.r.t. N and K

Suboptimality w.r.t. Parameters (2)

- For different average net degrees



- For different delay constraints



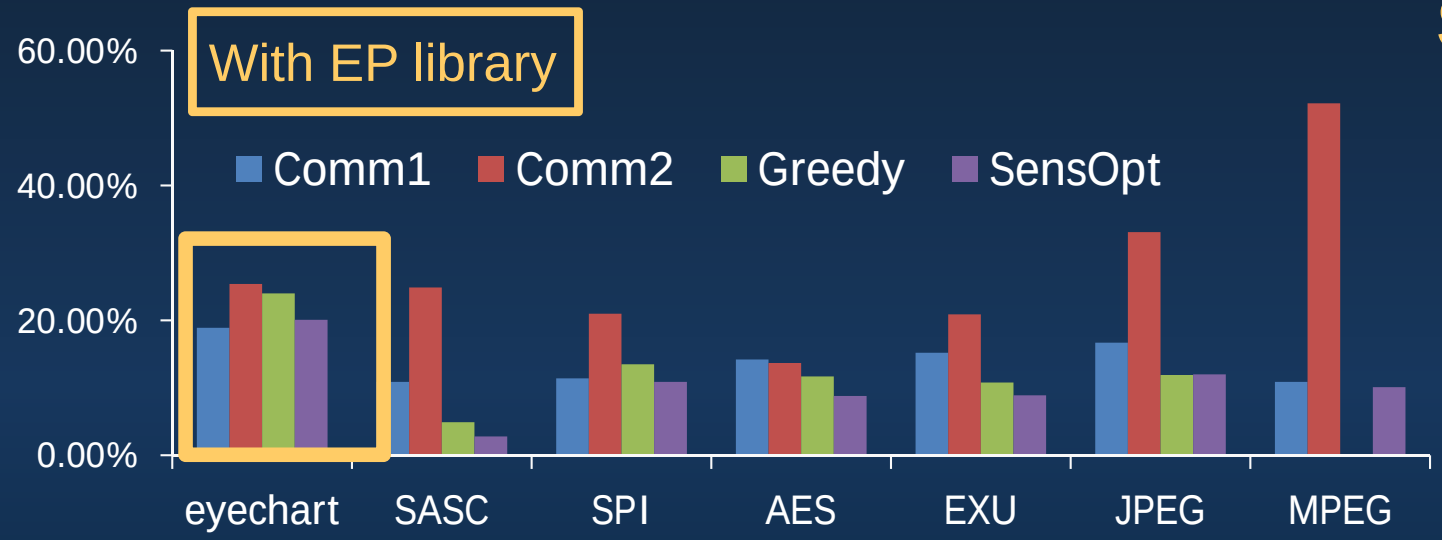
Generated Realistic Benchmarks

- Target benchmarks
  - SASC, SPI, AES, JPEG, MPEG (from *OpenCores*)
  - EXU (from *OpenSPARC T1*)
- Characteristic parameters of real and generated benchmarks

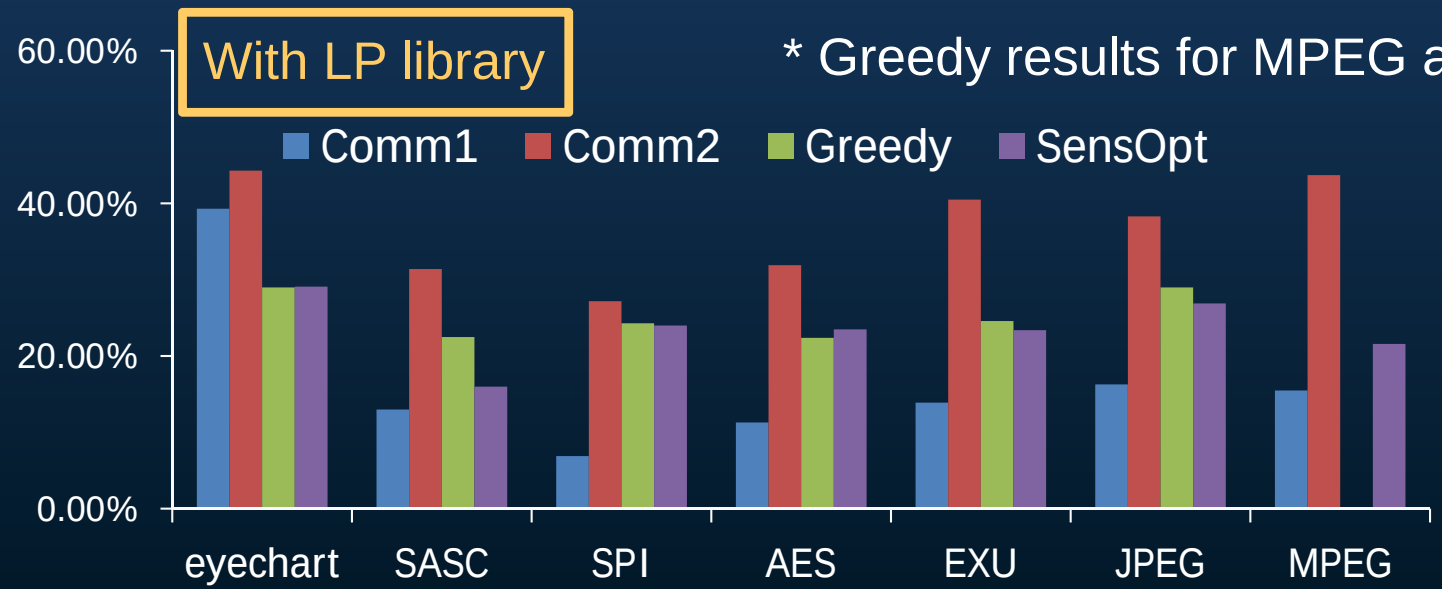
	data depth	#instance	real designs		generated	
			Rent param.	net degree	Rent param.	net degree
SASC	20	624	0.858	2.06	0.865	2.06
SPI	33	1092	0.880	1.81	0.877	1.80
EXU	31	25560	0.858	1.91	0.814	1.90
AES	23	23622	0.810	1.89	0.820	1.88
JPEG	72	141165	0.721	1.84	0.831	1.84
MPEG	33	578034	0.848	1.59	0.848	1.60

Suboptimality of Heuristics

- Suboptimality w.r.t. known optimal solutions for generated realistic benchmarks



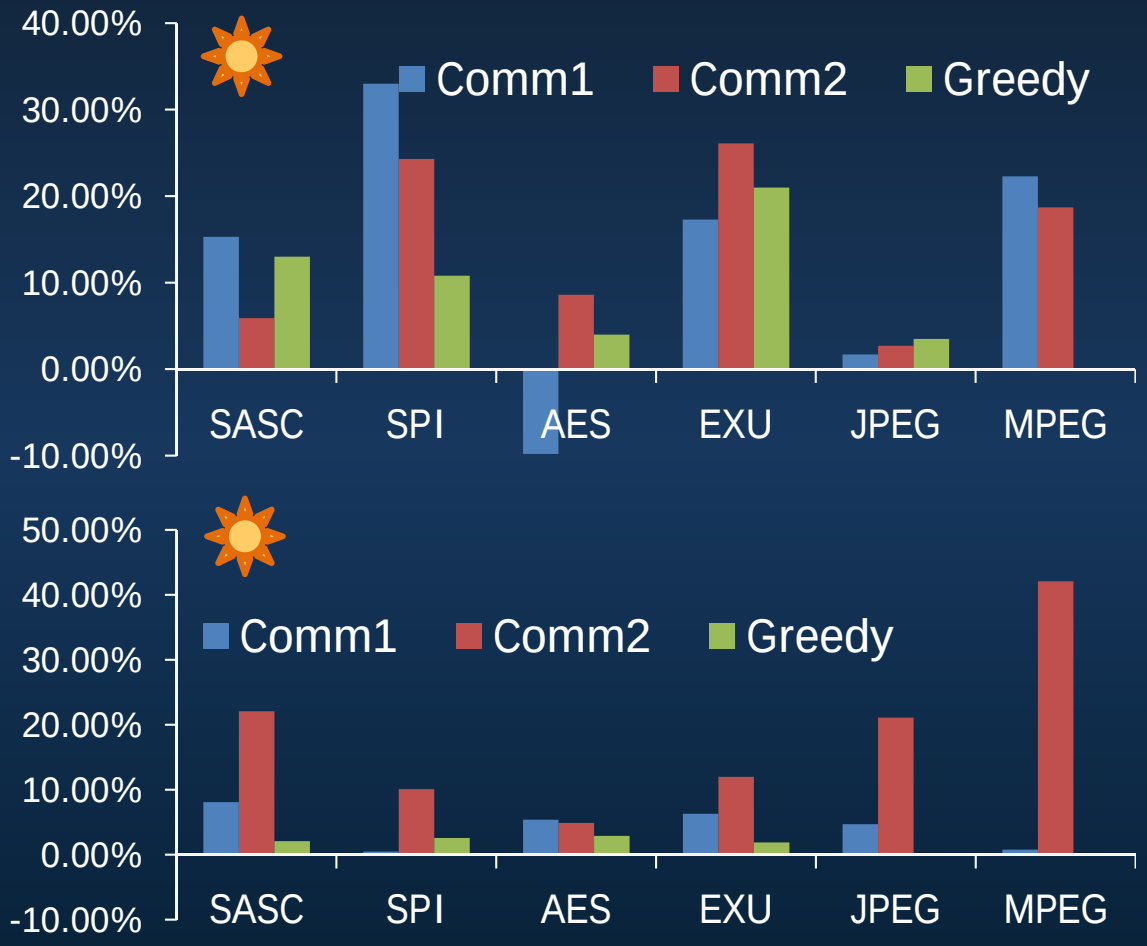
Vt swap context –
up to 52.2%
avg. 16.3%



Gate sizing context –
up to 43.7%
avg. 25.5%

Comparison w/ Real Designs

- Suboptimality versus one specific heuristic (SensOpt)



Real designs and real delay/leakage library (TSMC 65nm) case

Actual suboptimaltiy will be greater !

Suboptimality from our benchmarks

- Discrepancy: simplified delay model, reduced library set, ...

Conclusions

- A new benchmark generation technique for gate sizing → construct realistic circuits with known optimal solutions
- Our benchmarks enable systematic and quantitative study of common sizing heuristics
- Common sizing methods are suboptimal for realistic benchmarks by up to 52.2% (Vt assignment) and 43.7% (sizing)
- http://vlsicad.ucsd.edu/SIZING/

Ongoing Work

- Analyze discrepancies between real and artificial benchmarks
- Handle more realistic delay model
  - Use realistic delay library in the context of realistic benchmarks with tight upper bounds
- Alternate approach for netlist generation
  - (1) cutting nets in a real design and find optimal solution → (2) reconnecting the nets keeping the optimal solution

Thank you