Subgraph Extraction-based Feedback-guided Iterative Scheduling For HLS

Hanchen Ye[1], David Z. Pan[2], Chris Leary[3], Deming Chen[1], and Xiaoqing Xu[4]

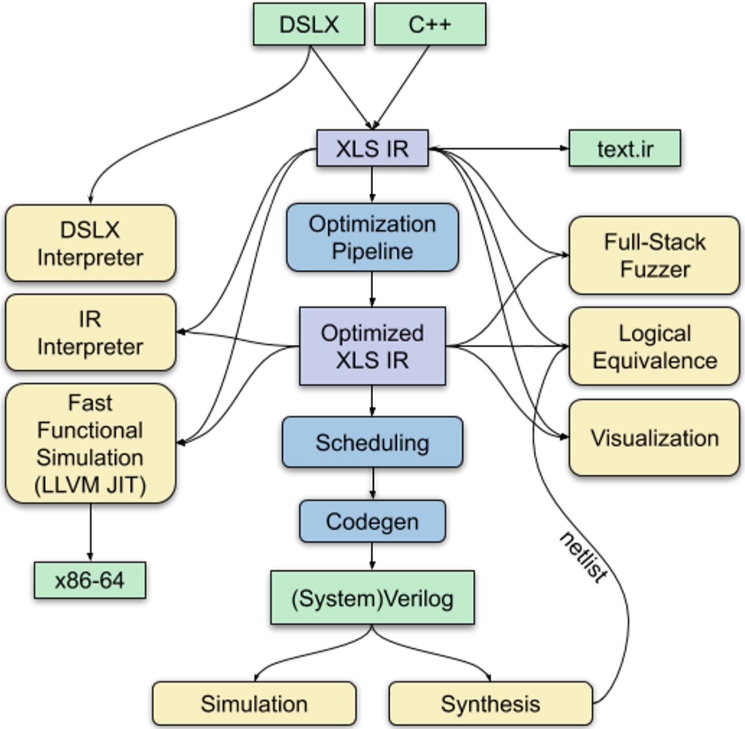
[1]University of Illinois at Urbana-Champaign; [2]University of Texas at Austin; [3]Google; [4]X, the moonshot factory

March 25, 2024

Background and Motivation

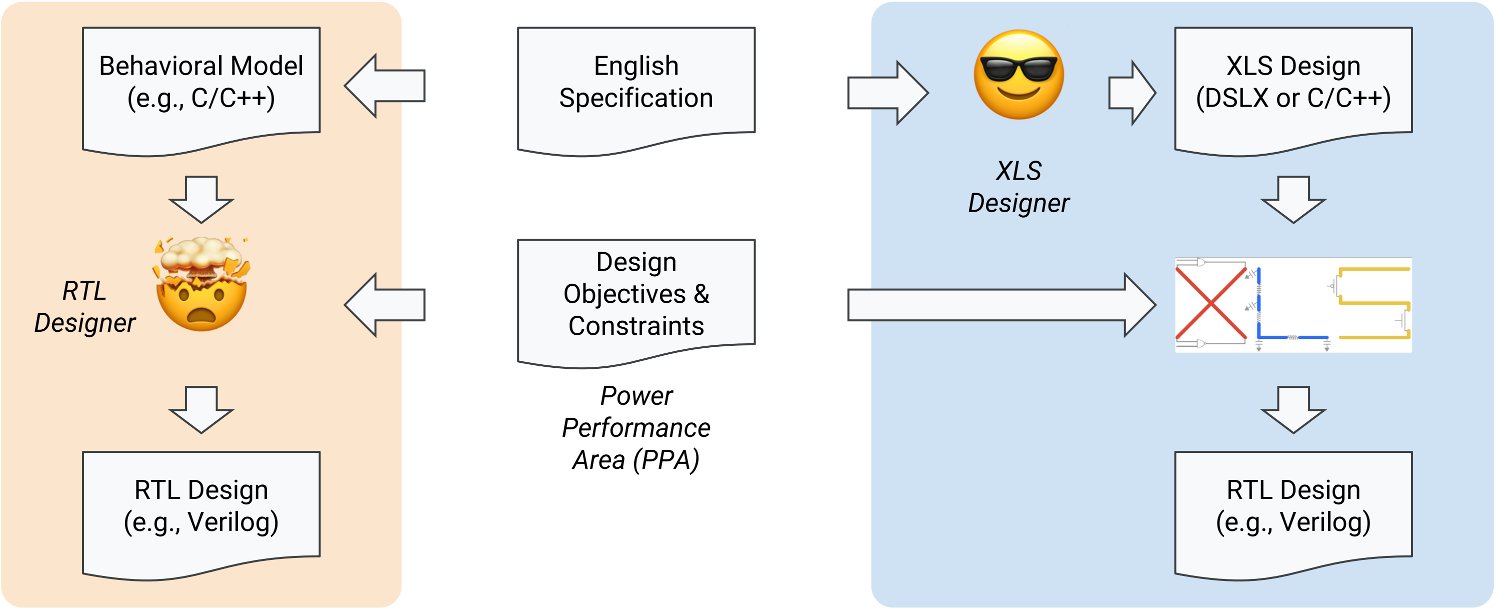
XLS: Accelerated HW Synthesis

- Takes high-level algorithmic description as input
  - C++ with customized compiler directives
  - DSLX, XLS domain-specific language
- Code optimizations
  - Constant Propagation
  - Dead-code elimination
  - Strength reduction
  - … …
- Generates Verilog as output
  - **Pipeline scheduling** (e.g., SDC scheduling [1])
  - Verilog code-generation
- Verification utilities
  - Functional simulation (with LLVM JIT [2])
  - Full-stack fuzzing
  - Logical equivalence check (with Z3 [3])



1. An efficient and versatile scheduling algorithm based on SDC formulation (paper)
2. JIT: Just-in-time compilation (wiki)
3. Z3: A satisfiability modulo theories (SMT) theorem prover (github)

XLS: Accelerated HW Synthesis (Cont’d)



Behavioral Model (e.g., C/C++)

RTL Designer

RTL Design (e.g., Verilog)

English Specification

Design Objectives & Constraints

Power Performance Area (PPA)

XLS Designer

XLS Design (DSLX or C/C++)

RTL Design (e.g., Verilog)

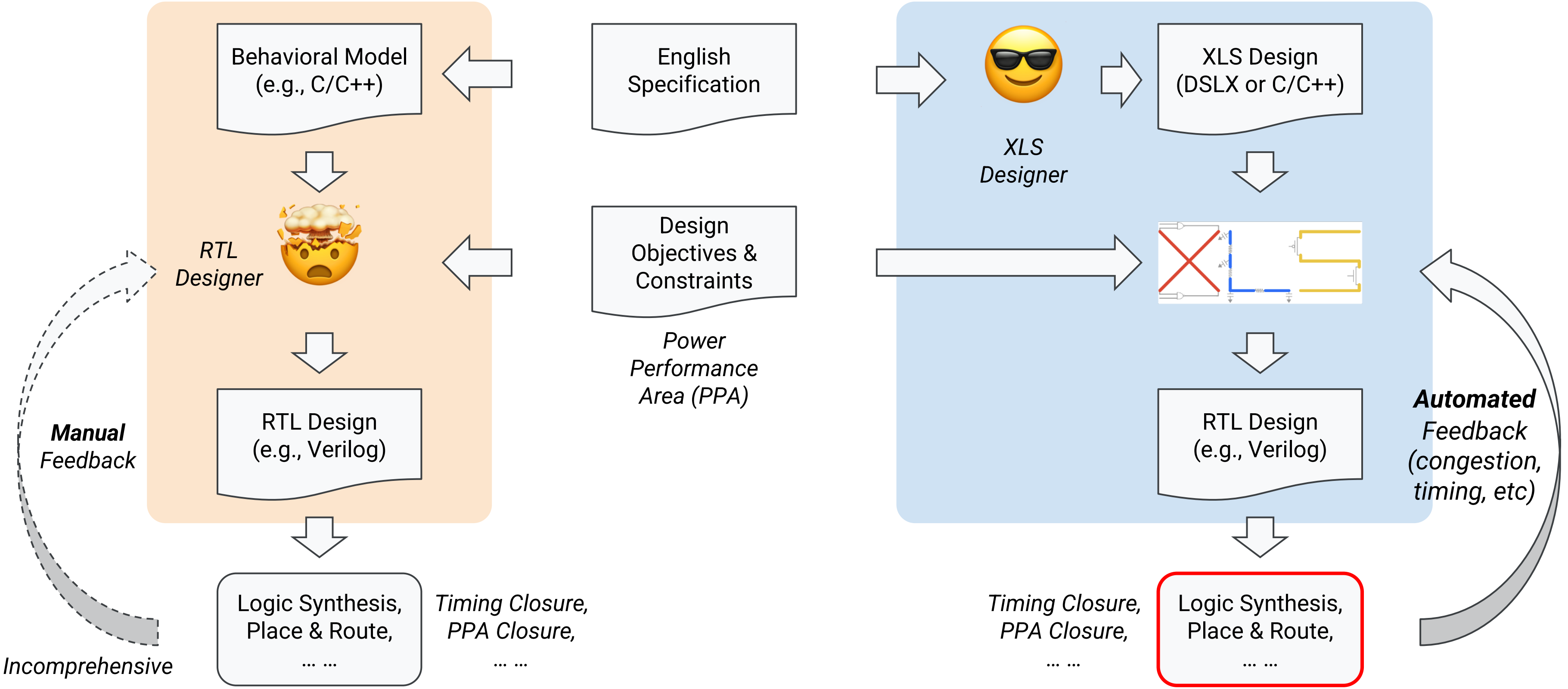
RTL Design Flow

- **Manual** optimization and scheduling
- **Long** design cycle
- **Low** portability against different PDK or PPA requirements

XLS Design Flow

- **Automated** optimization and scheduling
- **Short** design cycle
- **High** portability against different PDK or PPA requirements

Automated feedback-directed optimization (FDO)



Behavioral Model (e.g., C/C++)

RTL Designer

RTL Design (e.g., Verilog)

Logic Synthesis, Place & Route, … …

Timing Closure, PPA Closure, … …

***Manual** Feedback*

Incomprehensive

English Specification

Design Objectives & Constraints

Power Performance Area (PPA)

XLS Designer

XLS Design (DSLX or C/C++)

RTL Design (e.g., Verilog)

Timing Closure, PPA Closure, … …

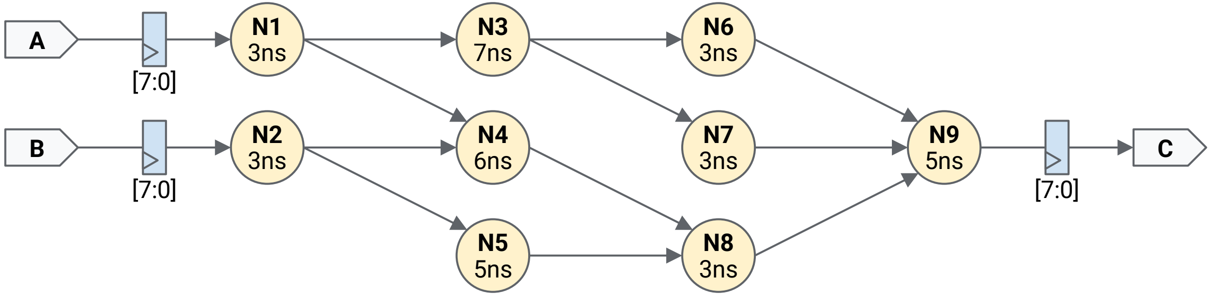
Logic Synthesis, Place & Route, … …

***Automated** Feedback (congestion, timing, etc)*

Feedback-guided Iterative SDC Scheduling

What is pipeline scheduling?

Unscheduled XLS Graph

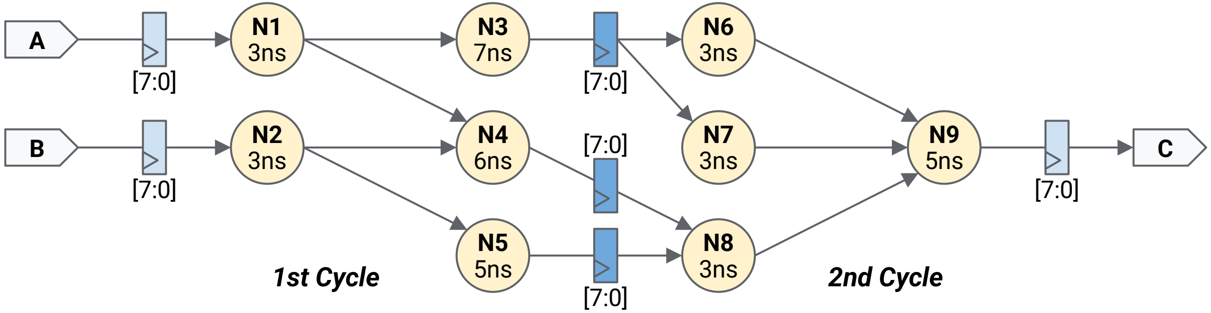


Pipeline Scheduling in XLS

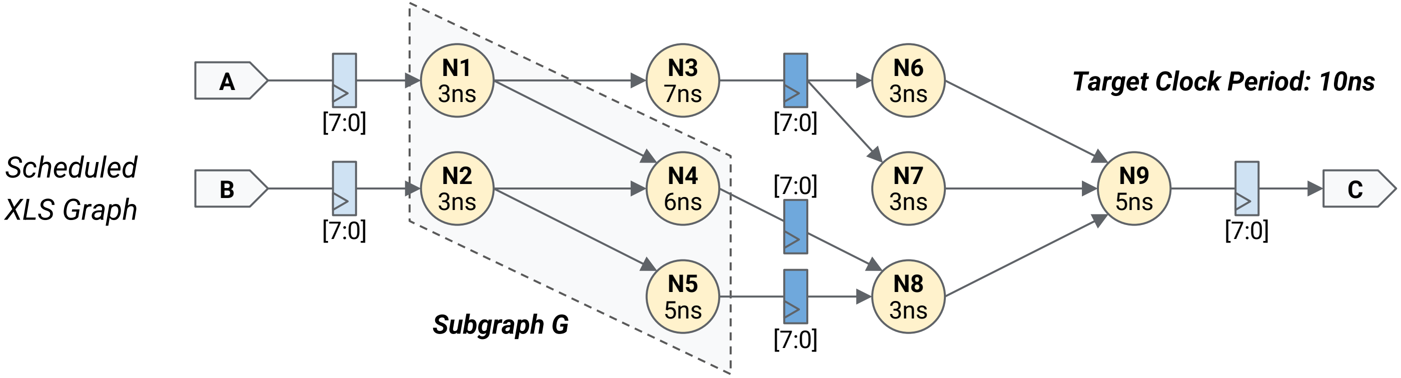
Target Clock Period: 10ns

Objective: Minimize register number

Scheduled XLS Graph

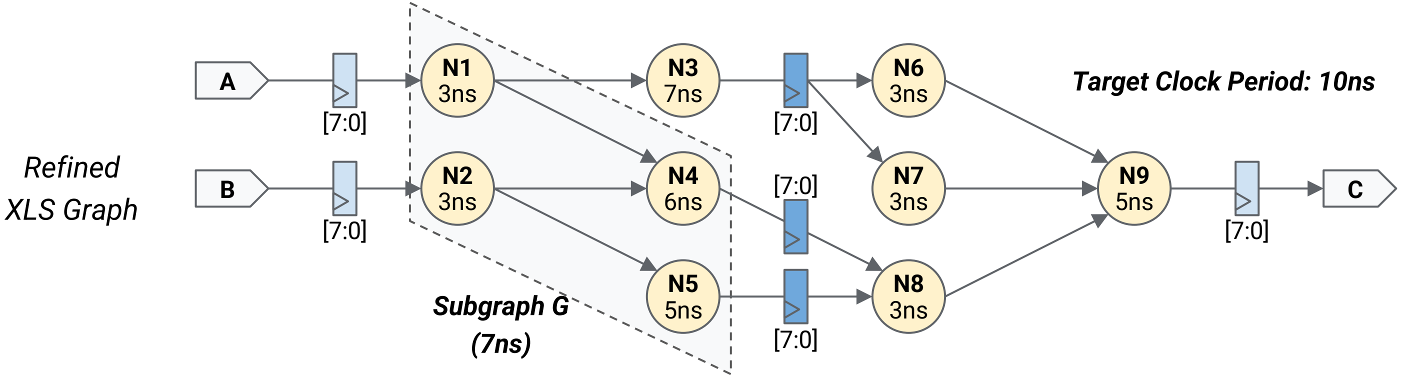


Intuition behind feedback-guided scheduling



Without feedback: ***Delay$_G$*** **= 9ns**

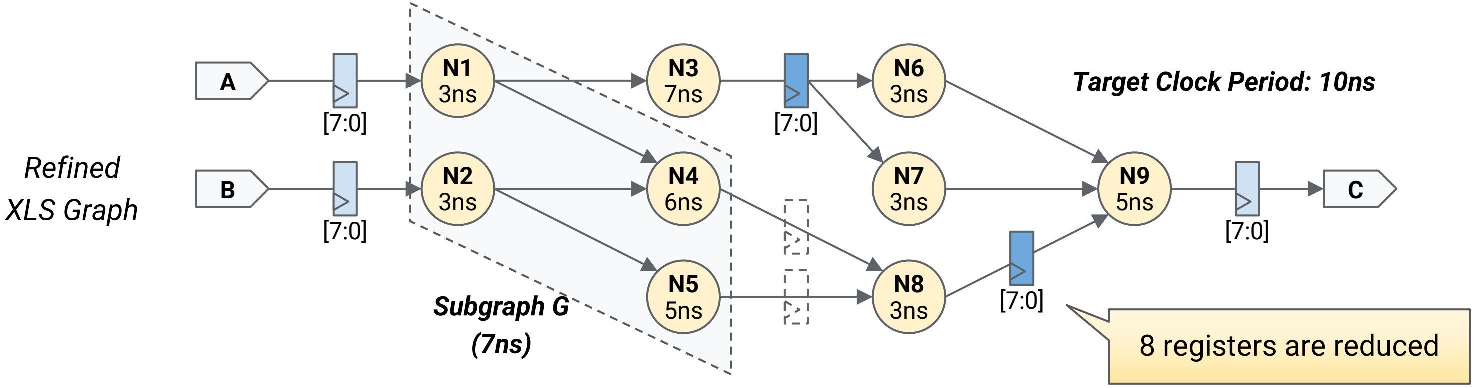
Intuition behind feedback-guided scheduling (Cont’d)



~~Without feedback: ***Delay$_G$* = 9ns**~~

With feedback (e.g., OpenROAD): ***Delay$_G$* = 7ns**

Intuition behind feedback-guided scheduling (Cont'd)



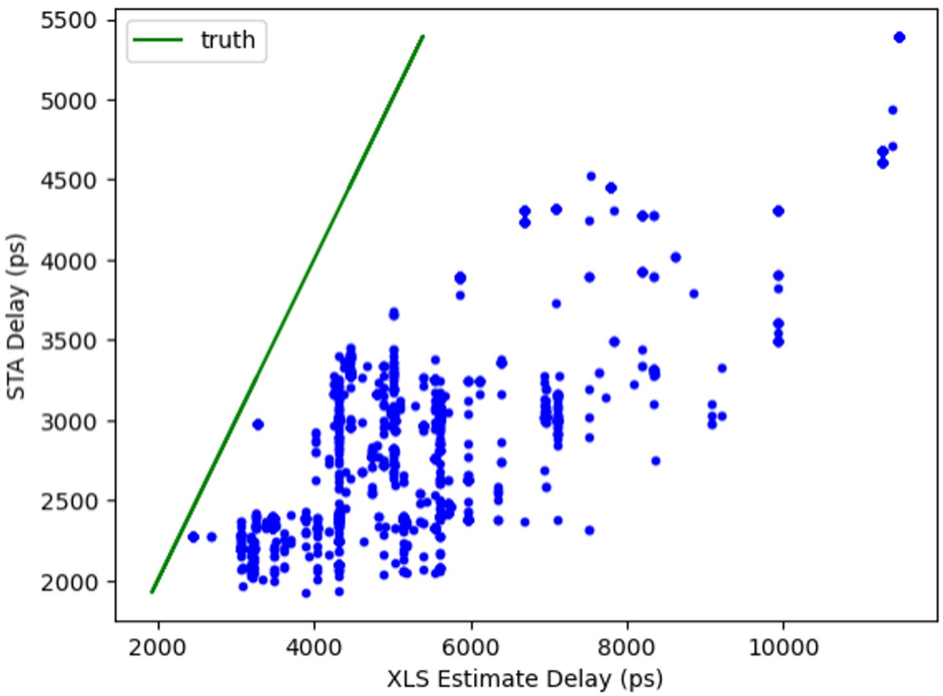
~~Without feedback: ***Delay$_G$* = 9ns**~~

With feedback (e.g., OpenROAD): ***Delay$_G$* = 7ns**

Q: Where does the difference come from?

A: Mainly comes from inter-node optimizations in downstream tools, such as logic synthesis.

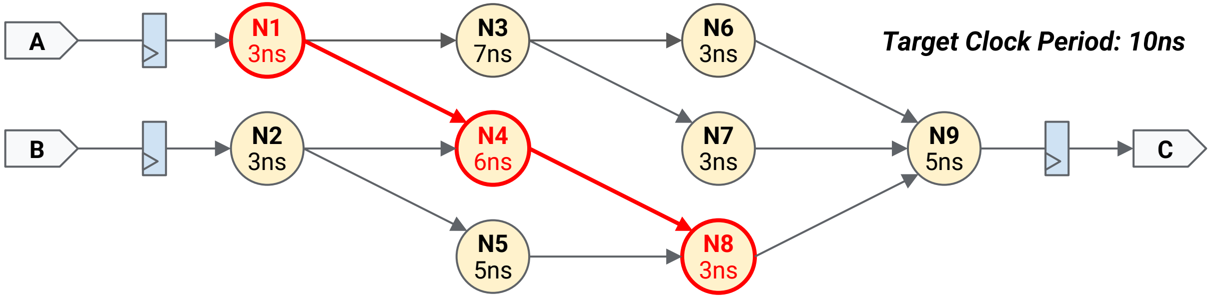
XLS delay estimation study



- **Design:** 6912 different design points of a 4-ways 24-bits floating-point adder
- **Technology:** SkyWater 130nm (SKY130)
- **Evaluation:** Yosys synthesis + OpenSTA

- Root mean square error (RMSE): 3478.0

Original pipeline scheduling in XLS



Unscheduled XLS Graph

Target Clock Period: 10ns

SDC (System of Difference Constraints) Scheduling [1]

Variables:

`cycle_1`
`cycle_2`
… …
`cycle_9`

Timing Constraints:

Delay_1_8 = 12ns > 10ns ⇒ `cycle_8 - cycle_1 >= 1`

Delay_2_8 = 12ns > 10ns ⇒ `cycle_8 - cycle_2 >= 1`

… …

(for each path longer than 10ns)

Def-use Constraints, Resource Constraints, etc.

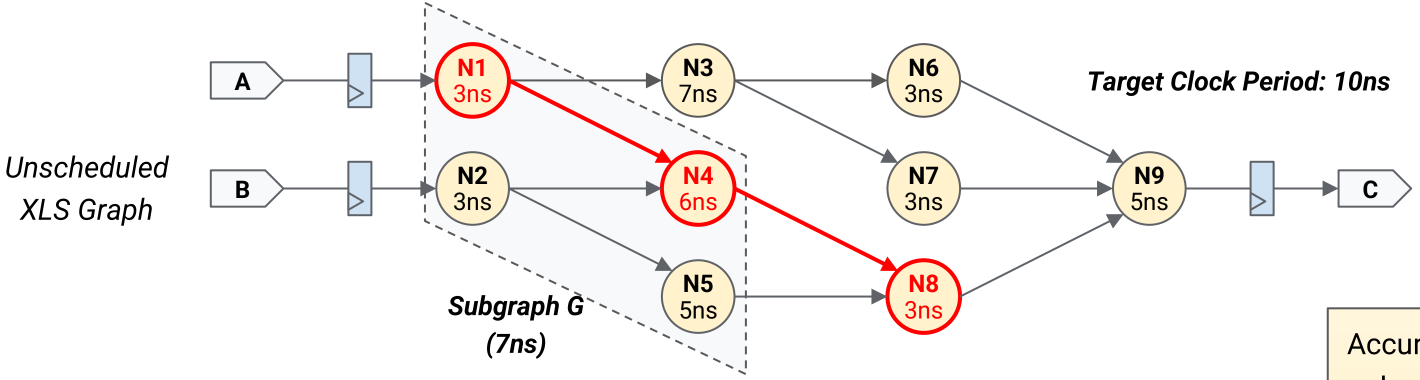
→

Minimize Register Number

Linear Programming Problem

[1] An efficient and versatile scheduling algorithm based on SDC formulation (paper)

SDC reformulation with feedbacks



SDC (System of Difference Constraints) Scheduling [1]

Accurate feedbacks
⇒ Less constraints
⇒ Larger search space
⇒ Better results

Variables:

Timing Constraints:

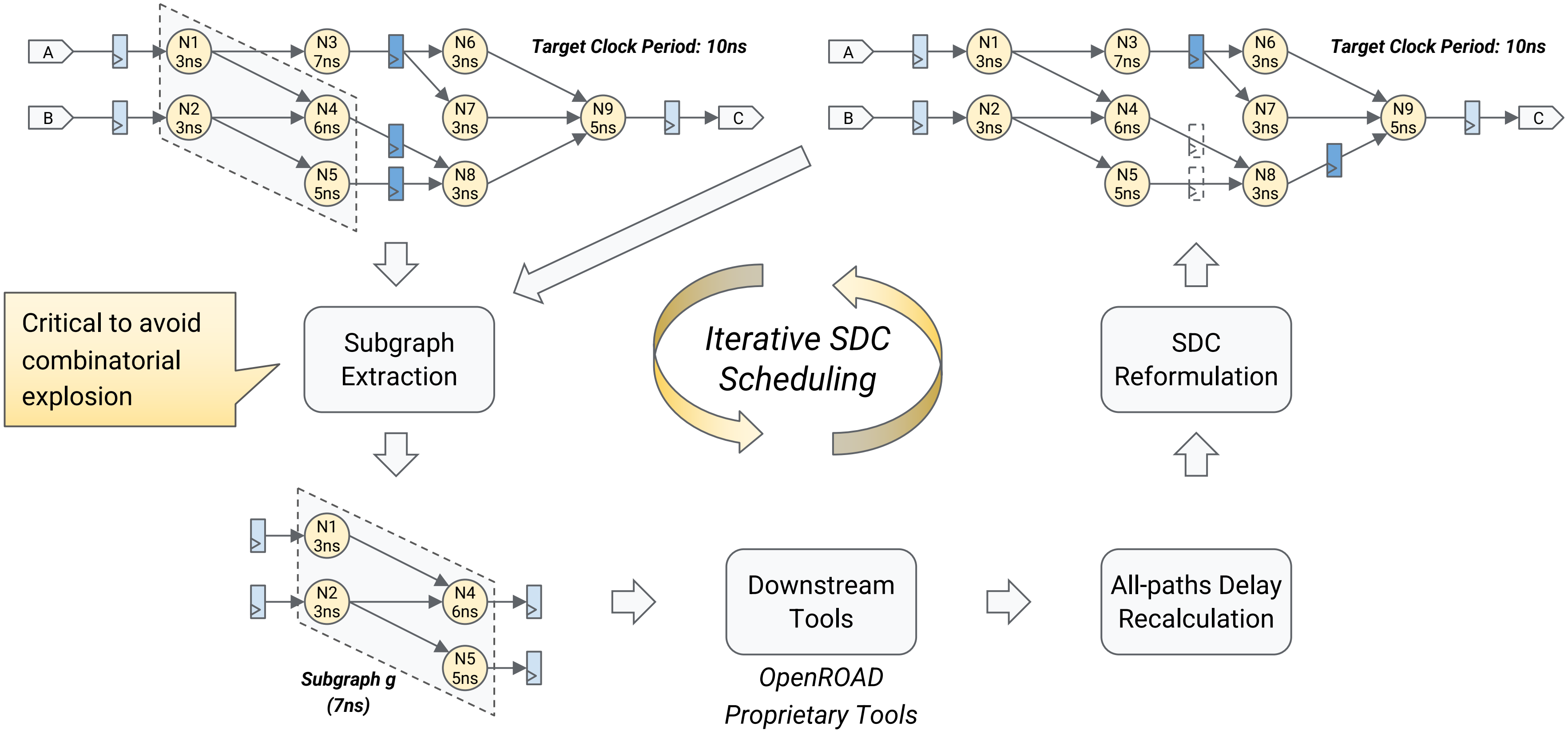
cycle_1	*Delay_1_8 <= 7ns + 3ns* ⇒ ~~cycle_8 - cycle_1 >= 1~~
cycle_2	*Delay_2_8 <= 7ns + 3ns* ⇒ ~~cycle_8 - cycle_2 >= 1~~
… …	… …
cycle_9	(for each path longer than 10ns)

Def-use Constraints, Resource Constraints, etc.

Minimize Register Number ⇒ Linear Programming Problem

[1] An efficient and versatile scheduling algorithm based on SDC formulation (paper)

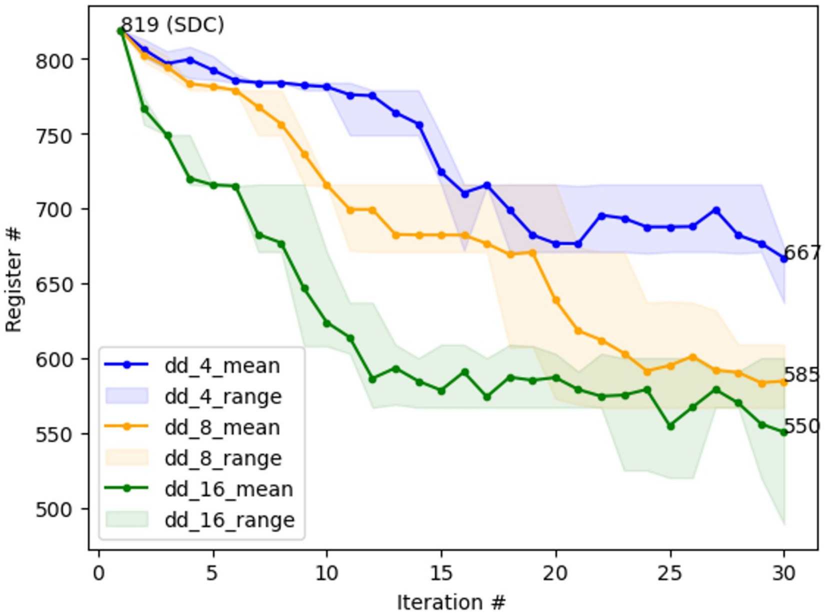
Automated iterative SDC scheduling



Critical to avoid combinatorial explosion

Subgraph Selection Strategy Studies

Delay-driven path extraction

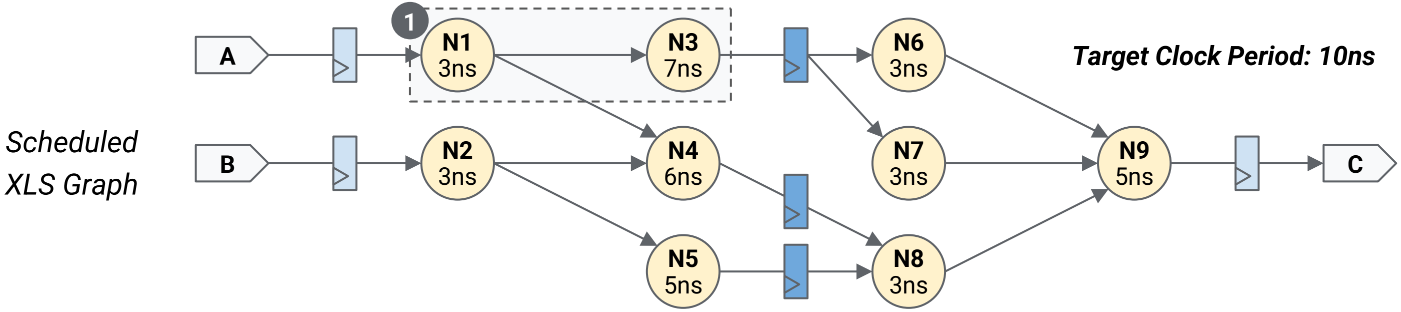


- **Design:** 4-ways 24-bits floating-point adder
- **Technology:** SkyWater 130nm (SKY130)
- **Clock period:** 2500ps (400MHz)

- **Settings:** 4 (blue)/8 (orange)/16 (green) longest paths per iteration
- After 30 iterations, 16-paths strategy reduces register number to 550 (**-32.8%** compared to the original SDC scheduling in XLS)

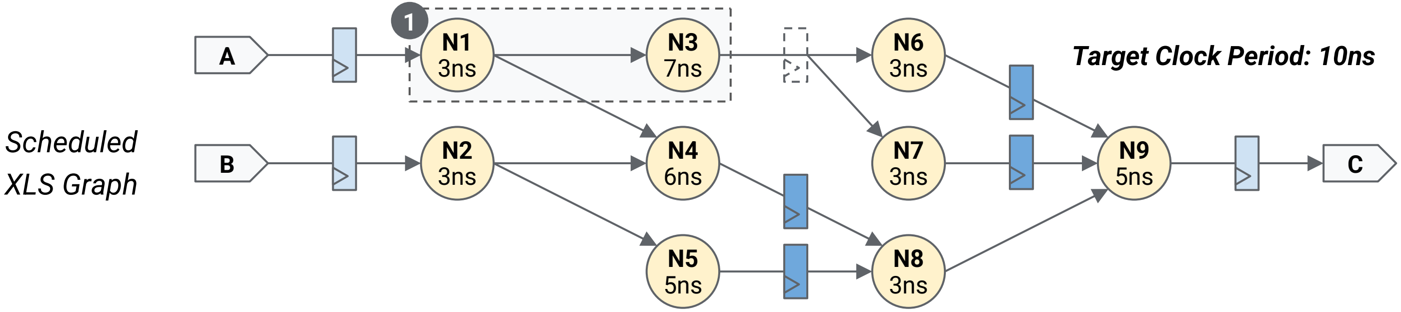
Path: a chain of nodes from register to register

Fanout-driven path extraction



- Path ❶
  - ○ Delay: **10ns**
  - ○ Target node fanout: **2**

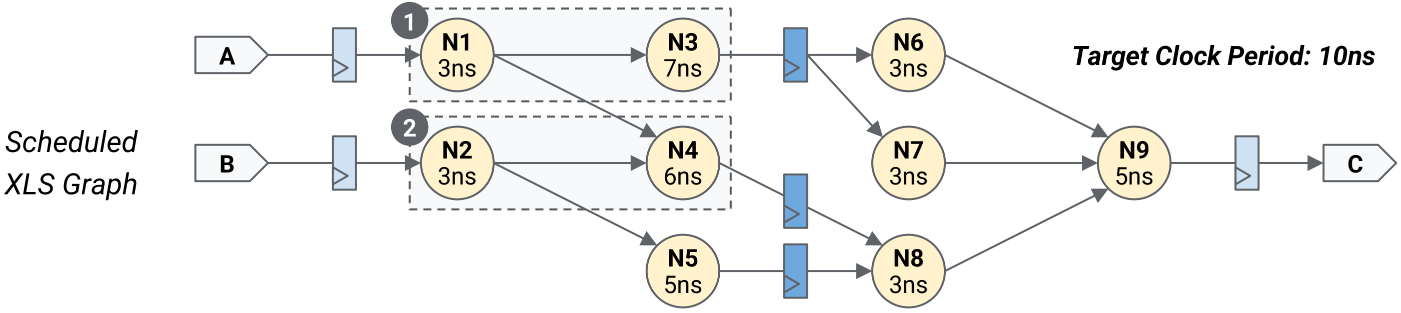
Fanout-driven path extraction (Cont'd)



- Path ❶
  - Delay: **10ns**
  - Target node fanout: **2**

❌

Fanout-driven path extraction (Cont’d)



- Path ❶
    - Delay: **10ns**
    - Target node fanout: **2**

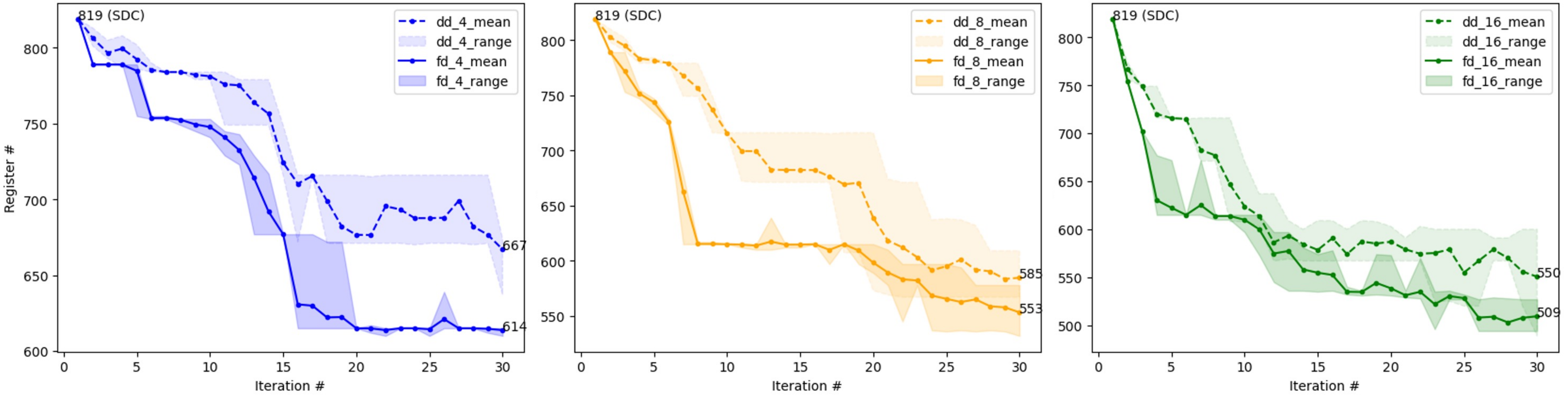
✗

- Path ❷
    - Delay: **9ns**
    - Target node fanout: **1**

✓

Fanout-driven strategy: Target node *bitcount/fanout*, delay as tie breaker

Fanout-driven path extraction (Cont'd)

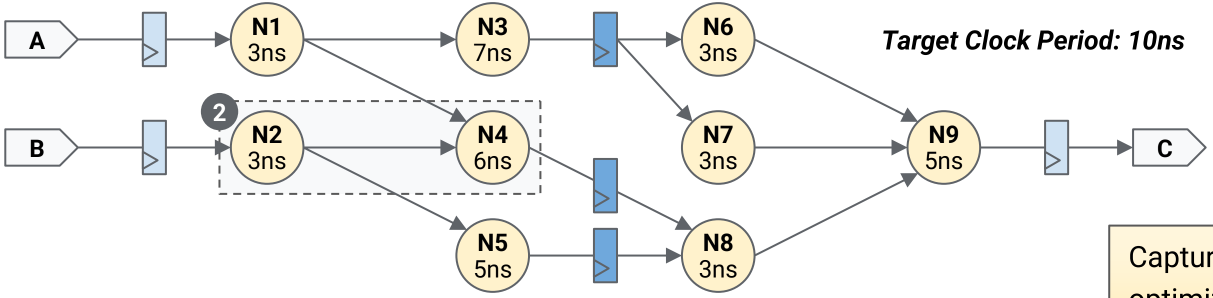


Fanout-driven path extraction vs. Delay-driven path extraction

- **Settings:** 4/8/16 delay-driven (dash)/fanout-driven (solid) paths per iteration
- After 30 iterations, fanout-driven strategy reduces register number to 509 **(-37.9%)**

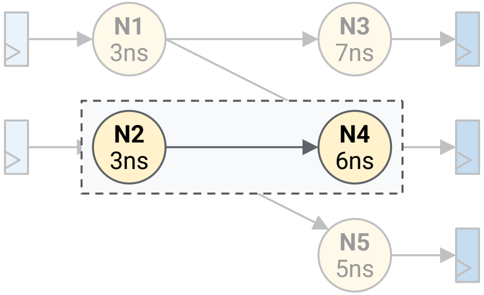
Cone or window extraction

Scheduled XLS Graph

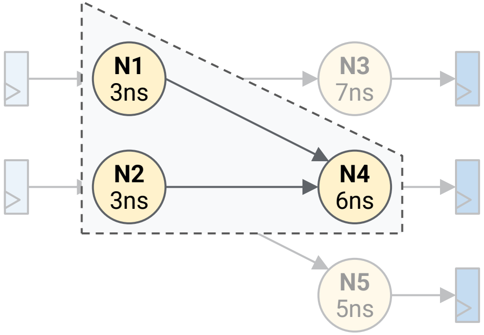


Target Clock Period: 10ns

Capture more inter-node optimizations

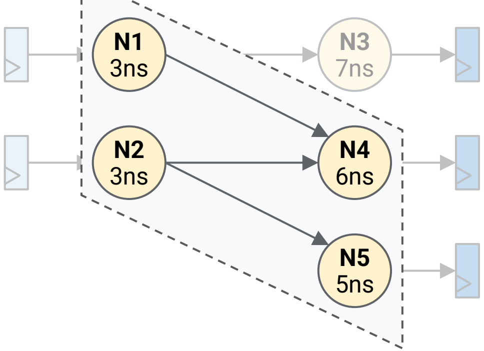


Path extraction



Cone extraction

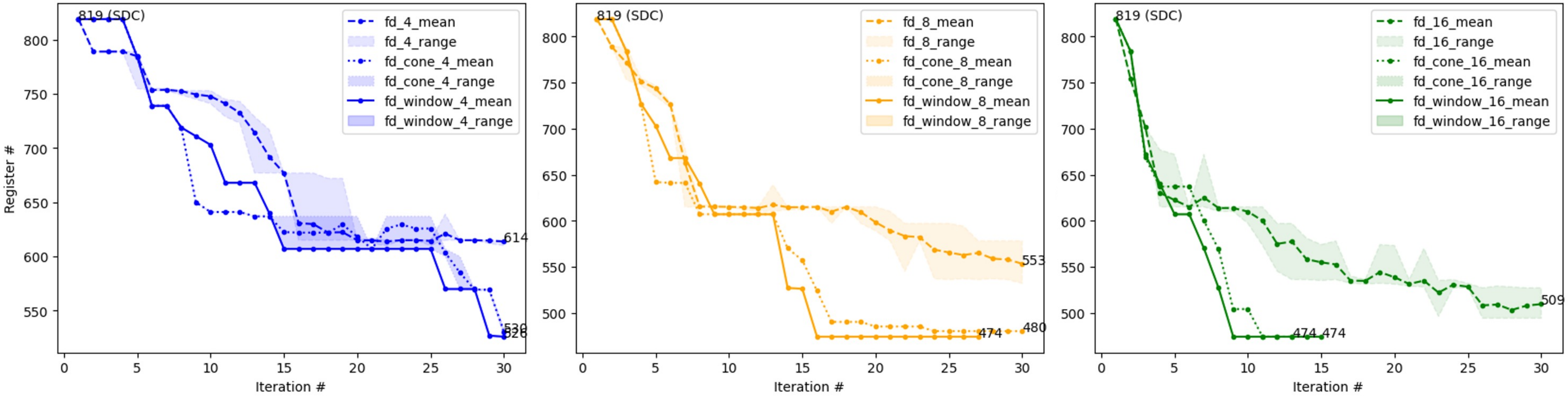
Multi-inputs single-output



Window extraction

Multi-inputs multi-outputs

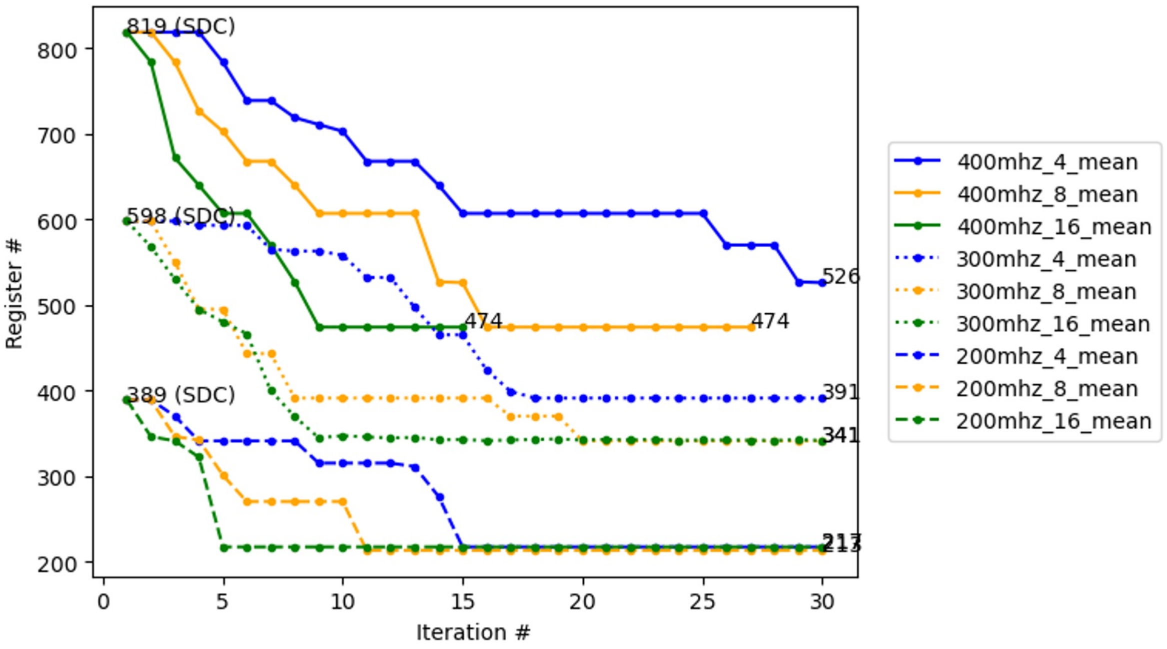
Cone or window extraction (Cont'd)



(Fanout-driven) Path extraction vs. Cone extraction vs. Window extraction

- **Settings:** 4/8/16 fanout-driven path (dash)/cone (dot)/window (solid) per iteration
- After 30 iterations, window-based strategy reduces register number to 474 **(-42.1%)**

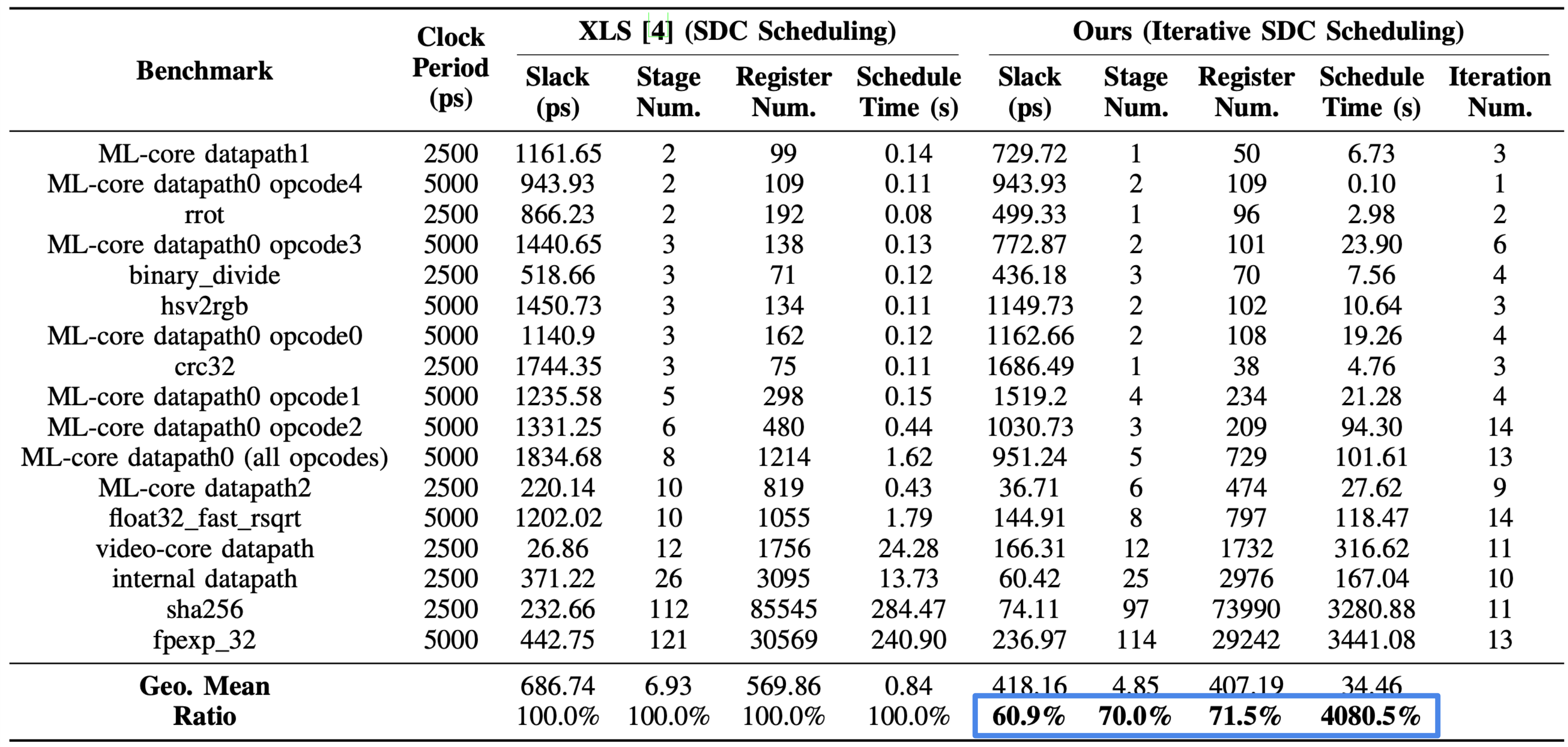
Timing constraints study



- **Clock period:**
  - 2500ps (400MHz, solid)
  - 3333ps (300MHz, dot)
  - 5000ps (200MHz, dash)
- **Settings:** 4 (blue)/8 (orange)/ 16 (green) fanout-driven windows per iteration

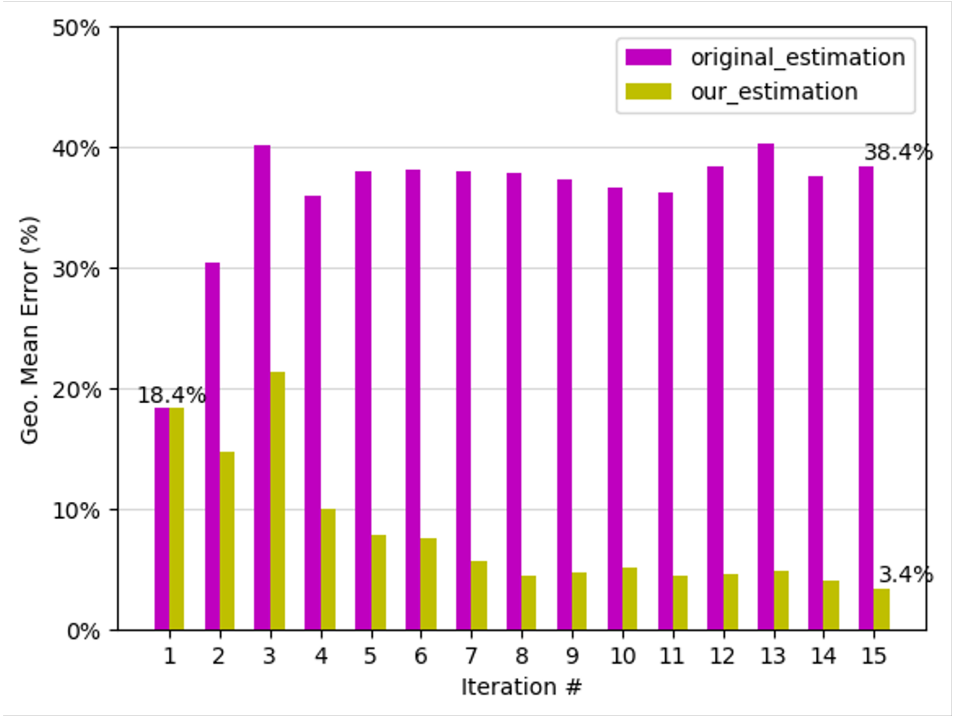
Evaluation

Full results



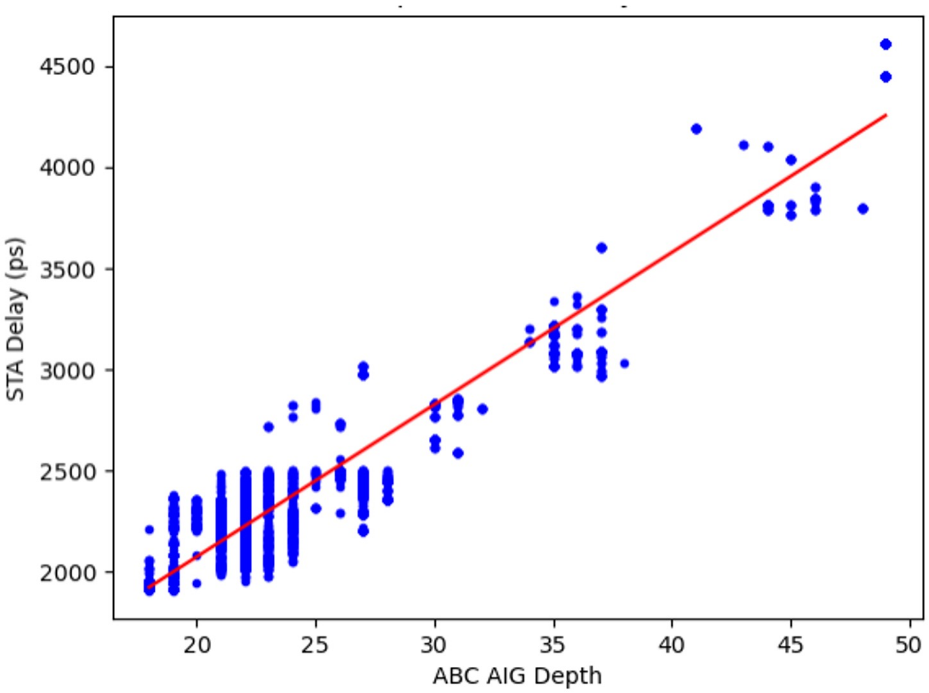
Benchmark	Clock Period (ps)	XLS [4] (SDC Scheduling)				Ours (Iterative SDC Scheduling)				
		Slack (ps)	Stage Num.	Register Num.	Schedule Time (s)	Slack (ps)	Stage Num.	Register Num.	Schedule Time (s)	Iteration Num.
ML-core datapath1	2500	1161.65	2	99	0.14	729.72	1	50	6.73	3
ML-core datapath0 opcode4	5000	943.93	2	109	0.11	943.93	2	109	0.10	1
rrot	2500	866.23	2	192	0.08	499.33	1	96	2.98	2
ML-core datapath0 opcode3	5000	1440.65	3	138	0.13	772.87	2	101	23.90	6
binary_divide	2500	518.66	3	71	0.12	436.18	3	70	7.56	4
hsv2rgb	5000	1450.73	3	134	0.11	1149.73	2	102	10.64	3
ML-core datapath0 opcode0	5000	1140.9	3	162	0.12	1162.66	2	108	19.26	4
crc32	2500	1744.35	3	75	0.11	1686.49	1	38	4.76	3
ML-core datapath0 opcode1	5000	1235.58	5	298	0.15	1519.2	4	234	21.28	4
ML-core datapath0 opcode2	5000	1331.25	6	480	0.44	1030.73	3	209	94.30	14
ML-core datapath0 (all opcodes)	5000	1834.68	8	1214	1.62	951.24	5	729	101.61	13
ML-core datapath2	2500	220.14	10	819	0.43	36.71	6	474	27.62	9
float32_fast_rsqrt	5000	1202.02	10	1055	1.79	144.91	8	797	118.47	14
video-core datapath	2500	26.86	12	1756	24.28	166.31	12	1732	316.62	11
internal datapath	2500	371.22	26	3095	13.73	60.42	25	2976	167.04	10
sha256	2500	232.66	112	85545	284.47	74.11	97	73990	3280.88	11
fpexp_32	5000	442.75	121	30569	240.90	236.97	114	29242	3441.08	13
Geo. Mean		686.74	6.93	569.86	0.84	418.16	4.85	407.19	34.46	
Ratio		100.0%	100.0%	100.0%	100.0%	**60.9%**	**70.0%**	**71.5%**	**4080.5%**	

Estimation error comparison



- Original XLS estimation (w/o feedback) and our estimation (w/ feedback) are compared with the post-synthesis STA result in each iteration
- After 15 iterations, the geometric mean error is reduce to **3.4%** with our approach

And-Inverter-Graph (AIG) depth study



- Proprietary tools are time-consuming and expensive
- AIG is a view of circuit design widely used in logic synthesis tools
- **AIG depth could be used for iterative optimization due to its linear correlation with STA delay**

Conclusion

- We propose an feedback-guided iterative SDC scheduling method in XLS, which can take feedbacks from downstream tools, e.g., OpenROAD, and refine the scheduling result in an iterative way.
- Based on our evaluation of 17 XLS benchmarks, the new method can reduce 28.5% registers in average compared to the original SDC algorithm.
- The code has been open-source at: https://github.com/google/xls.

Thanks for listening!

Q&A