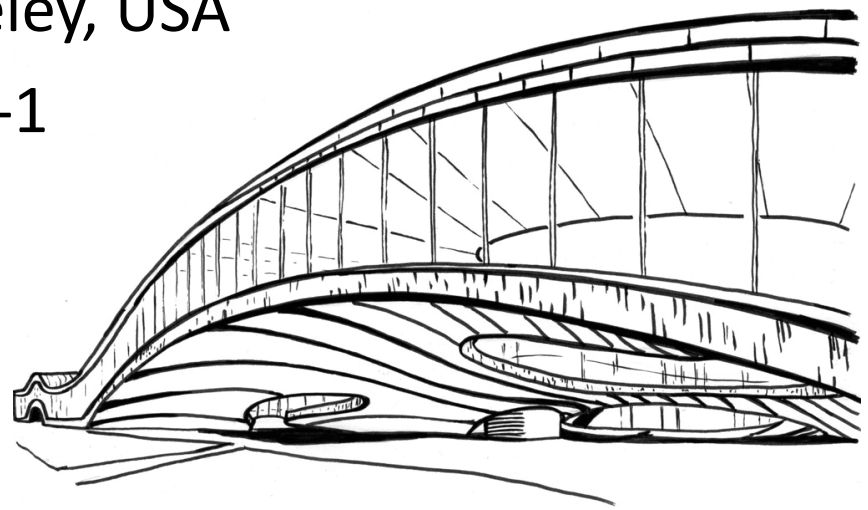
Boolean Rewriting Strikes Back: Reconvergence-Driven Windowing Meets Resynthesis

Heinz Riener[1], **Siang-Yun Lee**[1], Alan Mishchenko[2] and Giovanni De Micheli[1]

[1]EPFL, Switzerland and [2]UC Berkeley, USA

ASP-DAC 2022, Session 5D-1



0. (Classical) Boolean Rewriting	1. Reconvergence-Driven Windowing	3. Window Rewriting
	2. Heuristic Boolean Resynthesis	

Boolean Rewriting Strikes Back: Reconvergence-Driven Windowing Meets Resynthesis

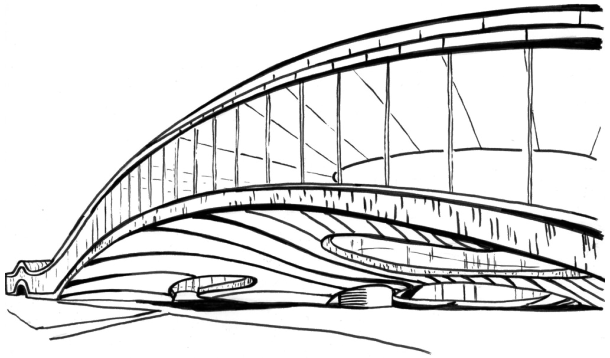
- Boolean rewriting: classical logic optimization algorithm
- Two bottlenecks to go beyond 4-cuts:
  - ~~Cut enumeration~~ → Reconvergence-driven windowing
  - ~~Database of optimum circuits~~ → Heuristic resynthesis

  } Window rewriting
- +3.2% better quality, 2.7x faster compared to ABC drw

[3] P. Bjesse and A. Borälv, “DAG-aware circuit compression for formal verification,” in *ICCAD 2004.*
[4] A. Mishchenko, S. Chatterjee, and R. K. Brayton, “DAG-aware AIG rewriting: A fresh look at combinational logic synthesis,” in *DAC 2006.*

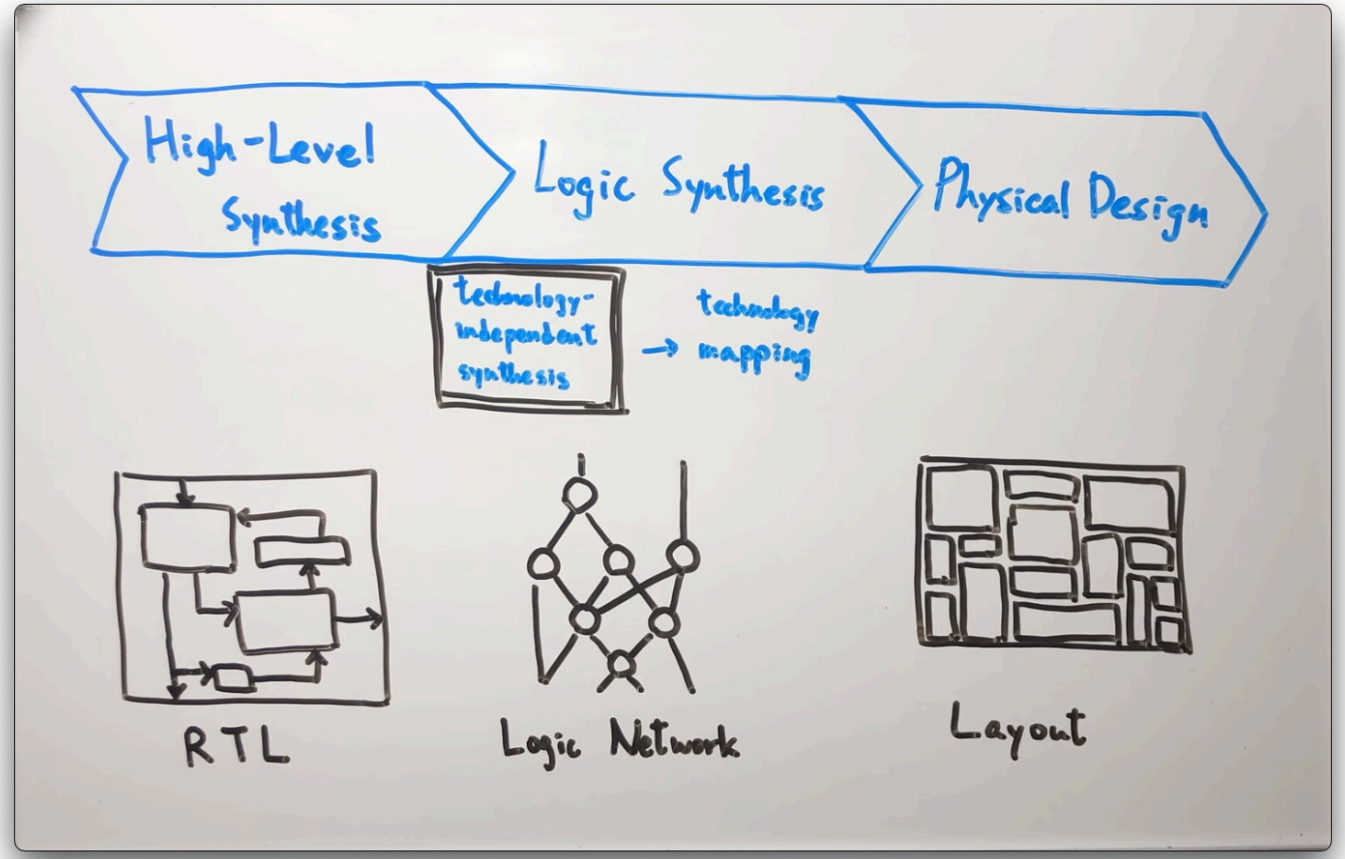
0. (Classical) Boolean Rewriting

Powerful classical algorithm, yet hard to go further



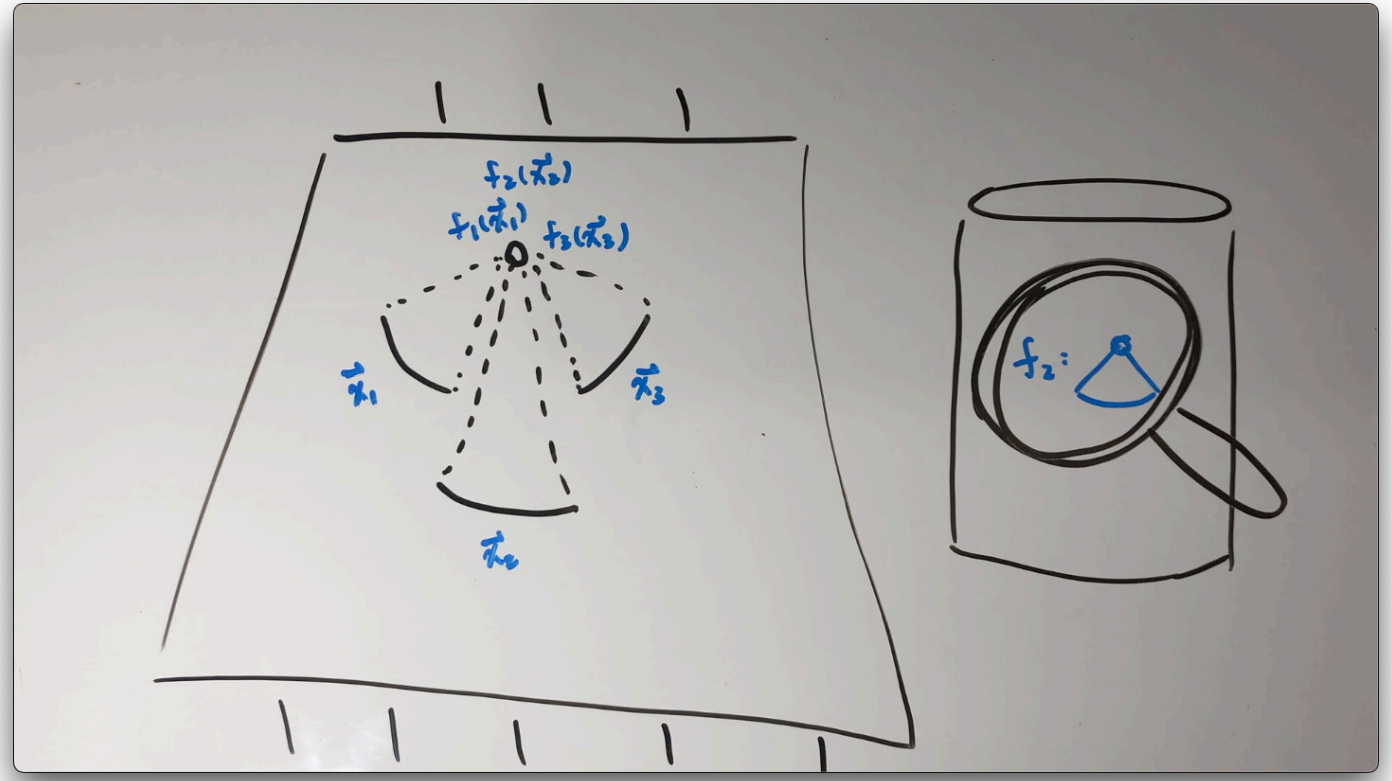
Technology-Independent Optimization

- Abstract data structure modeling digital circuits (e.g. AIGs, XAGs)
- Early step in logic synthesis
- Focus on area minimization (#nodes)



Cut-Based Rewriting: Idea

1. Choose a pivot
2. Enumerate cuts
3. Simulate
4. Find a better sub-circuit
5. Replace the sub-circuit
6. Choose another pivot (go back to 1.)



Cut-Based Rewriting: Bottlenecks

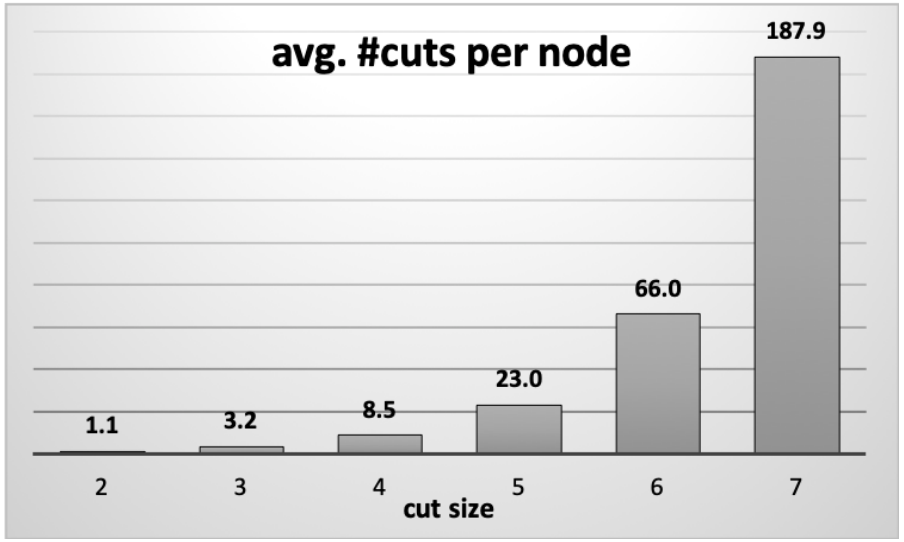
- Cut enumeration

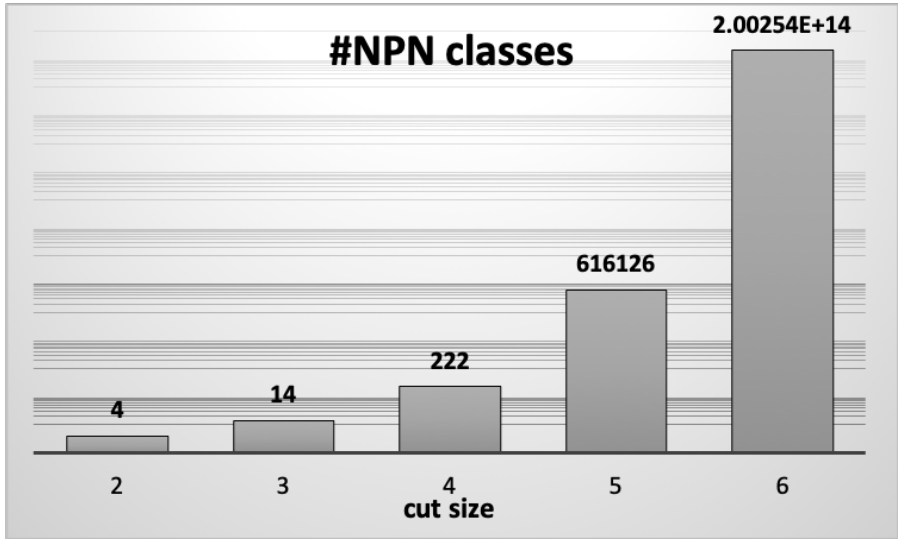
  #cuts ∝ exp(k) 🤯

k: cut size

- Database

  #functions ∝ exp(exp(k)) 🤯

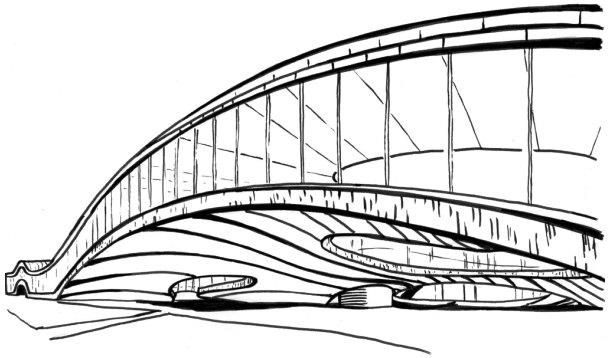




→ Hard to extend beyond 4-cuts 😵

1. Reconvergence-Driven Windowing

One window to rule them all



One “Good” Window Instead of Many Cuts

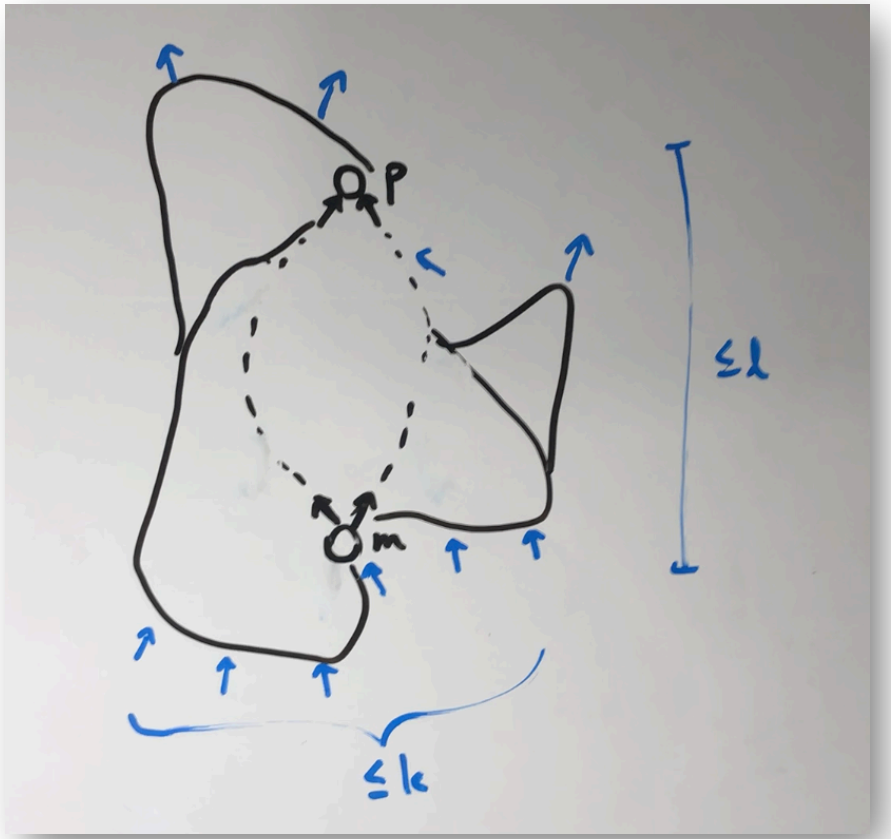
- Single-rooted cuts → multi-output windows
- A “good” window captures reconvergences
- Reconvergence is key to optimization

Reconvergence-Driven Windowing

Input: pivot p, cut size k, distance l

Output: a window around p with ≤ k inputs and reconvergence within l steps

1. Identify reconvergence
2. Collect inputs
3. Expand towards TFI
4. Expand towards TFO
5. Identify outputs



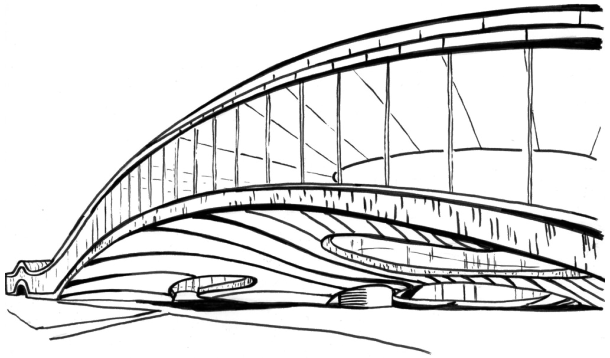
Quality of Windows

- Node containment: 98%
  → Most nodes are considered at least once in a window
- 4-cut containment: 41%
  → One 6-input window captures 41% of the pivot's 4-cuts

  (The other cuts may be contained in another window, so ≥ (⋙) 41% of the 4-cuts are contained in at least one 6-input window)

2. Heuristic Boolean Resynthesis

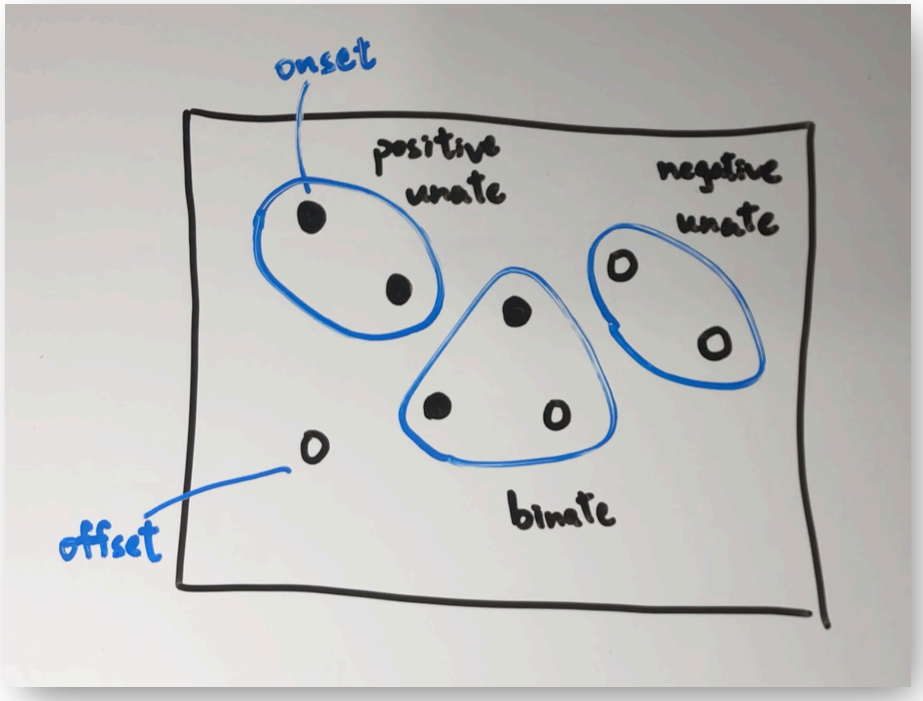
Getting rid of the exact database

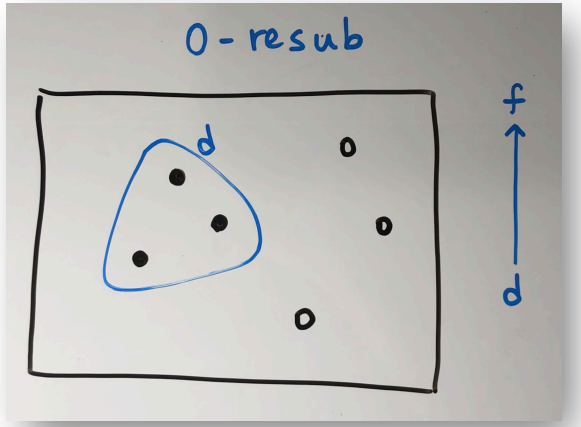


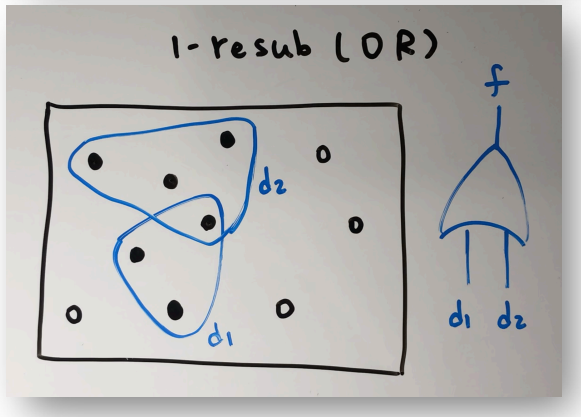
Resynthesize Node Functions On-the-fly

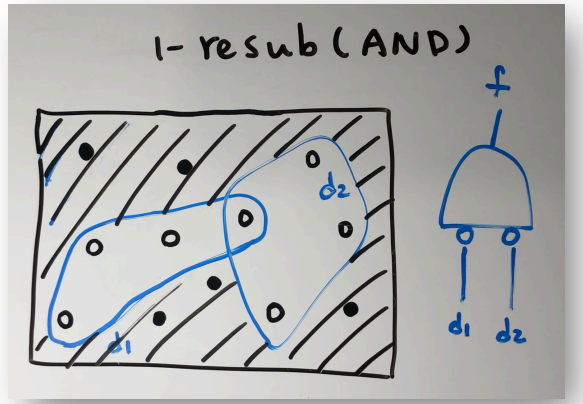
- Given a target function f and some divisor functions $g_1, \ldots, g_r$, find a dependency function (circuit) h, such that $f = h(g_1, \ldots, g_r)$
- Similar to resubstitution, but not limited to small dependency circuits
- Fast heuristic
- General, the functions' input size is not limited

“Unate” Divisors

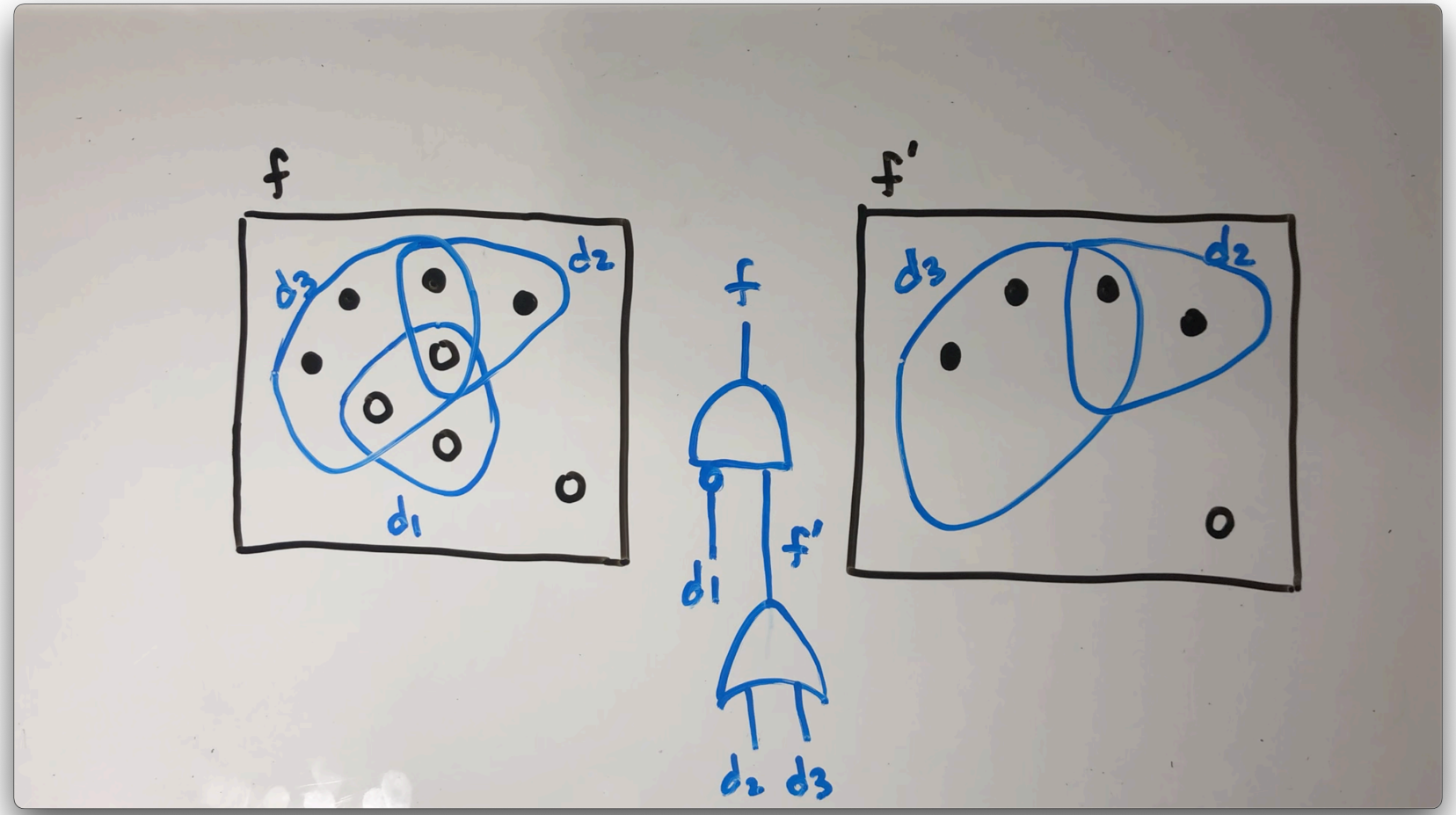








Recursive Synthesis

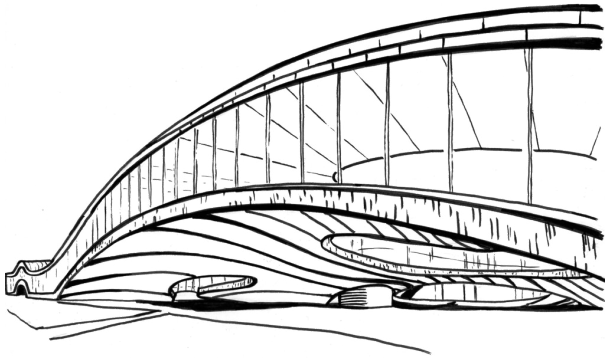


Quality of Resynthesis

#vars		Exact database	Heuristic resynthesis	
		#gates	Solved	#gates
3	AIG	794 ANDs	254/256 99%	890 ANDs +0.35 gates/function
	XAG	384 ANDs + 206 XORs	254/256	528 ANDs + 142 XORs
4	AIG	365276 ANDs	54622/65536 84%	499308 ANDs +2.05 gates/function
	XAG	178536 ANDs + 98940 XORs	54622/65536	351592 ANDs + 60332 XORs

3. Window Rewriting

A new hope



Windowing + Resynthesis

- Construct **one** window per pivot node
- Optimize the window by resynthesizing **every** node in it

Comparison with 4-Cut Rewriting

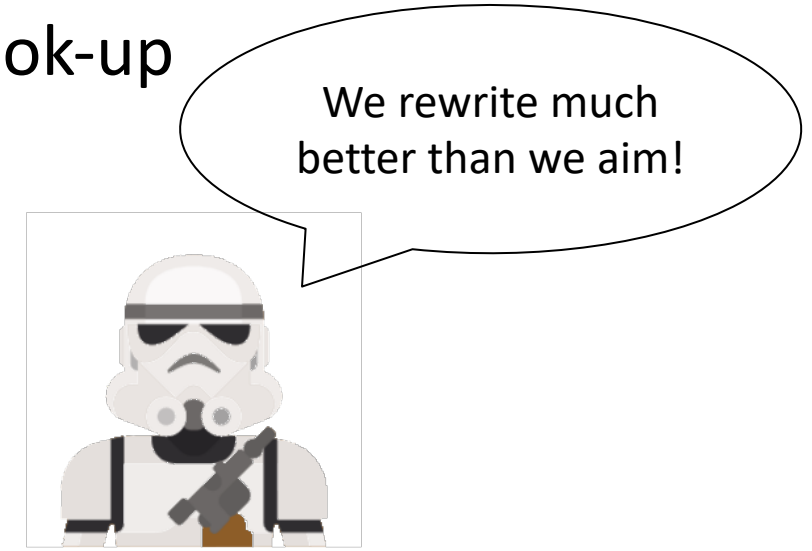
	ABC drw		Window rewriting	
	First iteration	Until convergence	First iteration	Until convergence
Size reduction	5.44%	5.61%	8.86%	9.16%
Runtime (s)	6.70	28.44	10.69	32.84

+3.2% better quality, 2.7x faster

Implementation available in `mockturtle`: https://github.com/lsils/mockturtle

Conclusions & Future Work

- One 6-input **window** instead of many 4-cuts
- Capture **reconvergences**
- Heuristic **resynthesis** instead of exact database look-up
- Local computation; potential for parallelization
- Resynthesis + other algorithms [15]
- Window rewriting + other resynthesis engines [14]



[14] S.-Y. Lee, H. Riener, and G. D. Micheli, “Logic resynthesis of majority-based circuits by top-down decomposition,” in *DDECS 2021.*
[15] S.-Y. Lee, H. Riener, A. Mishchenko, R. K. Brayton, and G. De Micheli, “A simulation-guided paradigm for logic synthesis and verification,” *TCAD 2021*.

Boolean Rewriting Strikes Back: Reconvergence-Driven Windowing Meets Resynthesis

Heinz Riener[1], **Siang-Yun Lee**[1], Alan Mishchenko[2] and Giovanni De Micheli[1]

[1]EPFL, Switzerland and [2]UC Berkeley, USA

ASP-DAC 2022, Session 5D-1