



Summary

Challenges:

- CPU implementation of transduction method for And-inverter Graph (AIG) in ABC is slow when number of inputs grows [1].
- Current CPU transduction framework uses Binary Decision Diagram (BDD) for functional manipulation which is not suitable for implementation.

Proposal: GPU-accelerated Efficient Transduction (**GTR**)

Results: GTR achieves **130x** speedup with slightly better depth control than &transduction in ABC.

Preliminary

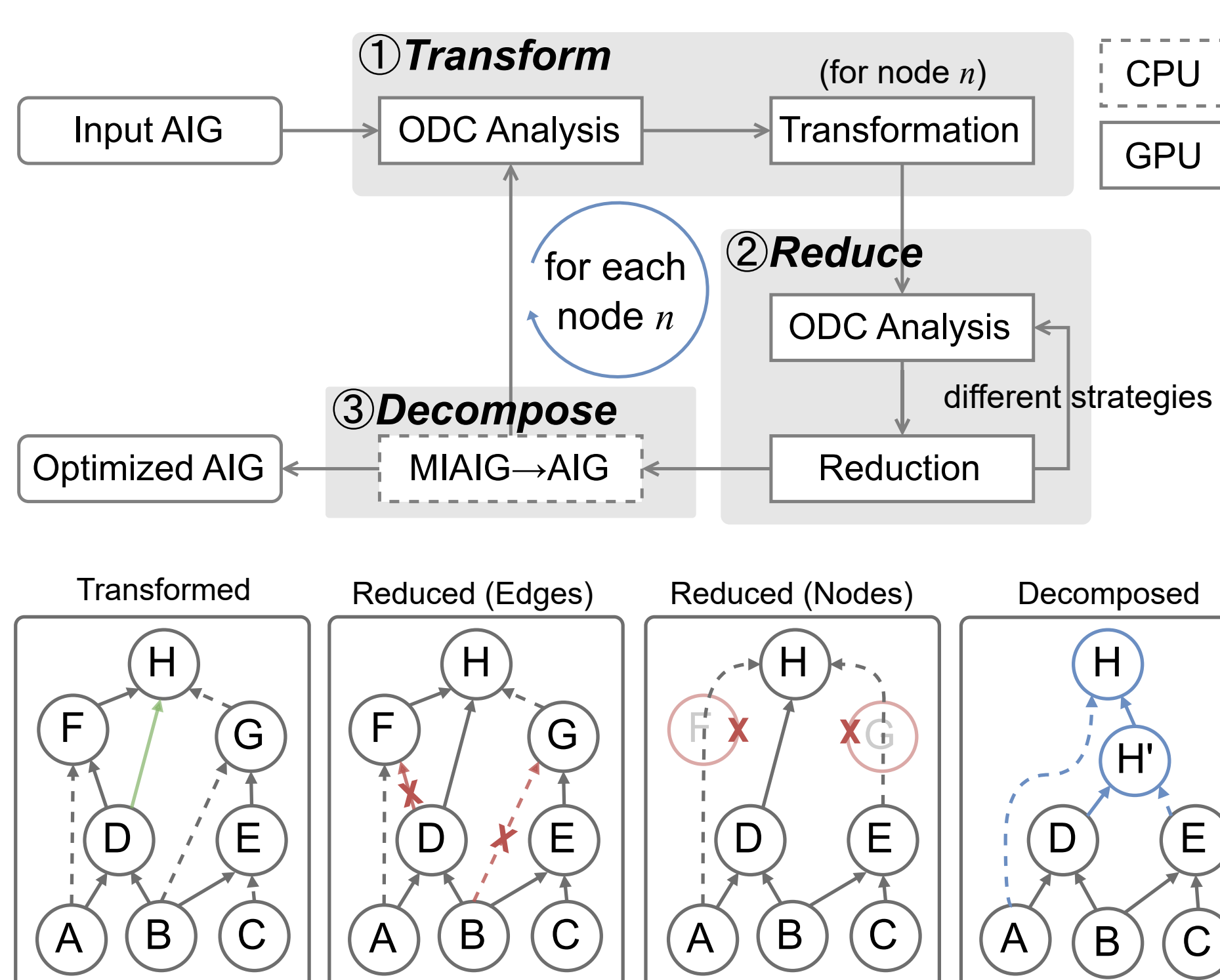
Multi-input AIG (MIAIG):

- A **Direct Acyclic Graph (DAG)** with primary inputs, AND nodes, inverter edges, and primary outputs.
- The AND nodes in MIAIG allows **more than 2 fanins**.

Observability Don't-cares (ODC) [2]:

- ODC **masks a node/edge's output** at POs.
- An input pattern (a specific set of Boolean values assigned to PIs) when a node or edge's change does **not affect any POs' value**.

Methodology



Transform

To create reconvergence and uncover optimization opportunities (connecting D to H)

- Candidate should not be in the TFO of the target.
- Input makes candidate=0 while target=1 doesn't exist.
- Depth-aware heuristic is used to further filter the legal candidates

Reduce

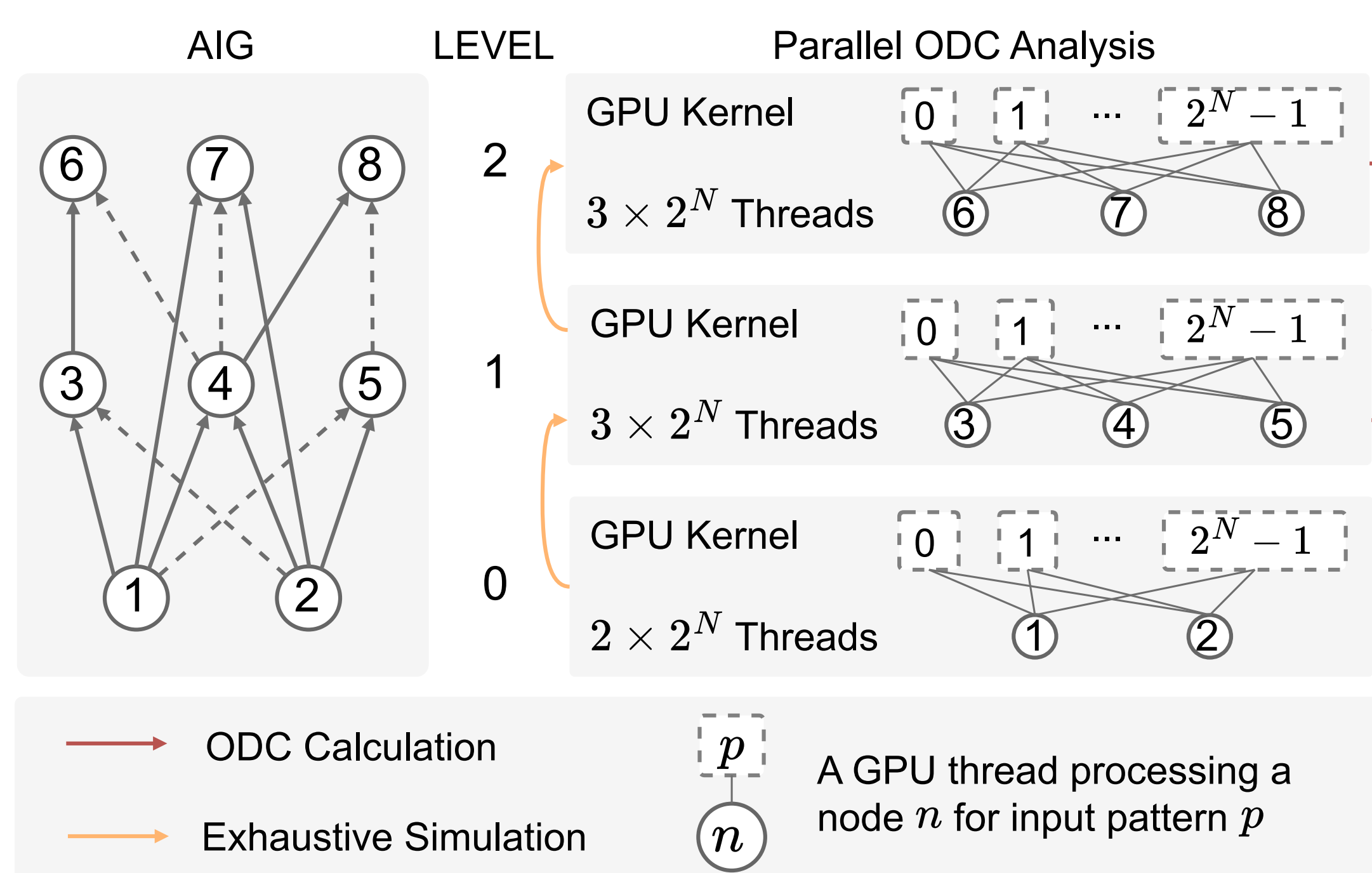
To remove redundant edges or nodes (disconnecting D to F, D to G, and then node D and F)

- By reversely propagating ODCs from POs, getting each node's ODC set
- Remove the constant edges first, and then nodes with 0 fanout or only 1 fanin are removed

Decompose

To break the multi-input AND into 2-input AND (decompose 3-input node H)

- Get standard AIG format after transform and reduce to calculate gain.
- If no gain, the modification of transform and reduce is discarded and next try will start.



Level- and input-pattern-based parallelism

The algorithm based on truth table, so there is no race condition among same-level nodes and different input patterns.

High memory consumption

Solution: bit-packing

- Pack Boolean value into 32-bit integer, optimized the storage of simulation truth table and ODC sets

Slow serialized modifications for packed bits

Solution: optimization #1

- Single thread process 32 Boolean value
- Thread-local memory copies are created

Slow and complex computations with branching

Solution: optimization #2

- Leverage bitwise XOR or AND operation

Experiment Results

Statistics				Single Transduction						Integrated Transduction					
				&transduction [1]			Ours			&transtoch [1]			Ours + CULS		
Case	#PIs	Size	Depth	Size	Depth	Time	Size	Depth	Time	Size	Depth	Time	Size	Depth	Time
ex48	16	1884	18	1507	265	437	1616	35	25	1385	168	606	1385	89	188
ex39	16	1944	19	954	101	144	1079	44	13	678	132	328	681	56	184
ex25	16	1949	20	922	117	184	1060	40	13	544	119	1392	549	46	186
ex30	16	2299	19	1242	197	786	1065	31	12	675	161	5451	675	41	184
ex26	17	3036	22	1166	175	302	1283	40	30	529	136	5530	546	40	186
ex27	18	4266	24	2275	247	2442	1480	39	64	432	128	45422	432	43	187
ex63	18	5512	22	5251	210	47144	5353	40	630	5311	87	17402	5311	27	632
ex65	18	10731	24	8936	793	285886	9036	43	1811	8885	130	231777	8887	35	1812
Average (Normalized)				1.01	6.75	129.88	1.00	1.00	1.00	1.00	2.81	86.54	1.00	1.00	1.00

Benchmark cases are selected from *IWLS 2022 Contest*

Script to produce local minima (start point): *ABC s t r a s h* $\rightarrow$ *CULS resyn2 (rs)* [3]

Until no further optimization

GTR VS &transduction:

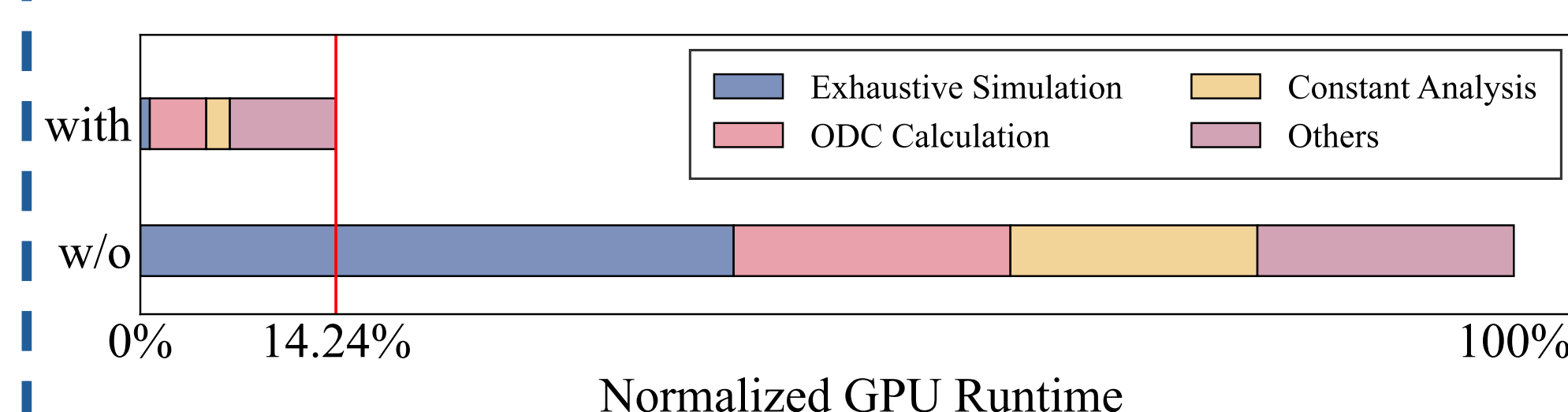
- Similar / better size
- Better depth

- 130x** speedup

GTR + CULS VS &transtoch:

- Better depth
- 86.54x** speedup

Effectiveness of solutions



[1] Y. Miyasaka, "Transduction Method for AIG Minimization," in 2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC), 2024, pp. 398–403

[2] S. Muroga, Y. Kambayashi, H. Lai, and J. Culliney, "The transduction method-design of logic networks based on permissible functions," IEEE Transactions on Computers, vol. 38, no. 10, pp. 1404–1424, 1989.

[3] T. Liu and E. F. Y. Young, "Rethinking AIG Resynthesis in Parallel," 2023 60th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 2023, pp. 1–6, doi: 10.1109/DAC56929.2023.10247961.