Algebraic and Boolean Methods for SFQ Superconducting Circuits

Alessandro Tempia Calvino

Giovanni De Micheli

ASP-DAC 2024

Integrated Systems Laboratory, EPFL, Lausanne, Switzerland

alessandro.tempiacalvino@epfl.ch

24th Jan 2024

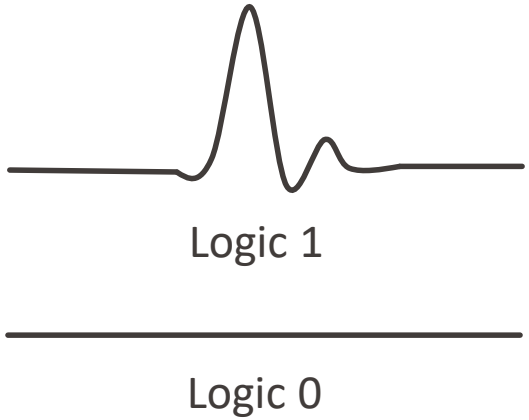
RSFQ motivations and challenges

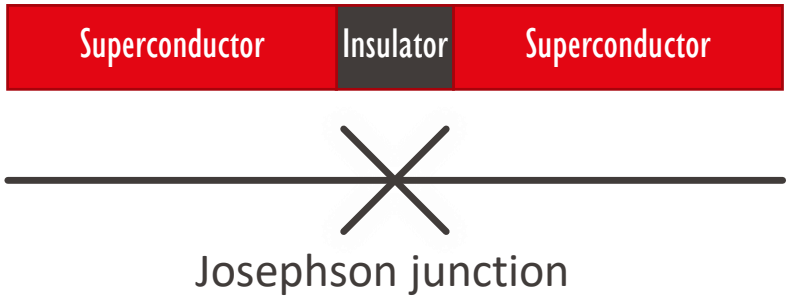
- Motivations
  - Worthwhile technology due to low power consumption [2]
- Applications
  - Large-scale computing
  - Routers and network switches
- Challenges
  - Expensive memory
  - Interconnects between room and low temperature
  - Fabrication technology
    - 6k JJs/mm^2

[2] D. S. Holmes, A. L. Ripple and M. A. Manheimer, "Energy-Efficient Superconducting Computing-Power Budgets and Requirements," in Trans. on Applied Superconductivity, 2013

RSFQ logic

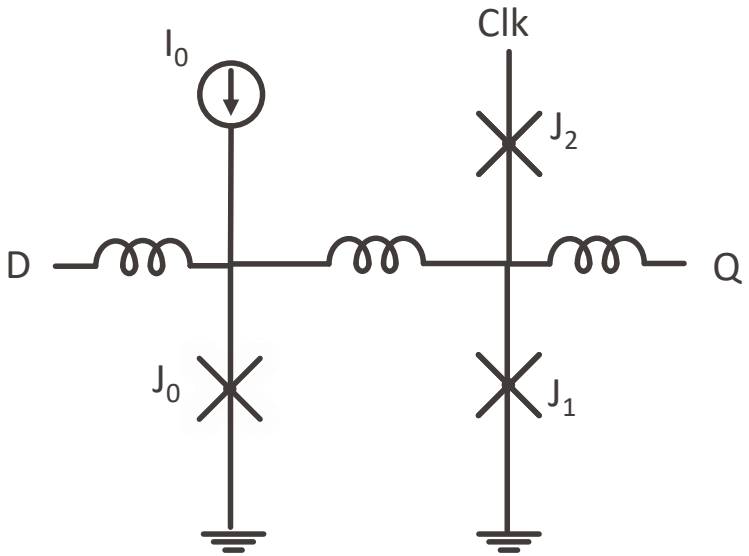
- Pulse-based logic
  - Distinguish ‘1’ from ‘0’
- Clocked logic
  - Output issued when clock tics
- Based on Josephson junctions and inductors





RSFQ basic elements

- Basic memory cells
  - **D-Flip-Flop (DFF)**
  - T-FF
  - NDRO-FF
- Asynchronous gates
  - Merger
  - **Splitter (1-to-2)**
- Synchronous gates (1 clock cycle)
  - **INV**
  - **AND**
  - **OR**
  - **XOR**
  - …



DFF

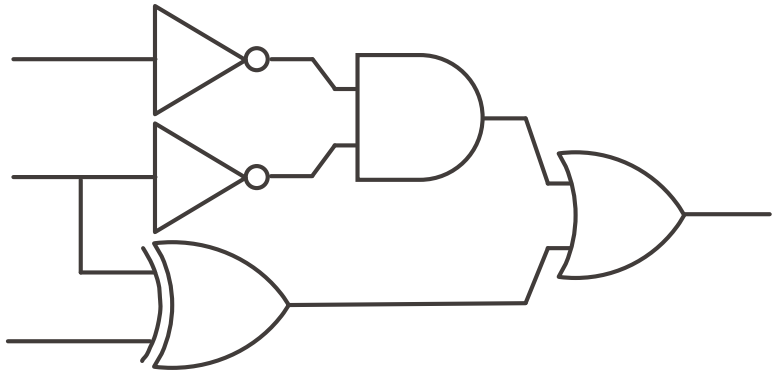
RSFQ technology constraints

1. **Fanout branching**
   - Need of splitting elements to drive multiple signals
   - Due to small currents
2. **Path balancing**
   - Need to synchronize data
   - Due to clocked logic

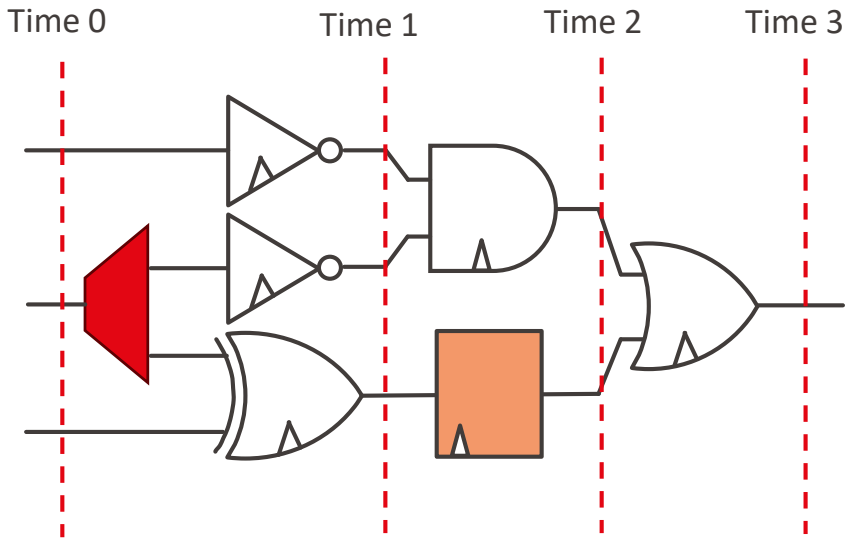
RSFQ technology constraints example

1. Fanout branching → adding splitters
2. Path balancing → adding FFs

Classic CMOS



SFQ



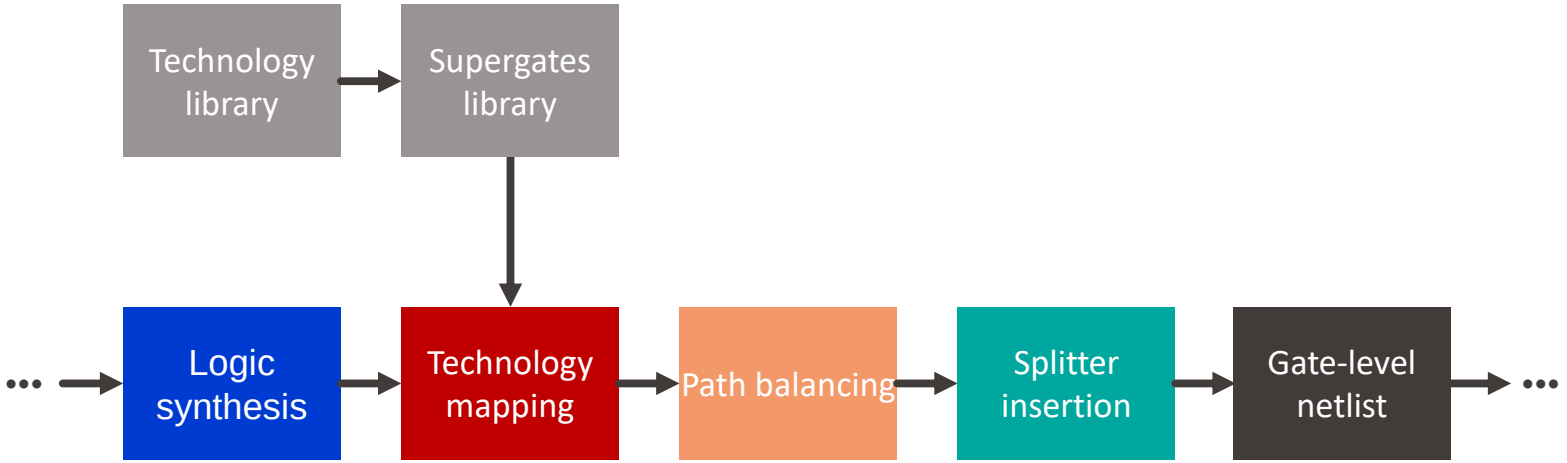
RSFQ synthesis challenges

- **Performance** highly depends on the number of **logic levels**
- A **large portion** of the area may be occupied by FF and splitters
  - Path balancing → reduce imbalances
  - Fanout branching → limit logic sharing
- Area is related to delay → path balancing

- **Contributions**
  - Design of an SFQ synthesis framework to optimize these metrics
  - Optimization over the *Xor-And Graph*
  - Revise the way of performing technology mapping

Synthesis for SFQ Circuits

Synthesis for SFQ circuits



Logic synthesis

- AND and XOR have similar delay and area
- Optimize over the **Xor-And Graph** (XAG)
  - More optimization opportunities
- Focus on **delay-oriented** optimization
  - Minimize the latency
  - Reduce imbalances
- No need of optimizing imbalances prior to technology mapping
  - Inverters are not represented
  - Logic is not mapped
- Apply iteratively optimization algorithms
  - Interleave delay- and area-oriented methods

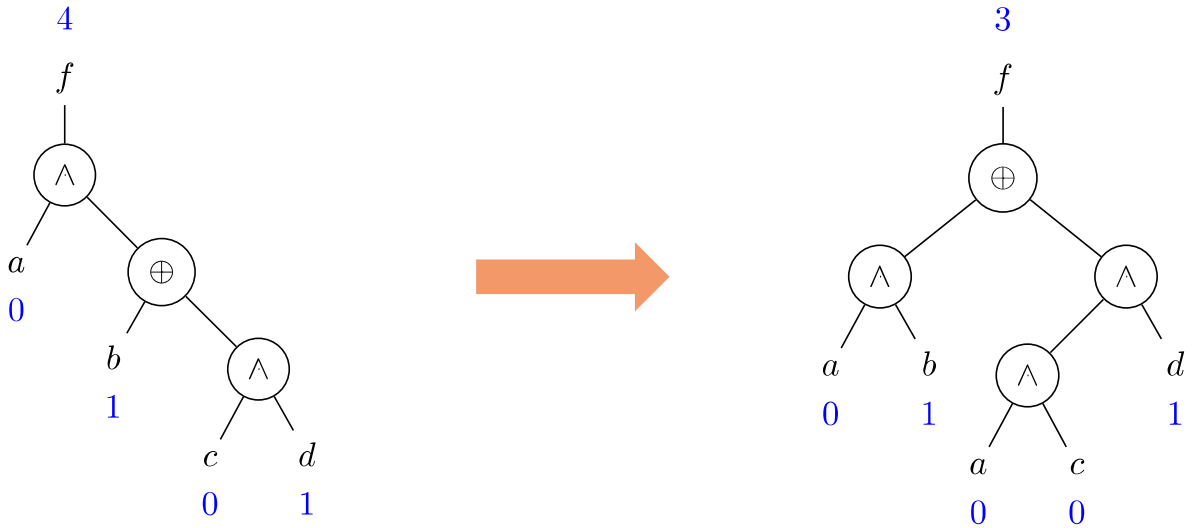
Logic synthesis: XAG mapping and rewriting

- Use of “Versatile mapper” [3]
- Maps from and to different graph representations while optimizing
  - Uses a database of XAG-optimum structures obtained using exact synthesis
- The best method to obtain a compact representation fast
- Used to get the **initial XAG** and perform **logic rewriting**

[3] A. T. Calvino, H. Riener, S. Rai, A. Kumar and G. De Micheli, "A Versatile Mapping Approach for Technology Mapping and Graph Optimization," *ASP-DAC*, 2022

Logic synthesis: XAG algebraic rewriting

- Use rules based on the **axioms of Boolean algebra**
  - Associativity of AND, OR, and XOR
  - Distributivity of AND over OR and AND over XOR
- Use to **reduce the depth** over the **critical paths**



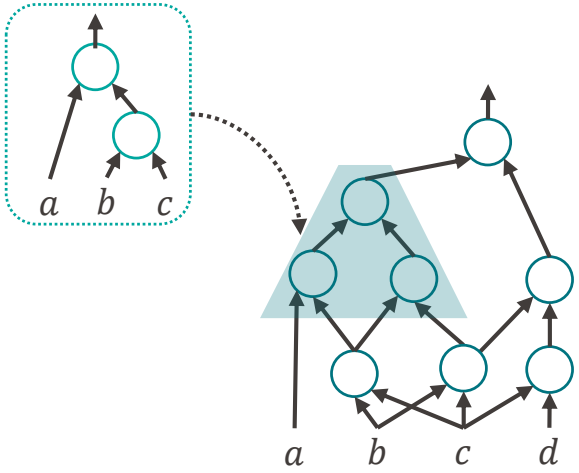
Logic synthesis: ESOP-based rewriting

- Rewrites cone of logic to minimize depth with *Exclusive-Sum-of-Products*
  - More compact than SOPs for many functions [4]
- Steps
  1. Extract the cone
  2. Compute the function as an *Exclusive-Sum-of-Products* using [5]
  3. Compute the Huffman decomposition to minimize depth
  4. Save if better
- Applied using technology mapping heuristics
  - Area recovery on the non-critical paths

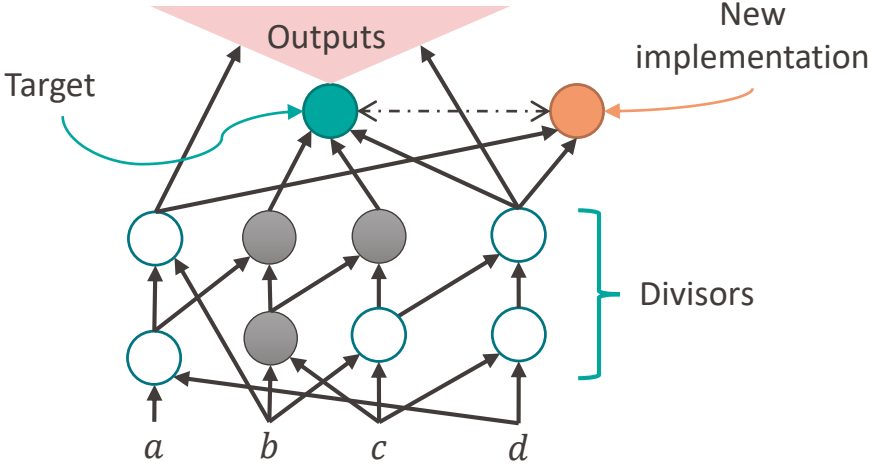
[4] T. Sasao and M. Fujita, “Representation of discrete functions,” Springer, 1996
[5] A. Mishchenko and M. Perkowski, “Fast heuristic minimization of exclusive-sums-of-products,” Intern. Reed-Muller Workshop, 2001

Logic synthesis: XAG area optimization

- XAG-based Boolean rewriting
  - Rewrites cones of the graph with optimum implementations
- XAG-based Boolean resubstitution
  - Substitute a node with a new implementation generated from divisor nodes



Boolean rewriting



Boolean resubstitution

Technology Mapping

- Compute **supergates**
  - Compose more complex SFQ gates by combining primitives
  - Reduces structural bias → better mapping
- Map directly from XAG
- Delay-oriented mapping
- Does not considers imbalances in the cost function
- Compared to previous work [6]
  - Does not considers imbalances, because:
    - Overestimate the number of DFFs
    - No optimal placement of DFFs
  - Cost functions are based on delay, area, and #input pins

[6] G. Pasandi and M. Pedram, "PBMap: A Path Balancing Technology Mapping Algorithm for Single Flux Quantum Logic Circuits," in IEEE Transactions on Applied Superconductivity, 2019

Balancing optimization and splitters

- Balancing **DFFs insertion**
  - ASAP policy
  - Sharing common DFFs
  - Algorithm is linear w.r.t. the number of gates
- DFFs are optimized using **minimum-area retiming**
  - Optimal number of DFFs
- **Splitter insertion**
  - Usual capacity is *1-to-2*
  - Balanced trees to limit the propagation time

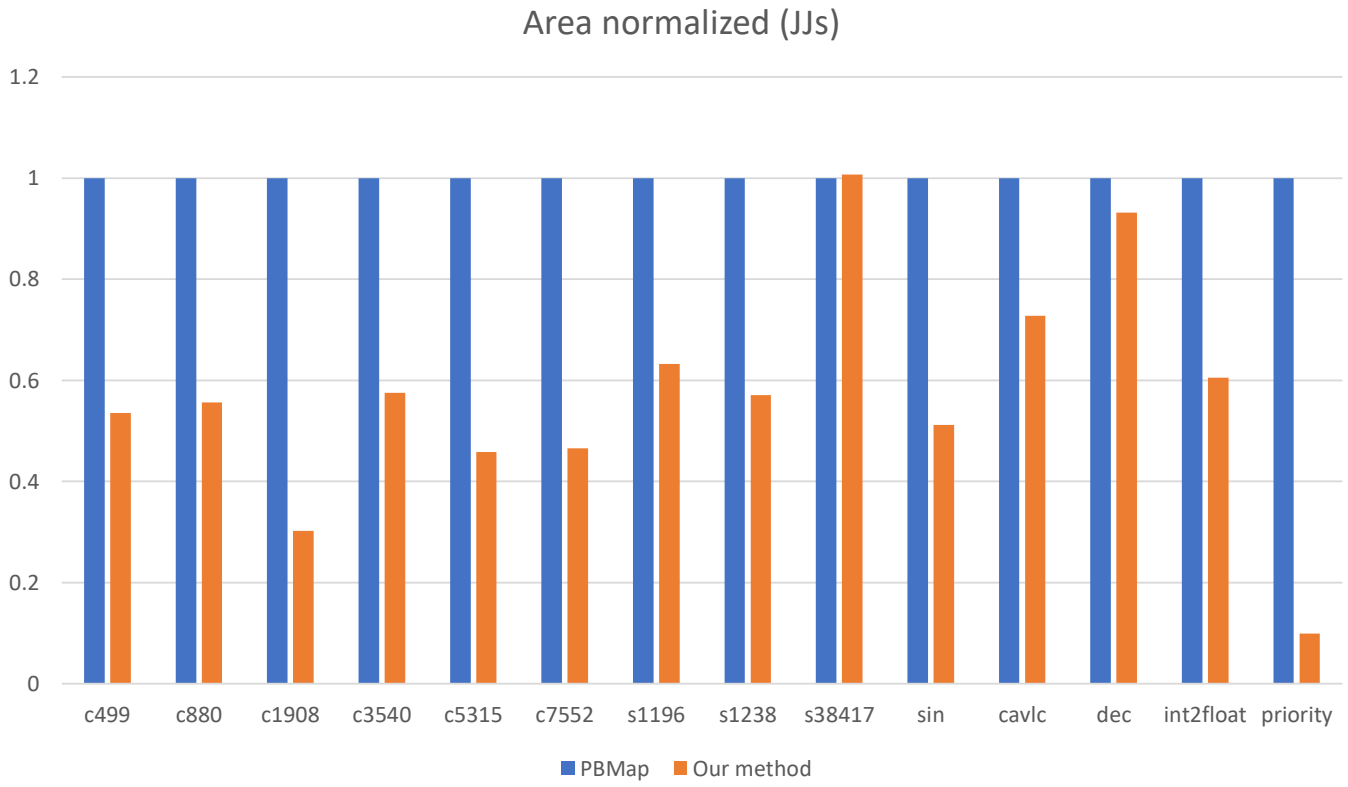
Experiments

Setup

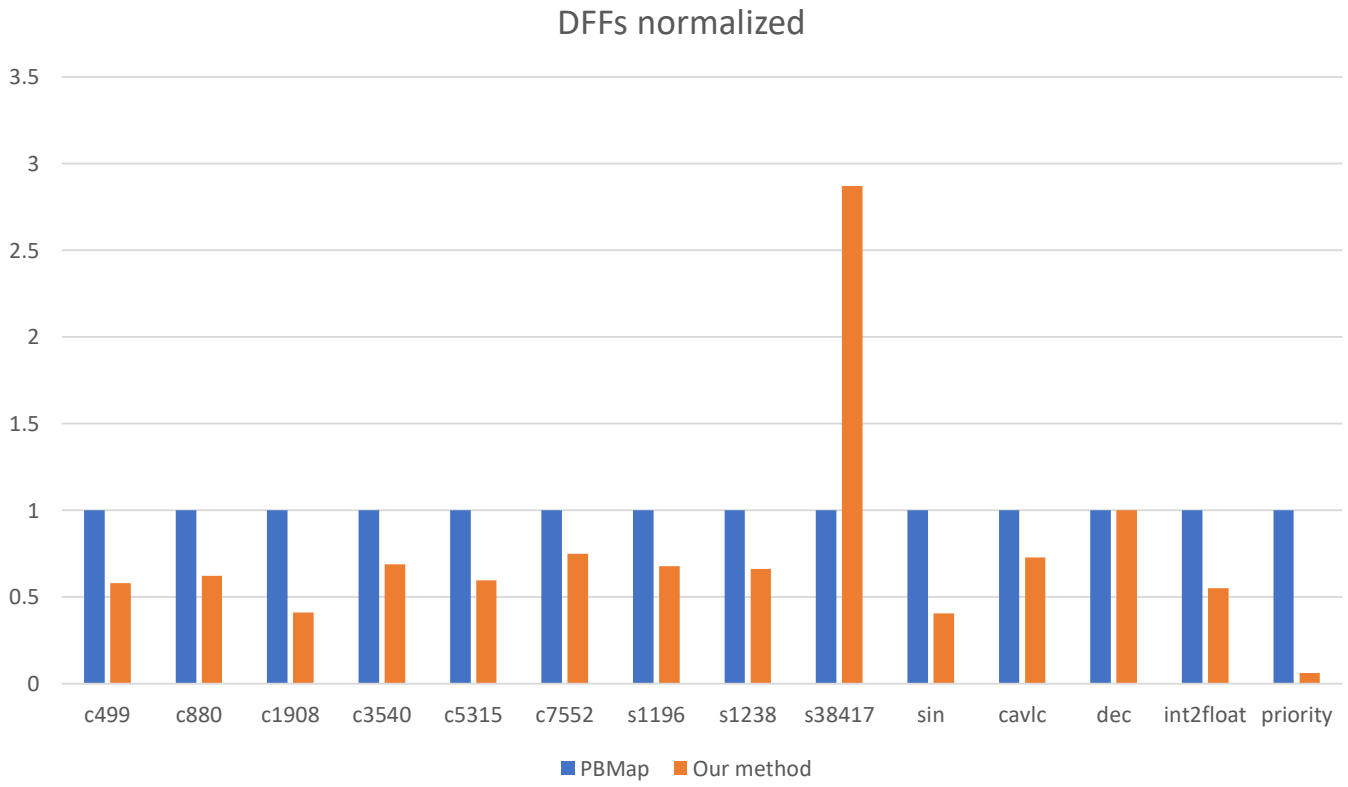
- State of the art [6]
  - Cost function in technology mapping based on **imbalances**
    - Optimal for trees of logic
    - **Not optimal** for DAGs
    - With supergates
  - Post-mapping optimization
    - Insertion and optimization of DFFs
  - Does not balance the POs (lower throughput)
- Our setting
  - XAG-based synthesis
  - Technology mapping
  - Post-mapping optimization
  - PO balancing (higher throughput)

[6] G. Pasandi and M. Pedram, "PBMap: A Path Balancing Technology Mapping Algorithm for Single Flux Quantum Logic Circuits," in IEEE Transactions on Applied Superconductivity, 2019

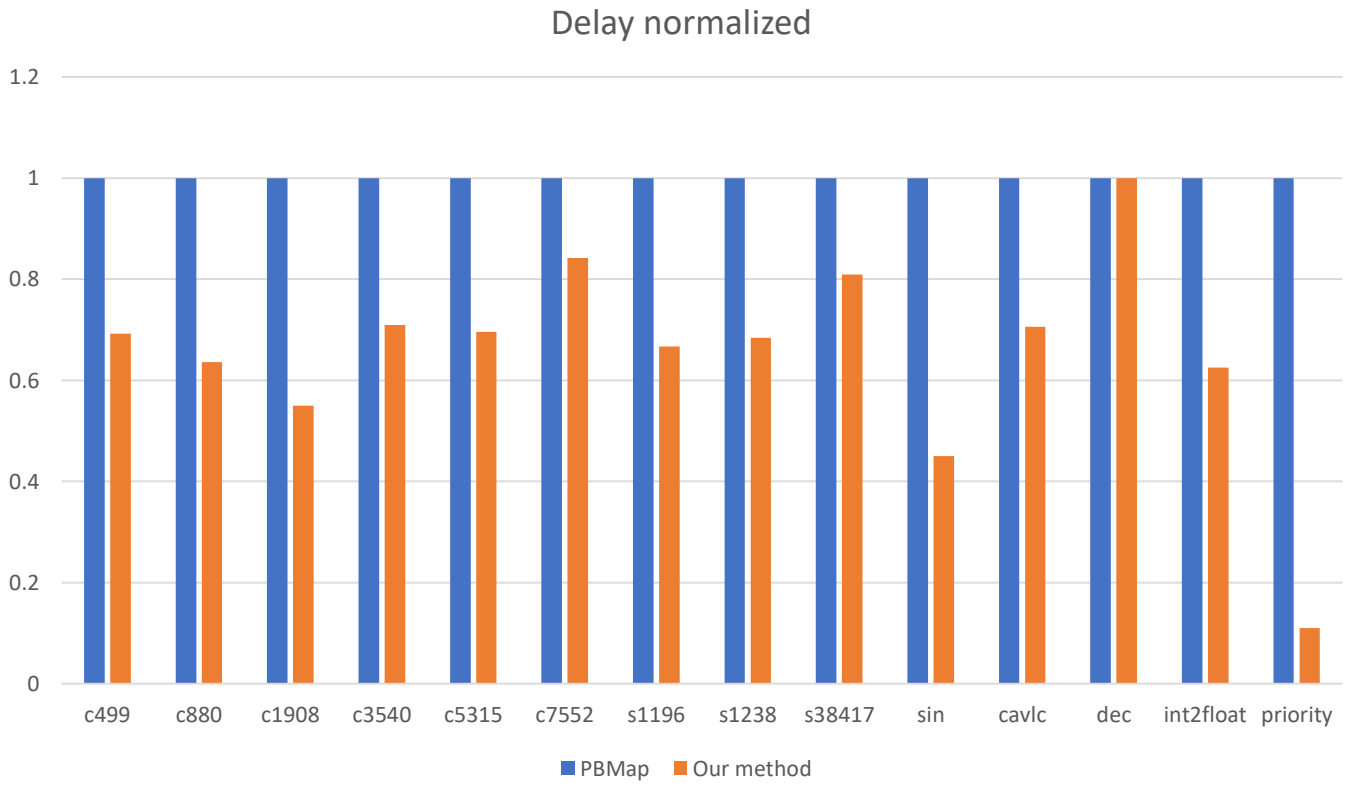
Area results



DFFs results



Delay results



Conclusion

Conclusion

- Framework to **synthesize** SFQ circuits
- Algebraic and Boolean optimization methods
  - Based on the AND and XOR primitives
- Technology Mapping and balancing DFF and splitters insertion
- Compared to the *state of the art*
  - **-43% Area**
  - **-24.2% DFFs**
  - **-34.44% Delay**
- Open-source implementation in the library *Mockturtle*
  https://github.com/lsils/mockturtle

Algebraic and Boolean Methods for SFQ Superconducting Circuits

Alessandro Tempia Calvino

Giovanni De Micheli

ASP-DAC 2024

Integrated Systems Laboratory, EPFL, Lausanne, Switzerland

alessandro.tempiacalvino@epfl.ch

24th Jan 2024