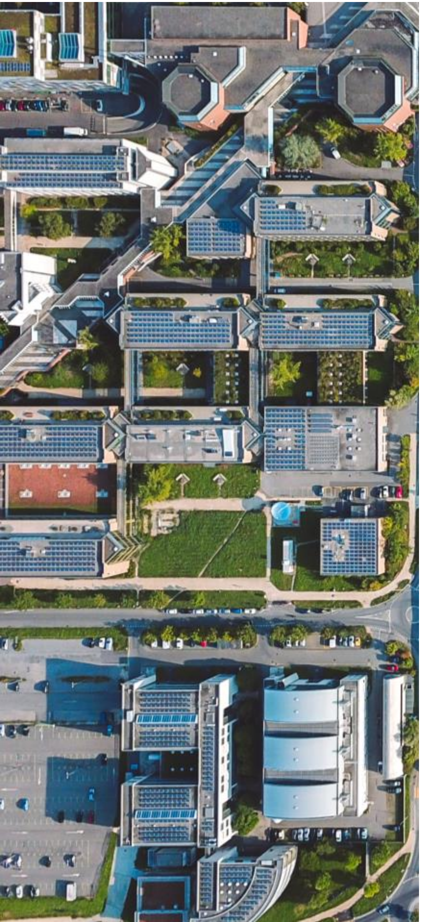


In Medio Stat Virtus: Combining Boolean and Pattern Matching

Gianluca Radi

Alessandro Tempia Calvino

Giovanni De Micheli

ASP-DAC 2024

Integrated Systems Laboratory, EPFL, Lausanne, Switzerland

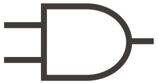
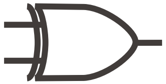
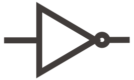
alessandro.tempiacalvino@epfl.ch

24th Jan 2024

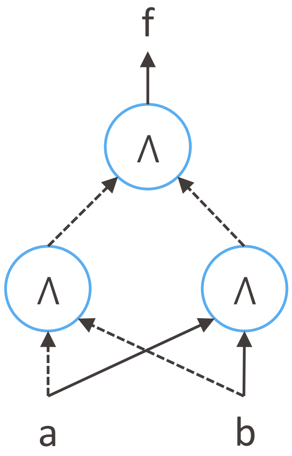
G. Radi, A. Tempia Calvino, G. De Micheli

Technology mapping

Problem: given a library of cells, implement a Boolean circuit using instances of library cells

Cells		Cost
	AND2	4
	XOR2	12
	INV	1

Cell library



Boolean circuit

Technology mapping formulation

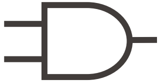
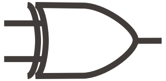
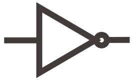
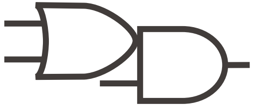
- Optimum-cost mapping is an intractable problem
- Formulation:
  - Multi-level graph obtained by (logic) synthesis
  - Mapping by a series of **local substitutions** on a graph
    - Transform into an interconnection of library cells
- Minimize:
  - Delay
  - Power
  - Area
  - …

Technology mapping problems

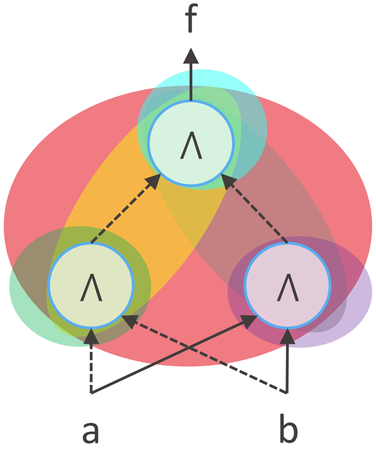
- Decomposition into primitives
  - And-inverter graph
- **Matching**:
  - A cell matches a sub-network when the terminal behavior is the same
- **Covering**:
  - Partition into sub-networks that can be replaced by library cells

G. Radi, A. Tempia Calvino, G. De Micheli

Matching

Cells	Cost
 AND2	4
 XOR2	12
 INV	1
 OA21	8

Cell library

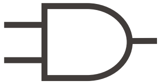
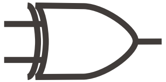
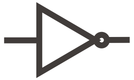
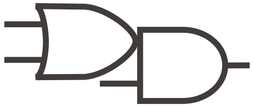


Boolean circuit
with a simple structure

M1: {AND2(INV(x),INV(y))}
M2: {AND2(x, y)}
M3: {AND2(INV(x),INV(y)}
M4: {OA21(x,y,INV(z)}
M5: {OA21(INV(x),INV(y),INV(z)}
M6: {XOR2(x,y)}

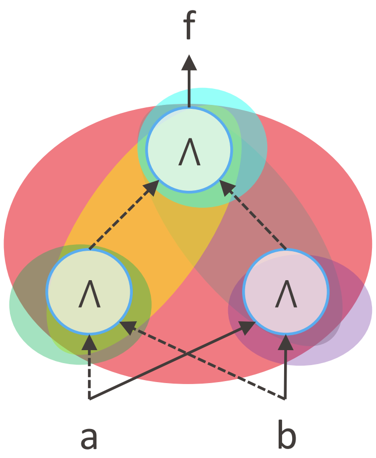
Matching

Covering

Cells		Cost
	AND2	4
	XOR2	12
	INV	1
	OA21	8

Cell library

M1: {AND2(INV(x),INV(y)))}
M2: {AND2(x, y)}
M3: {AND2(INV(x),INV(y))}
M4: {OA21(x,y,INV(z))}
M5: {OA21(INV(x),INV(y),INV(z))}
M6: {XOR2(x,y)}



Cover	Cost
M1,M5	17
M1,M2,M3	16
M2,M4	13
M6	**12**

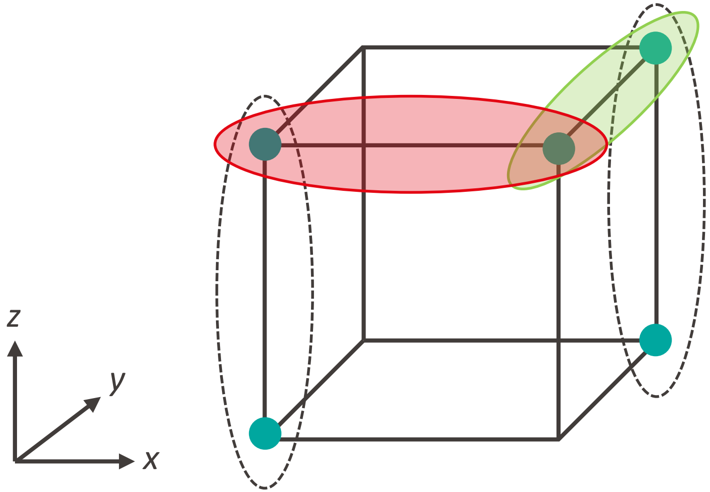
Covering

Matching techniques

- **Structural matching**
  - Models functions by patterns (simple structure, e.g., AND + INV)
  - Relies on pattern matching techniques
  - Scales well for tree patterns (beyond 9-inputs cells)
- **Boolean matching**
  - Use Boolean methods
  - Solves a tautology problem
    - Truth tables, BDDs, (canonical forms)…
  - More powerful
  - Scales up to 6-input cells

Structural vs Boolean example

- Structural vs Boolean:
  - $f = xy + \bar{x}\bar{y} + \boxed{\bar{y}z}$
  - $g = xy + \bar{x}\bar{y} + \boxed{xz}$
- Function equality is a tautology
  - Boolean match exists
- Patterns are different
  - Structural match may not exist



State of the art

- Rule-based methods were used for technology mapping originally in the 80's
  - LSS [1], Socrates [2]
- Patterns can be automatically generated using Boolean decomposition
- Some technology mappers are still based on pattern matching
  - ABC *amap*
- Boolean matching was introduced in the 90's [3]
- Most of the modern open-source mappers work using Boolean matching
  - ABC *map* and *&nf*
  - Mockturtle *map*

[1] J. A. Darringer, D. Brand, J. V. Gerbi, W. H. Joyner and L. Trevillyan, "LSS: A system for production logic synthesis," in IBM Journal of Research and Development, 1984
[2] D. Gregory, A. de Geus, K. Bartlett and G. Hachtel, "SOCRATES: A System for Automatically Synthesizing and Optimizing Combinational Logic," DAC, 1986
[3] F. Mailhot and G. De Micheli, "Technology mapping using Boolean matching and don't care sets," EDAC, 1990

NPN classification

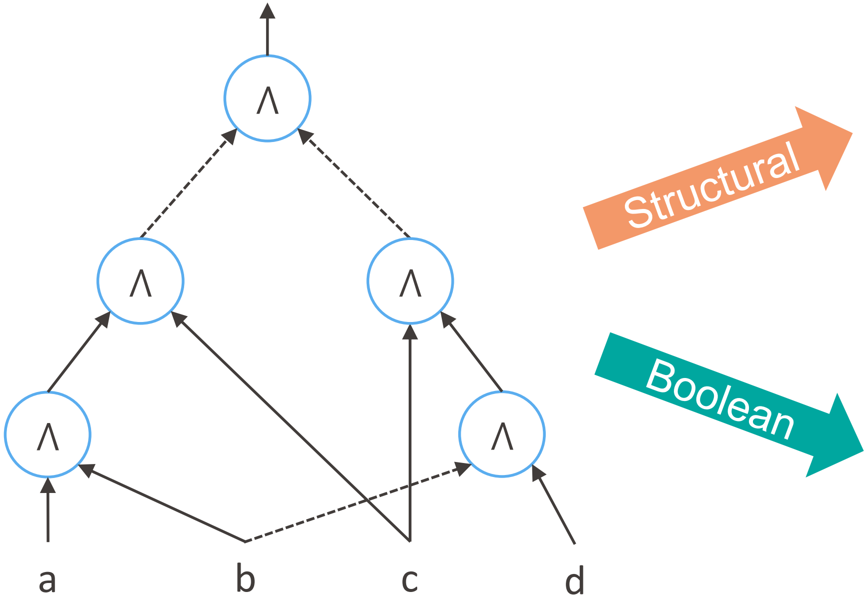
- **NPN** classification of logic functions
- NPN-equivalence
  - Permutation operator ***P***
  - Complementation operator $\boldsymbol{N_i}$ and $\boldsymbol{N_o}$
  - $f(x) = \boldsymbol{N_o}\, g(\boldsymbol{P}\ \boldsymbol{N_i}\ x)$ is a tautology
- Example:
  - $f = ab + cd$
  - $g = \bar{a}c + b\bar{d}$
  - NPN-equivalent → they can be implemented by the same cell by adding input inverters and re-ordering the inputs

Analyzing structural vs Boolean matching

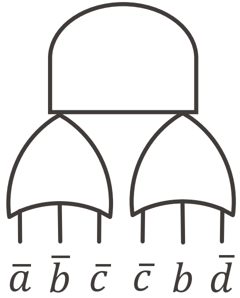
Metric	Structural	Boolean
Algorithm	Graph isomorphism	Tautology checking
Supported cells	Up to 9 (or more) inputs	Up to 6 inputs
Complexity limitations	Number of patterns	NPN configurations ($\leq n!\,2^n$)
Matching for trees	Fast	Match over NPN
Matching for DAGs	Complex → find reconvergences	Match over NPN
Redundancies	Remain	Removed

Matching on redundancies

$$f = (\bar{a} \vee \bar{b} \vee \bar{c}) \wedge (\bar{c} \vee b \vee \bar{d})$$



Match with redundancy → OA33 cell



Support of 4 variables → no match

Motivations and contribution

- Both structural and Boolean matching have **pros & cons**
  - Structural is very fast for tree-decomposable functions
  - Structural scales better to large functions (trees)
  - Boolean is generally better up to 6-input cells
  - Structural/functional matches can differ

- We propose *hybrid matching "In medio Stat Virtus"*
  - Combine the strengths of structural and Boolean matching
  - Better mapping quality
    - Support large cells
    - Speed-up mapping

Hybrid matching

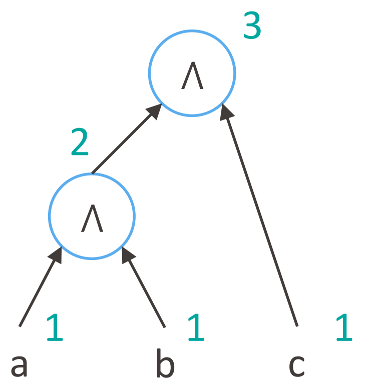
Hybrid matching: pattern generation

- Build a database of patterns
- Used for cells that are **decomposable into a tree** of ANDs and INVs
  - Fully-DSD (*disjoint support decomposable*) functions
- Initial pattern is computed using DSD
  - Top: $f(X) = x_i \odot g(X¥\{x_i\})$
  - Bottom: $f(X) = h(x_i \odot x_j, g(X¥\{x_i, x_j\}))$
- Patterns are not unique
  - $a \wedge (b \wedge c) = (a \wedge b) \wedge c$
- Additional patterns are found using associative properties
  - Enumerate tree rotations

G. Radi, A. Tempia Calvino, G. De Micheli

Hybrid matching: structural matching

- Assign a unique index to each pattern node
  - Defined from the index of the fanin nodes and their polarity
  - Canonicalized with input and output negation
- Graph isomorphism reduces to an index check
  - Similar to structural hashing
- If two graphs have the same index each sub-graph as matching indexes



Hybrid matching: Boolean matching

- Build a database of cells
- Compute all the NPN-configurations of the cells
  - Functions in the NPN class of the cell function
  - For cells with less than 7 inputs
- Matching reduces to a function look-up
  - Using truth tables

AND(x, y): 1000

AND(x, y)	: 1000
AND(INV(x), y)	: 0100
AND(x, INV(y))	: 0010
AND(INV(x), INV(y))	: 0001
INV(AND(x, y))	: 0111
…	

Mapping with hybrid matching

- Sub-graphs regions are defined by cuts
  - Cuts are enumerated using a **cut merging technique**
  - Compute the pattern index during merging → structural matching
  - Negligible run time
- Boolean matching
  - Compute the truth table of the cuts
  - Matching by accessing the NPN cell database
- Hybrid matching performs both computations
- The resulting matched cells is a union of the matches by the two methods
  - A limited number of matches are saved per node

Experiments

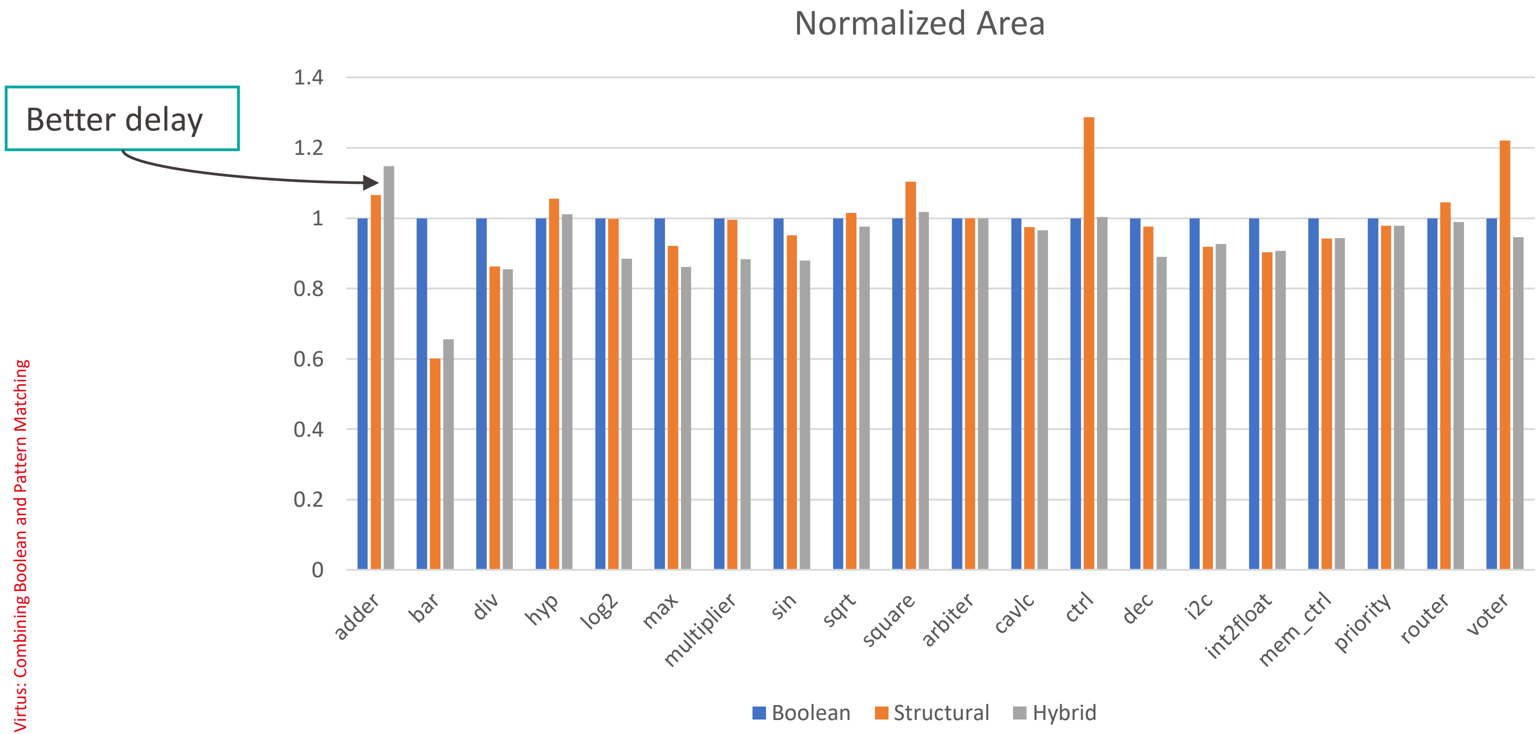
Setup

- Use the ASAP 7nm standard cell library [4]
- Compare
  - Boolean matching (baseline)
  - Structural matching (for tree patterns)
  - Hybrid matching
- Experimental results over the EPFL benchmarks
  - Delay-oriented mapping
  - Area-oriented mapping

[4] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "ASAP7: A 7-nm finFET predictive process design kit," Microelectronics Journal, 2016.

G. Radi, A. Tempia Calvino, G. De Micheli

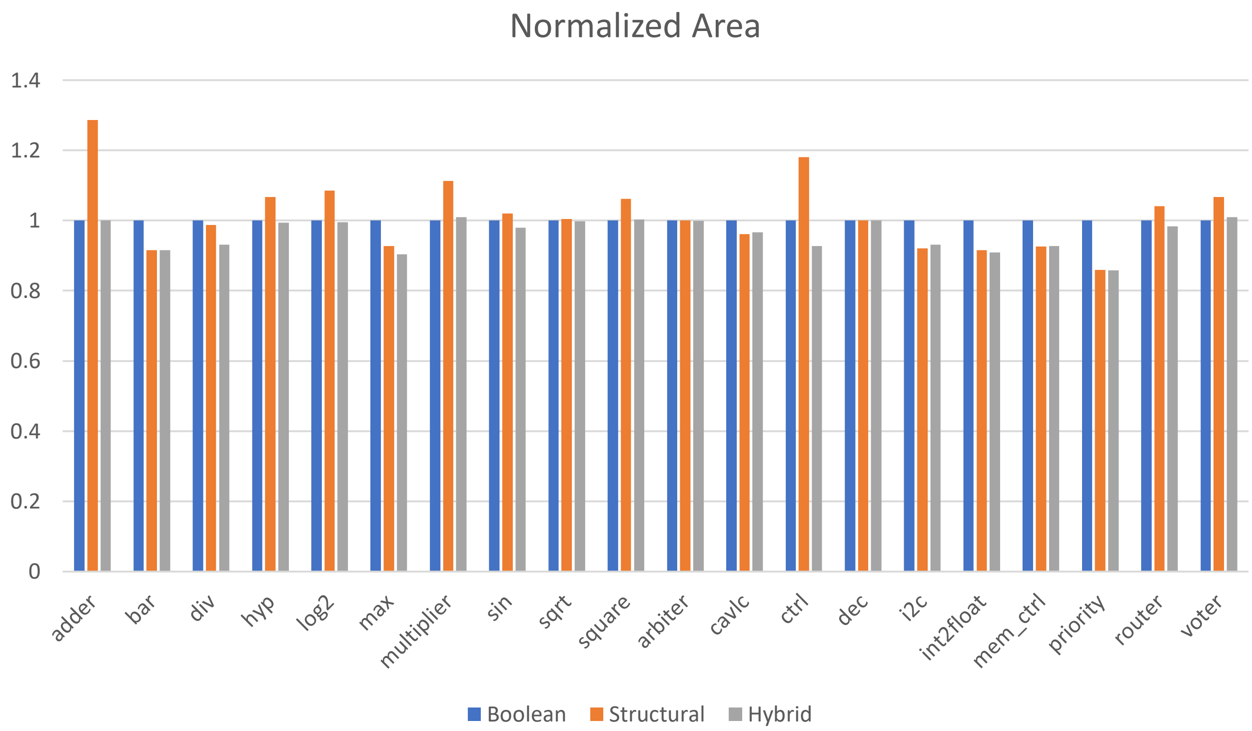
Delay-oriented mapping



Method	Area	Delay	Total Time
Structural	-0.9%	+1.6%	-77%
Hybrid	**-4.8%**	+0.2%	**-27%**

G. Radi, A. Tempia Calvino, G. De Micheli

Area-oriented mapping

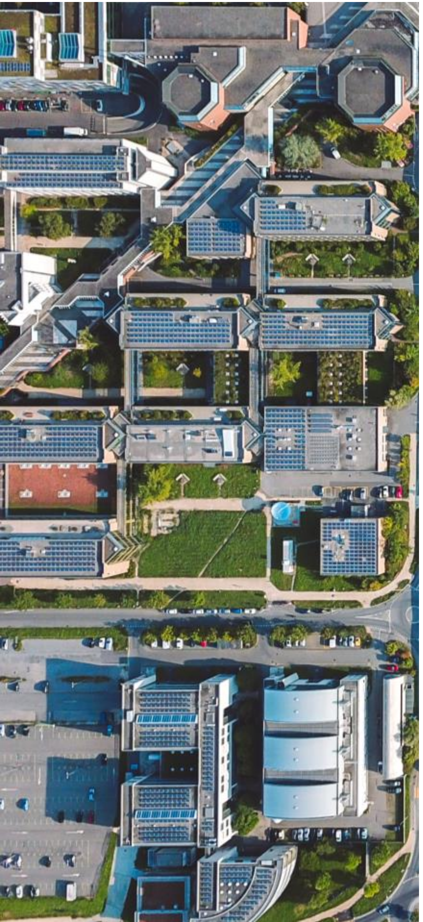


Method	Area	Delay	Total Time
Structural	+1.7%	+10%	-78%
Hybrid	**-3.8%**	**-1.48%**	**-25%**

Conclusion

Conclusion

- We presented *hybrid matching*
- Combine the **strengths** of *structural* and *Boolean matching*
  - Better quality
  - Support large cells
  - Better runtime
- Compared to Boolean matching
  - **-6% Area compared to ABC**
  - **Comparable delay**
  - **-25% runtime**
- Open-source implementation in the library *Mockturtle*
  https://github.com/lsils/mockturtle



In Medio Stat Virtus: Combining Boolean and Pattern Matching

Gianluca Radi

<u>**Alessandro Tempia Calvino**</u>

Giovanni De Micheli

ASP-DAC 2024

Integrated Systems Laboratory, EPFL, Lausanne, Switzerland

alessandro.tempiacalvino@epfl.ch

24th Jan 2024