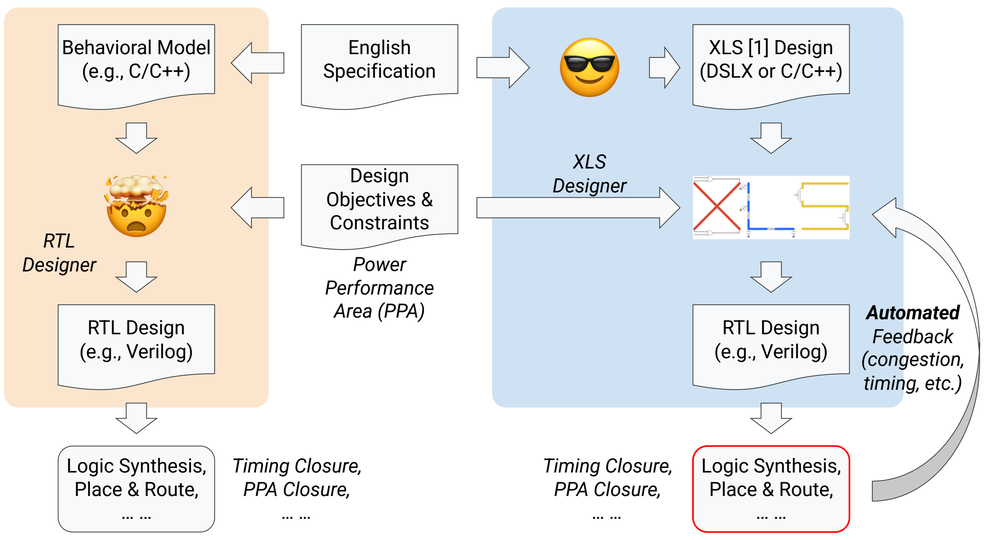
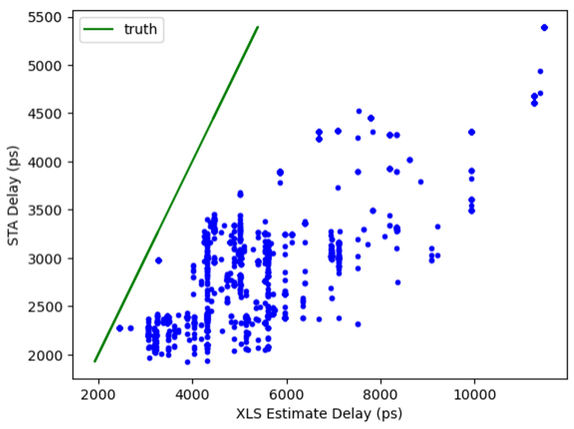
Subgraph Extraction-based Feedback-guided Iterative Scheduling For HLS

Hanchen Ye[1], David Z. Pan[24], 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

Background and Motivation

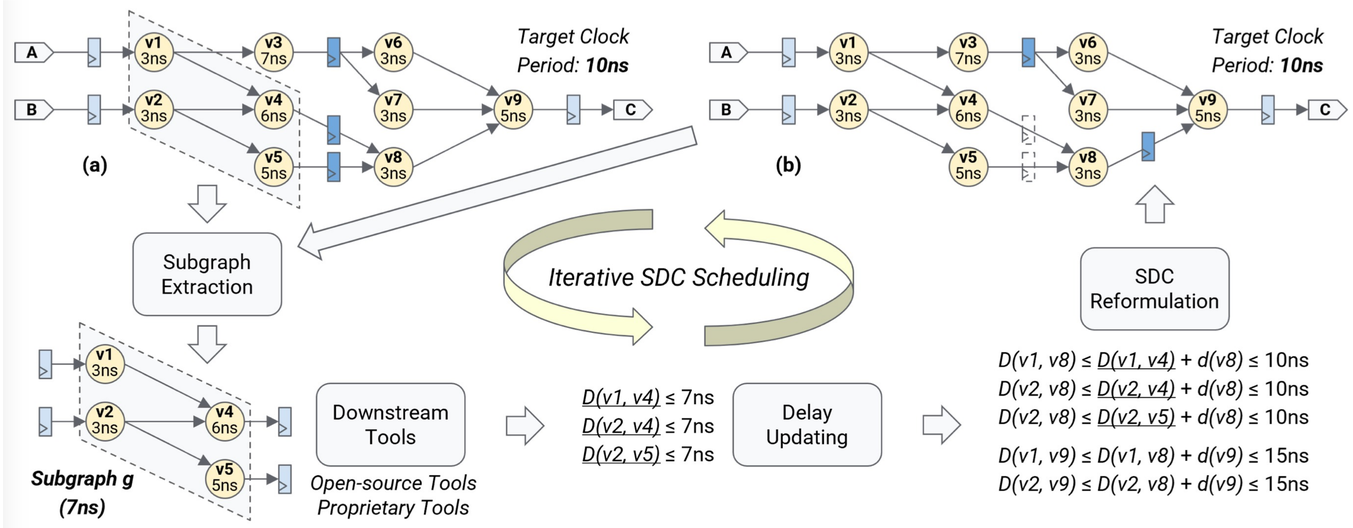




Post-synthesis STA vs. XLS-estimated critical path delay of 6912 designs

- **Evaluation:** Google XLS + Yosys synthesis + OpenSTA + SkyWater 130nm (SKY130)
- **Observation:** XLS-estimated delays (blue dots) exhibit significant deviation from the post-synthesis STA delays (the green line).
- These deviations create unused slack and present opportunities to refine scheduling quality, such as **reducing register usage.**

Iterative SDC Scheduling Algorithm



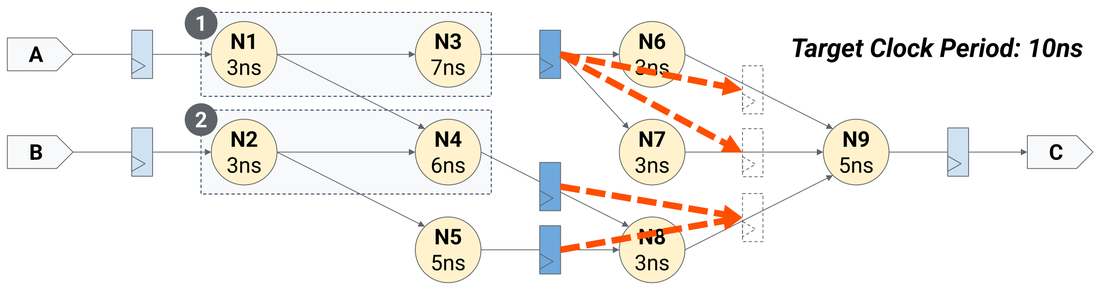
Variables:	*Original SDC [2] Timing Constraints:*		*Our ISDC Timing Constraints:*
`cycle_1`	*Delay_1_8 = 12ns > 10ns* ⇒ `cycle_8 - cycle_1 >= 1`	***Linear Programming Problem Reformulation***	*Delay_1_8 <= 7ns + 3ns* ⇒ ~~`cycle_8 - cycle_1 >= 1`~~
`cycle_2`	*Delay_2_8 = 12ns > 10ns* ⇒ `cycle_8 - cycle_2 >= 1`		*Delay_2_8 <= 7ns + 3ns* ⇒ ~~`cycle_8 - cycle_2 >= 1`~~
... ...	... ...		... ...
`cycle_9`	(for each path longer than 10ns)		(for each path longer than 10ns)

SDC (System of Difference Constraints) Scheduling Reformulation

- **Delay Updating:** In each iteration, the delay estimations between all pairs of node are recalculated based on the feedback from downstream tools.
- **SDC Reformulation:** In each iteration, the linear programming (LP) problem of SDC is reformulated based on the updated delay estimations.
- ***Intuition: Accurate feedback => less LP constraints => larger design space => better scheduling results.***

Code is fully open-source!

Subgraph Extraction Strategy



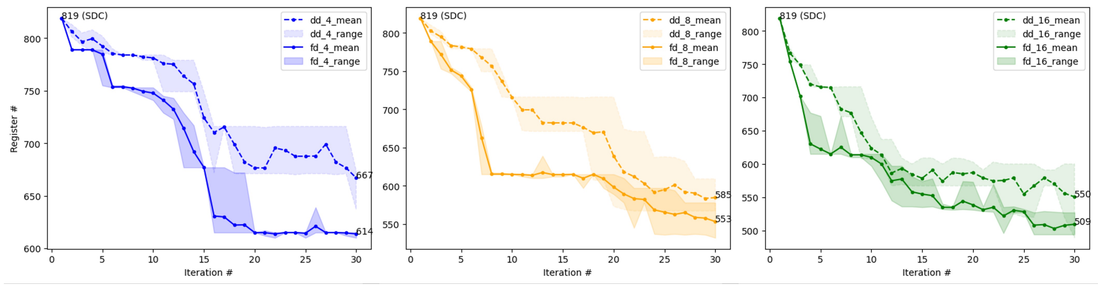
Fanout-driven vs. Delay-driven

- Path delay: **10ns**, Target node fanout: **2**
- Register usage increased

Path ❶: Delay-driven ✗

- Path delay: **9ns**, Target node fanout: **1**
- Register usage decreased

Path ❷: Fanout-driven ✓



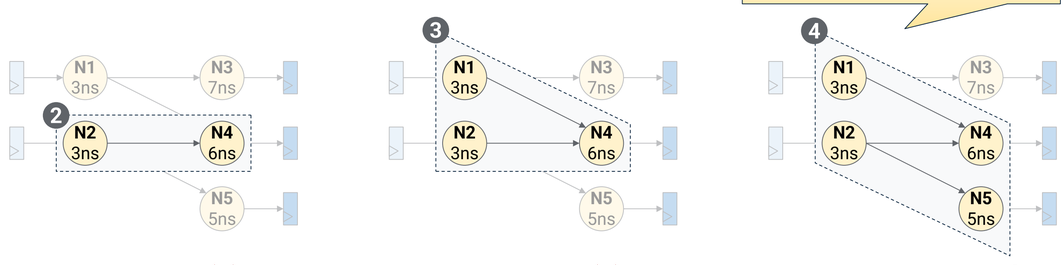
Fanout-driven path extraction vs. Delay-driven path extraction

- **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%)**

Window-based vs. Cone-based vs. Path-based

Window is derived by merging multiple cones that have different roots but share an identical or overlapping set of leaves.

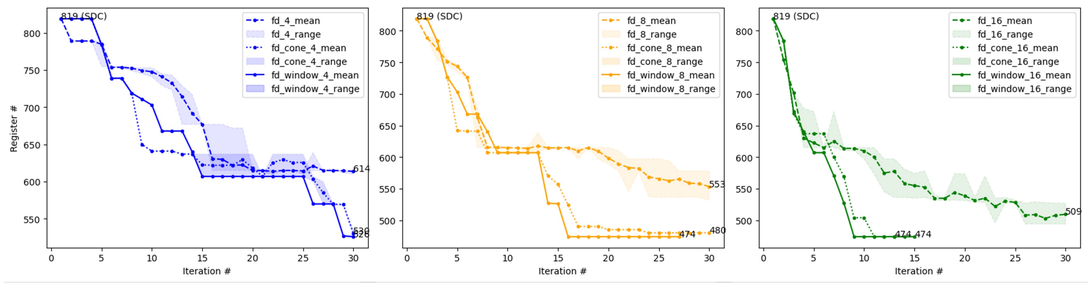
Capture more inter-node optimizations



Path-based ✗

Cone-based ✗
Multi-inputs single-output

Window-based ✓
Multi-inputs multi-outputs

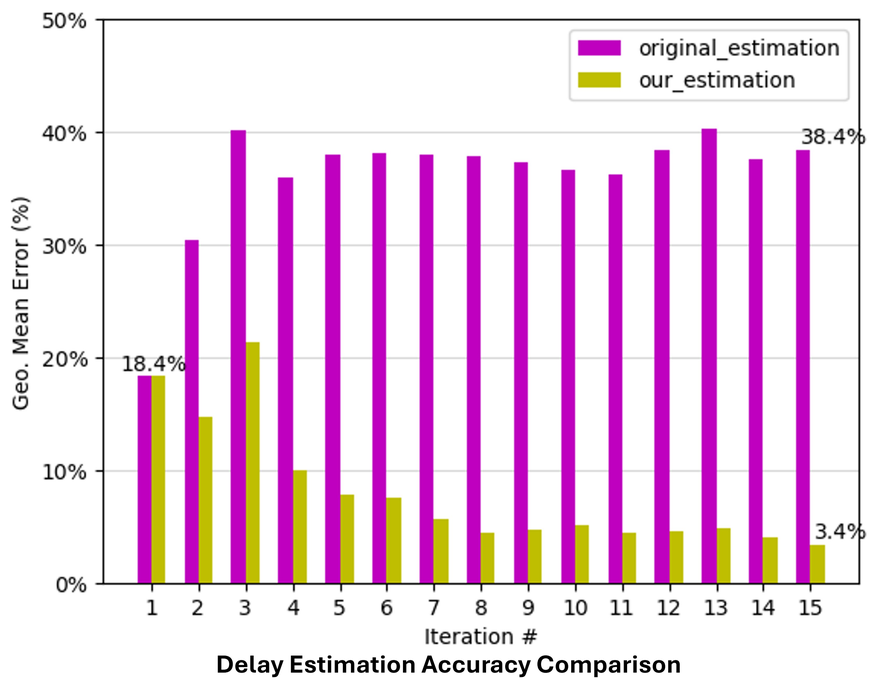


(Fanout-driven) Path extraction vs. Cone extraction vs. Window extraction

- **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%)**

Evaluation Results on 17 XLS Designs

Benchmark	Clock Period (ps)	XLS [1] (SDC Scheduling) Slack (ps)	Stage Num.	Register Num.	Schedule Time (s)	Ours (Iterative SDC Scheduling) 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%**	



Delay Estimation Accuracy Comparison

[1] The XLS Authors, "XLS: Accelerated HW synthesis," https://github.com/google/xls.

[2] J. Cong et al., "An efficient and versatile scheduling algorithm based on SDC formulation," in Proc. of DAC, 2006.