Post-Mapping Resubstitution For Area-Oriented Optimization

ASP-DAC 2025

Authors:

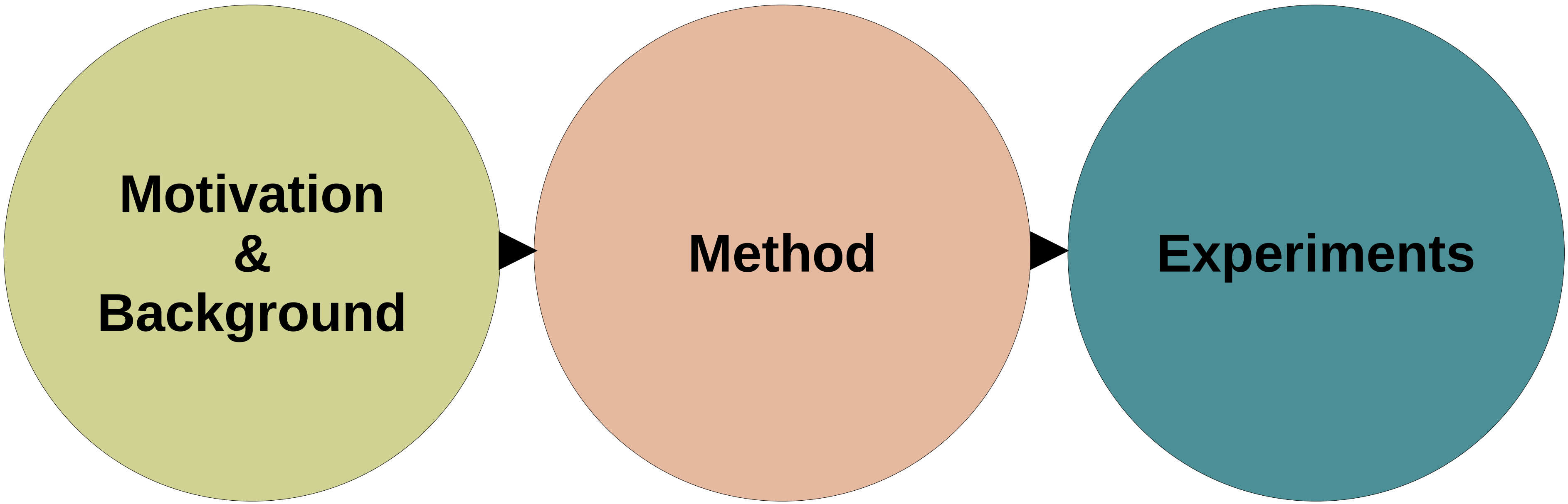
Andrea Costamagna (speaker)

Alessandro Tempia Calvino

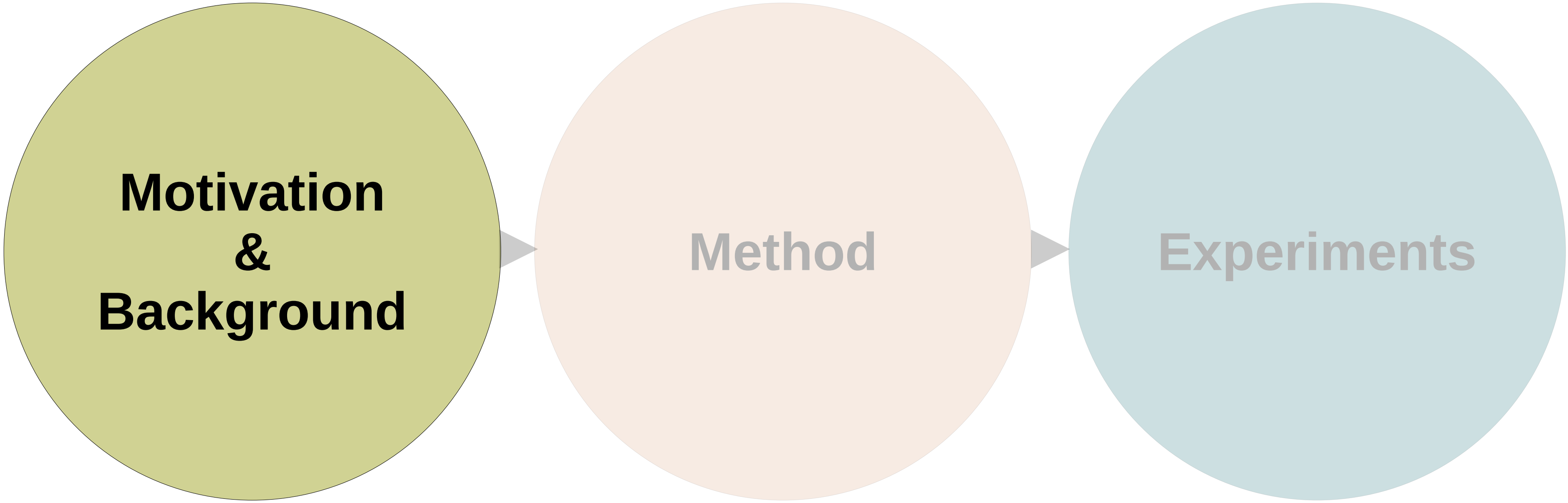
Alan Mishchenko

Giovanni De Micheli

Outline



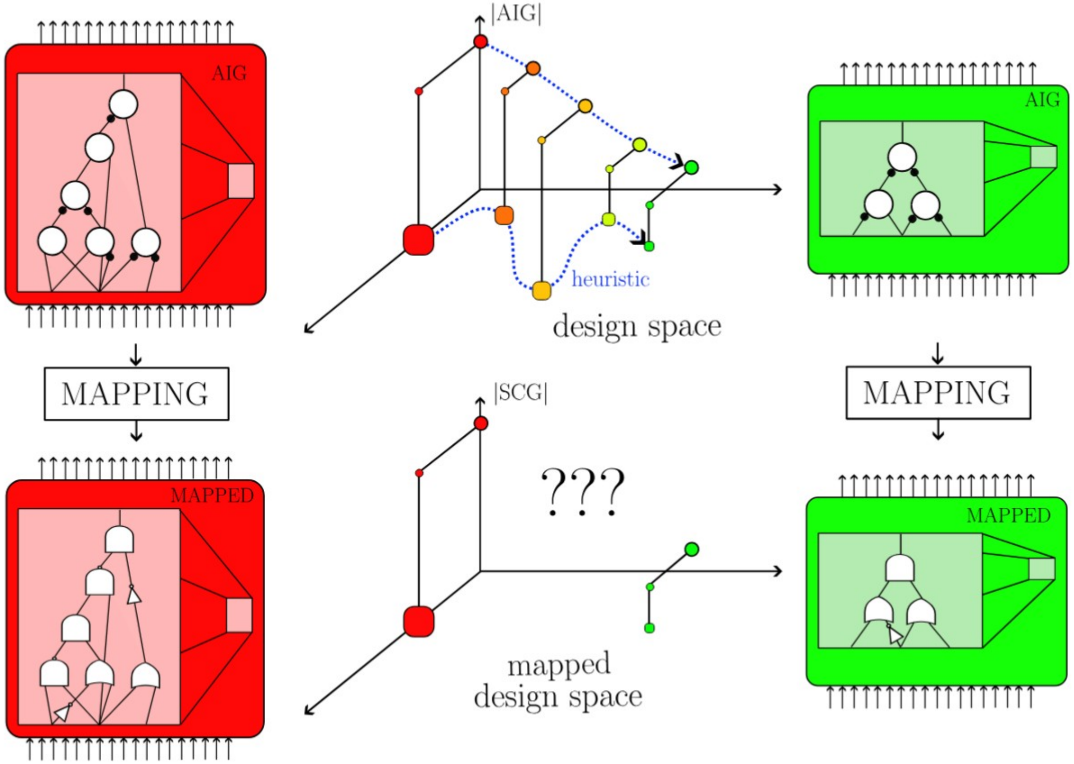
Motivation And Background



Logic Synthesis And Optimization

- Traditional approach:
  - Optimize AIG
  - Map to technology
- Assumption:

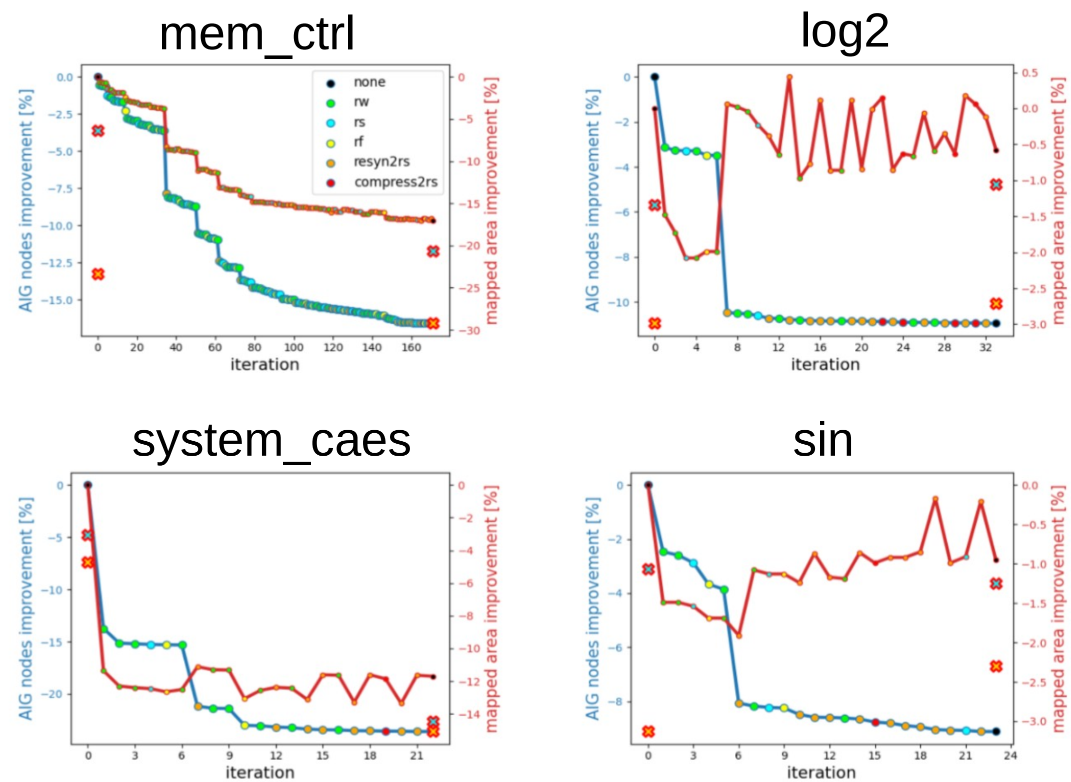
  Minimizing the AIG size correlates with reduced area after mapping



Pitfalls Of The Traditional Approach

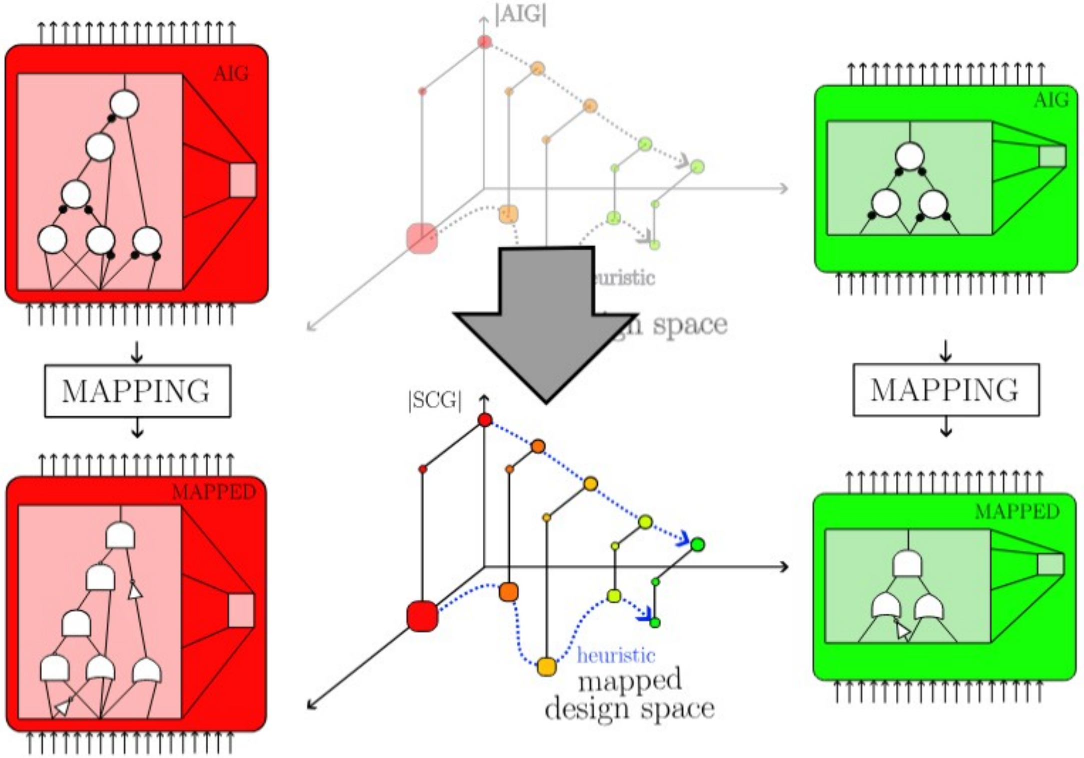
- High-effort traditional optimization
  - Iterative AIG optimization
  - Map each AIG
- Empirical evidence:
  - Anti-correlation
  - Need for

    technology awareness



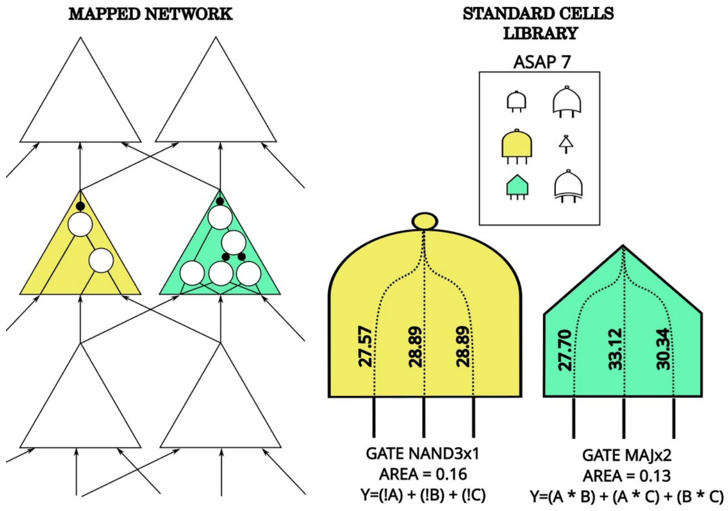
Moving Optimization To The Mapped Space

Idea:
Can we optimize a circuit directly in the mapped space?



Technology Mapping [1]

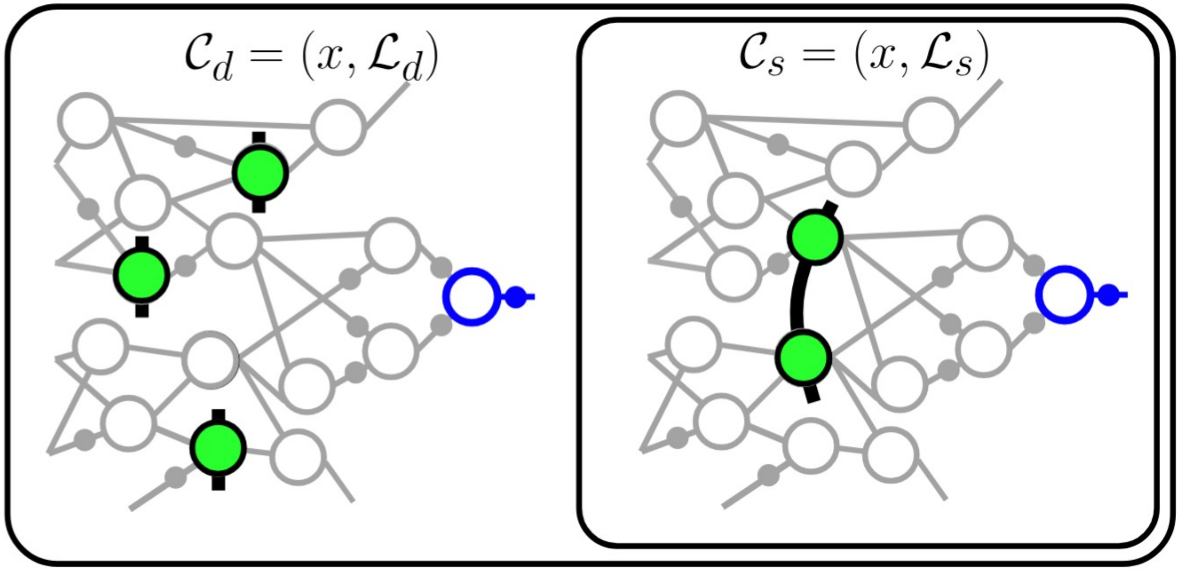
- Technology mapping:
  - Perform covering using structural cuts
  - Based on *supergates*
  - Uses load-independent model
- **Remark**
  covering is almost entirely based on ***structural*** *information*



[1] Chatterjee et al. Reducing structural bias in technology mapping, *International Conference on Computer-Aided Design, ICCAD 2005*

Dependency Cuts [2]

- *Dependency cuts*:
  - Based on functional information
  - Representation-independent concept
  - More general than structural cuts



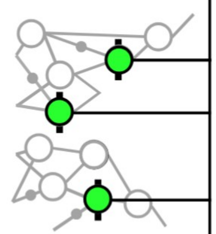
[2] Hong et al., “Functional Dependency For Verification Reduction”, *Computer Aided Verification, CAV 2004*

Outline

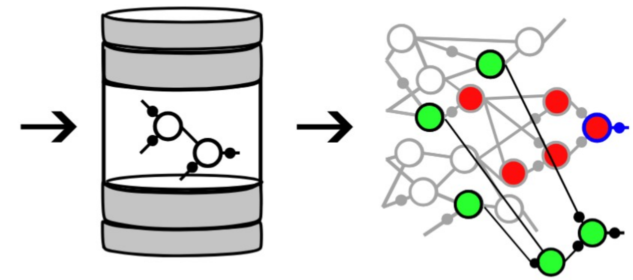
Motivation & Background → Method → Experiments

The Method In A Nutshell

- Method
  - Find a dependency cut
  - Extract the functionality
  - Look-up a database of supergates
- **Remarks**:
  - Non-structural cuts
  - Non-local optimization
  - In the mapped space



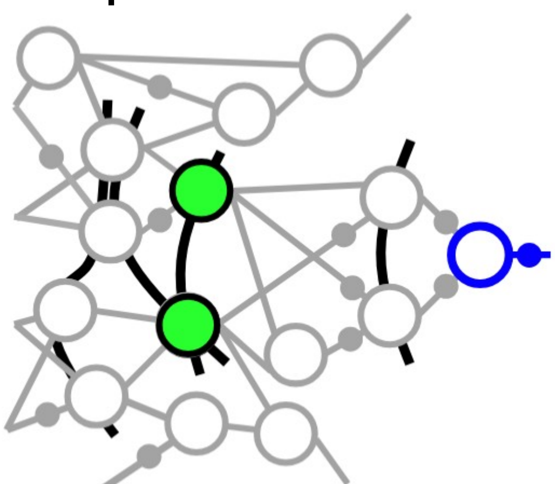
x_2	x_1	x_0	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	*
1	0	0	1
1	0	1	*
1	1	0	*
1	1	1	1



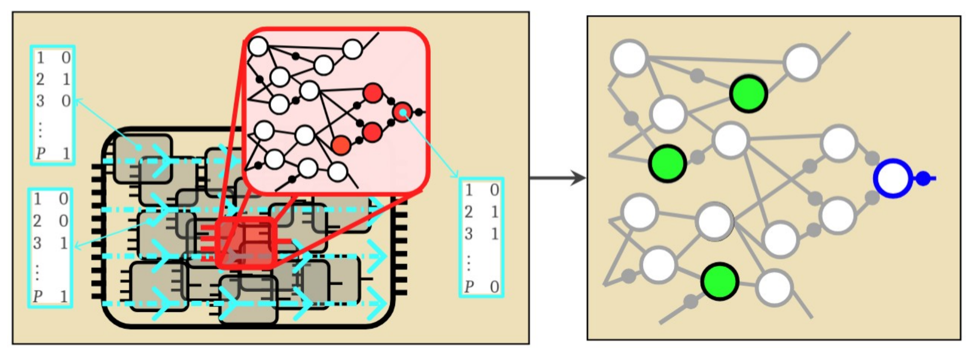
Find Dependency Cuts

The nodes in a cut contain enough information to resynthesize the target node

Structural cuts:
simple enumeration



Non-structural cuts:
problem mapped to a set-covering problem



- Window-based: exhaustive simulation
- Simulation-based: *signatures*

Build A Database

Zero-cost inverters (AIG, XAIG, MIG):

- NPN canonization
- Exact synthesis

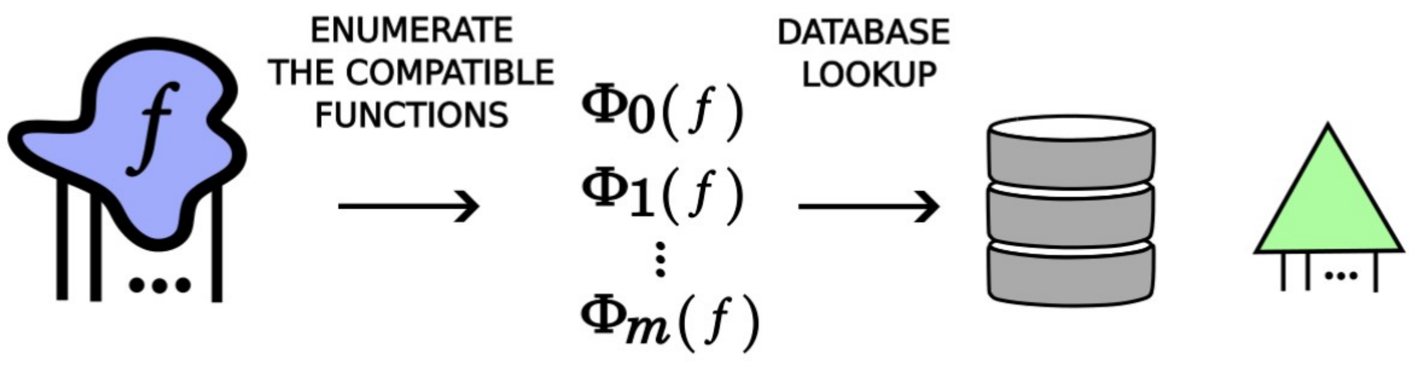
Standard-Cell Libraries:

- P canonization
- Map compact AIG

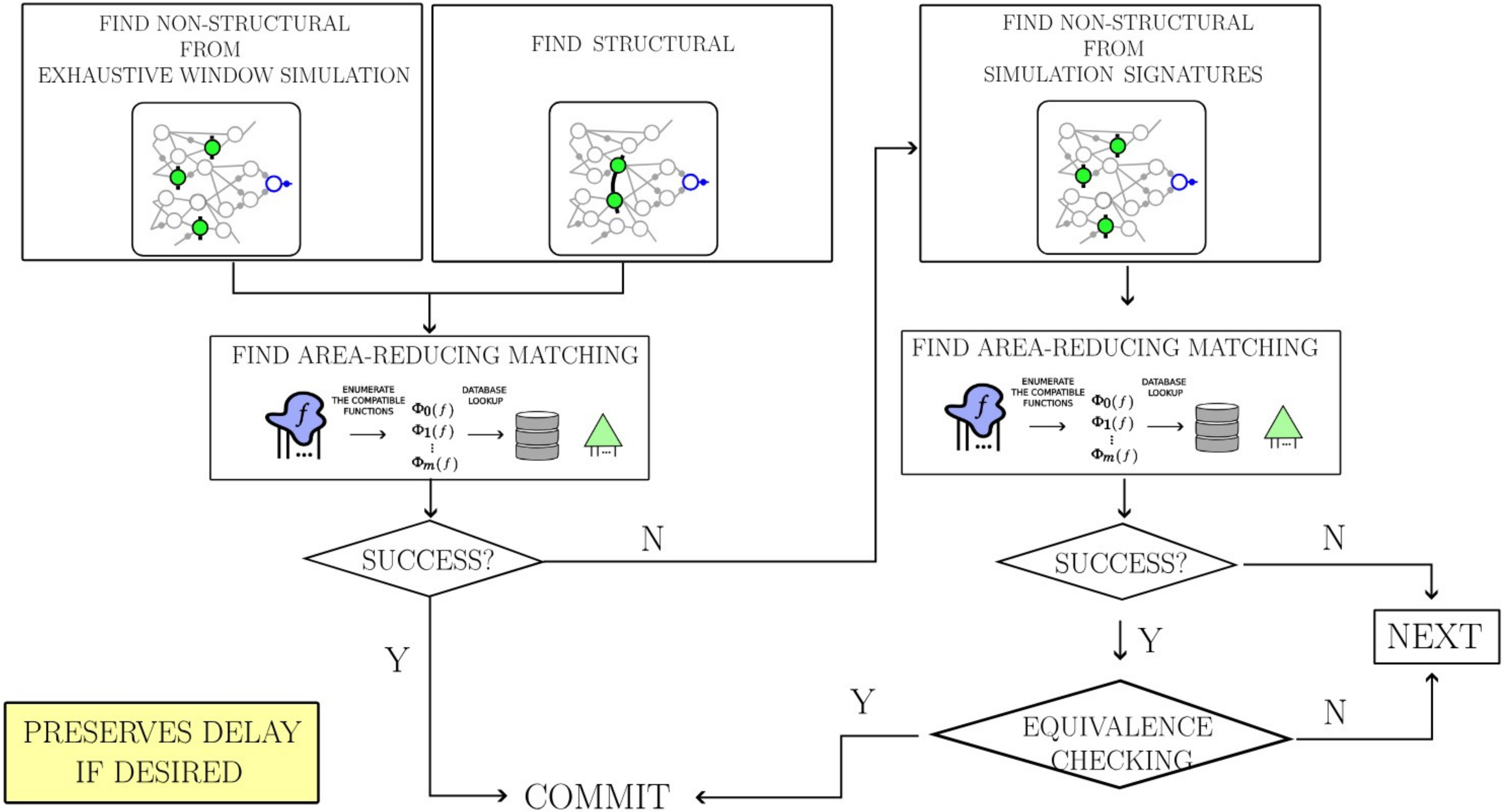
- **Goal:**
  - Optimize for area
  - Preserve delay constraints
- Not necessary to use optimum-area supergates

Boolean Matching With Don’t Cares

- Don’t cares: input patterns not appearing at the cut
- On average 2 don’t cares $\rightarrow$ 4 functions
- Method:
  - Enumerate all the functions
  - Choose the one resulting in smallest area

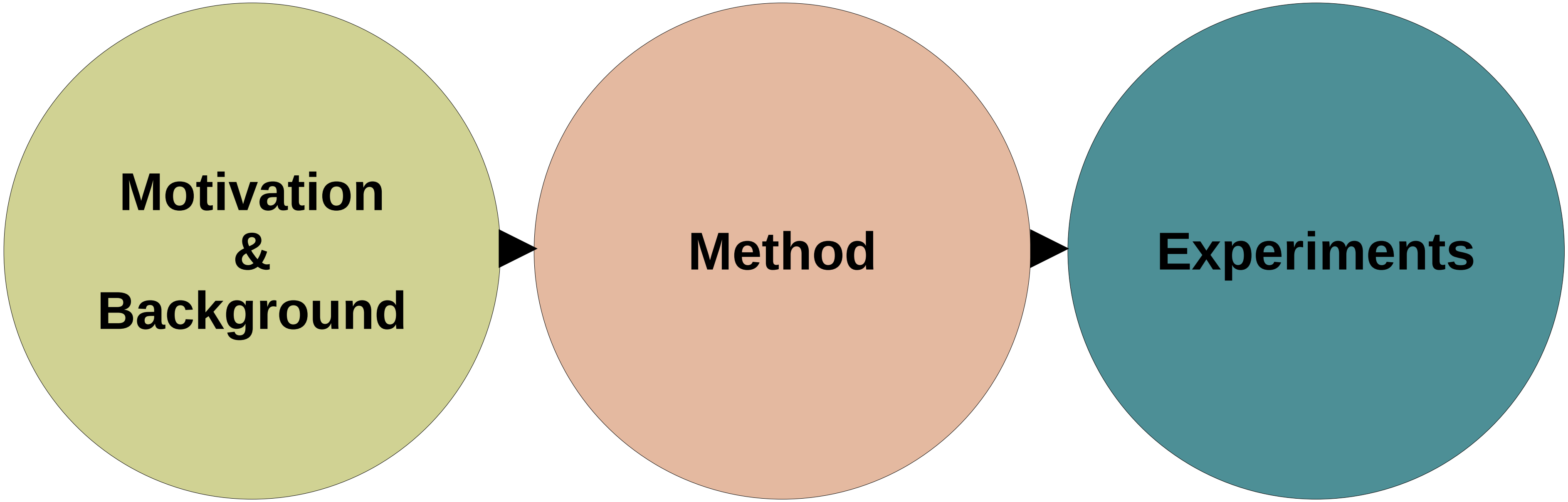


Post-Mapping Area-Oriented Optimization



Motivation & Background
Method
Experiments

Outline



Bridging Rewriting And Resubstitution

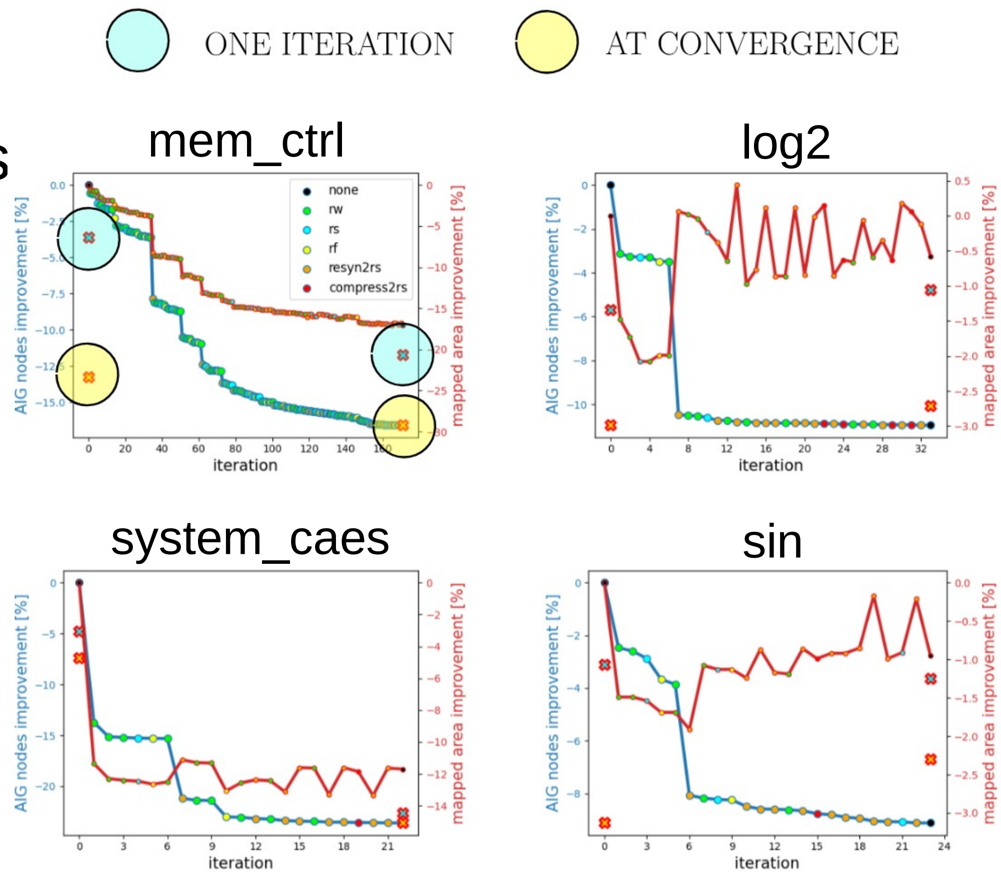
- **Question**:
  Can non-structural dependency cuts improve rewriting?

- Experiment:
  - *Rewrite*: 9 structural cuts per node
  - *Rewrub*: 8 structural + 1 non-structural cut per node
  - Each applied twice on non-optimized networks

library	$\delta A_{rewrite}[\%]$	$\delta A_{rewrub}[\%]$	$\delta d_{rewrite}[\%]$	$\delta d_{rewrub}[\%]$	$t_{rewrite}[s]$	$t_{rewrub}[s]$
AIG	-11.98	$\mathbf{-14.22}$	-4.90	-4.04	0.65	0.80
XAIG	-21.03	$\mathbf{-22.76}$	-8.30	-7.60	0.58	0.73
MIG	-20.02	$\mathbf{-22.63}$	-15.90	-12.71	0.74	0.91

Optimizing Networks Mapped From Different AIGs

- Optimize networks mapped from
  - Unoptimized AIGs
  - Highly-optimized AIGs



- In some benchmarks
  - Better results from unoptimized AIGs
  - Recover area from anti-correlation

Optimizing Area With Delay Constraints

- 2.81% after area-oriented AIG optimization and mapping
- No delay increase
- Even better results for more realistic settings

Design	$A_i[\mu m^2]$	$\delta A_i^1[\%]$	$\delta A_i^\infty[\%]$	$A_e[\mu m^2]$	$\delta A_e^1[\%]$	$\delta A_e^\infty[\%]$	$D_e[ps]$	$\delta D_e^1[\%]$	$\delta D_e^\infty[\%]$	$t_e^1[s]$	$t_e^\infty[s]$
div	3914.60	−14.35	−23.28	1296.90	−7.81	−9.30	60248.23	0.00	0.01	7.09	38.40
sqrt	1372.25	−12.44	−16.30	1171.15	−3.16	−5.22	78957.63	0.00	−1.21	4.09	50.83
arbiter	557.84	−13.87	−41.31	557.84	−12.01	−51.81	999.95	0.00	−39.57	1.59	17.38
mem_ctrl	2547.32	−7.87	−18.47	2063.01	−5.86	−12.79	1649.46	−5.82	−13.00	18.41	183.16
aes_core	1198.55	−4.18	−5.80	1106.60	−1.07	−1.81	434.52	−2.29	−2.05	11.41	165.45
ethernet	4411.85	−2.99	−4.35	3123.35	−0.96	−3.68	588.34	0.00	0.00	54.60	214.07
iwls05_i2c	66.32	−8.56	−12.26	49.96	−1.46	−2.00	288.07	0.00	0.00	0.19	0.58
RISC	4145.01	−8.10	−8.10	3172.54	−1.03	−1.03	1304.74	0.00	0.00	350.90	350.90
sasc	40.46	−1.29	−1.83	31.72	−1.07	−1.32	191.00	0.00	0.00	0.16	0.48
simple_spi	54.77	−4.78	−6.50	41.58	−0.53	−1.39	287.00	0.00	0.00	0.17	1.06
spi	205.45	−4.16	−9.05	167.59	−1.52	−2.08	489.49	0.00	0.00	0.43	3.66
systemcaes	611.55	−4.10	−5.75	530.89	−3.25	−3.62	784.00	0.00	0.00	1.23	11.28
systemcdes	172.17	−8.11	−13.69	142.22	−1.21	−2.38	530.29	0.00	0.00	0.39	2.22
tv80	512.39	−7.43	−12.29	356.38	−0.70	−1.48	907.86	0.00	0.00	1.03	6.84
usb_funct	887.05	−4.16	−7.41	702.45	−0.70	−0.98	722.00	−2.65	−2.65	1.64	12.45
usb_phy	28.82	−6.45	−10.58	23.97	−0.83	−1.42	179.06	−7.79	−8.51	0.15	0.61
		−5.75%	**−10.21%**		**−1.21%**	**−2.81%**		**−0.48%**	**−1.81%**	17.62	47.15

Mapped Design Space Exploration

- Compare:
  - Iterative optimization
  - Map-unmap-optimize
- Trick allowing for further improvements

Design	$A[nm^2]$	$\delta A^{\infty}[\%]$	$\delta A^{5\times5}[\%]$	$t^{\infty}[s]$	$t_e^{5\times5}[s]$
bar	149.13	−0.04	−3.24	0.50	7.88
div	1302.07	−9.96	−15.54	91.97	75.89
sin	289.47	−0.24	−1.41	2.58	35.02
sqrt	1171.15	−3.99	−5.98	28.14	89.16
arbiter	557.84	−45.59	−55.49	11.74	19.71
cavlc	34.53	−0.96	−1.13	0.85	6.90
ctrl	5.90	−2.71	−3.90	0.29	4.92
i2c	59.23	−0.95	−1.08	0.62	6.09
mem_ctrl	2164.98	−13.96	−11.38	187.81	256.89
priority	27.66	−0.29	−2.75	0.30	5.34
		−4.23%	**−5.47%**		

Conclusions

- We proposed a post-mapping optimizer
- The method preserves delay
- The method combines:
  - Structural cuts
  - Non-structural cuts
- Enables aggressive post-mapping optimization

Thank You!