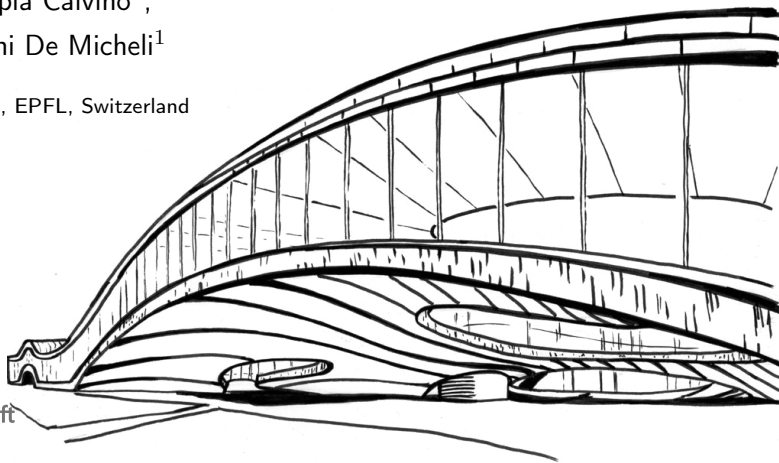
Back-end-aware Fault-tolerant Quantum Oracle Synthesis

Mingfei Yu[1], Alessandro Tempia Calvino[1],
Mathias Soeken[2] and Giovanni De Micheli[1]

[1]Integrated Systems Laboratory (LSI), EPFL, Switzerland
[2]Microsoft Quantum, Switzerland

January 22, 2025, at ASP-DAC 2025



EPFL LSI Integrated Systems Laboratory Microsoft

Outline

- Introduction
- Motivation
- Methodologies
- Experimental Evaluation
- Conclusion and Discussion

Introduction: Quantum Oracles

What is a quantum oracle:

- A quantum circuit that implements a Boolean function.
- Given a Boolean function $f(x)$, an oracle O_f realizes: $|x\rangle |y\rangle |0\rangle^l \mapsto |x\rangle |y \oplus f(x)\rangle |0\rangle^l$.

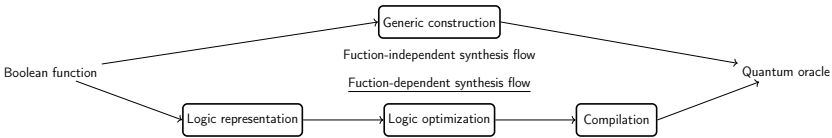
Why do we need quantum oracles:

- Fundamental component in many quantum algorithms and applications:
  - *Shor's algorithm* [1]: Realizing modular exponentiation function in *phase estimation*.
  - *Quantum chemistry applications* [2]: Encoding Hamiltonian matrices.

[1] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (1997), pp. 1484–1509. ISSN: 1095-7111

[2] Alán Aspuru-Guzik et al. "Simulated Quantum Computation of Molecular Energies". In: *Science* 309.5741 (2005), pp. 1704–1707.

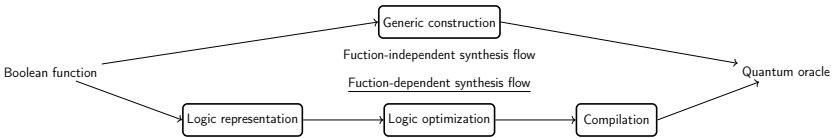
Introduction: Quantum Oracle Synthesis Flow



Function-independent oracle synthesis:

- Requiring more resources than function-dependent ones but might be favored in practice.
  - A generic construction ensures a uniform layout.
  - Suited for cases where target functions require frequent reconfiguration.

Introduction: Quantum Oracle Synthesis Flow



Function-independent oracle synthesis:

- Requiring more resources than function-dependent ones but might be favored in practice.
  - A generic construction ensures a uniform layout.
  - Suited for cases where target functions require frequent reconfiguration.

Function-dependent oracle synthesis:

- *XOR-AND-invertor graphs* (XAGs) is an ideal logic representation.
  - Correlation between AND nodes in an XAG and T gates in a Clifford+T oracle.

Motivation: XAGs for Function-dependent Oracle Synthesis

High-fidelity T gates are resource-intensive compared to Clifford gates.

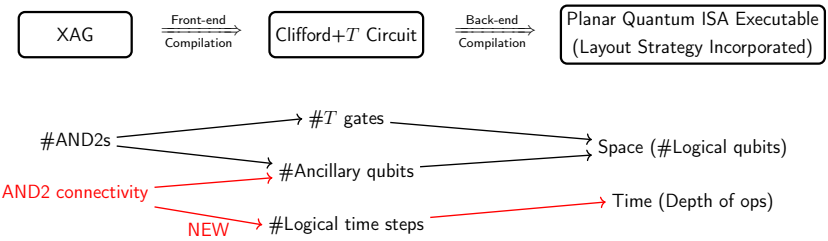
- Two-input XOR node (XOR2) can be realized using a CNOT gate.
- Two-input AND node (AND2) is the only primitive in an XAG whose quantum implementation requires T gates.

Motivation: XAGs for Function-dependent Oracle Synthesis

High-fidelity T gates are resource-intensive compared to Clifford gates.

- Two-input XOR node (XOR2) can be realized using a CNOT gate.
- Two-input AND node (AND2) is the only primitive in an XAG whose quantum implementation requires T gates.

The role of XAGs in generating low-cost oracle designs:



Motivation: A Cost Model Facilitated by Considering Layout Strategy

Consider the *parallel synthesis sequential Pauli computation* (PSSPC) layout strategy [1]:

- A T-efficient construction of 3-control Toffoli gates is available [2].
- Analysis on cost measures:

Logic operation	#T gates	#Logical time steps	#Ancillary qubits
AND2	4	4	1
Two AND2s	8	8	2
AND3	8	7	1

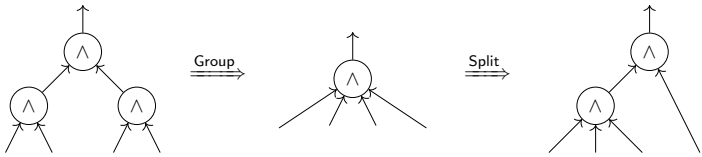
  Concatenated AND2s are cheaper than isolated ones!

- How to achieve lower-resource-cost quantum oracle designs?

[1] Michael E. Beverland et al. *Assessing Requirements to Scale to Practical Quantum Advantage*. 2022. arXiv: 2211.07629

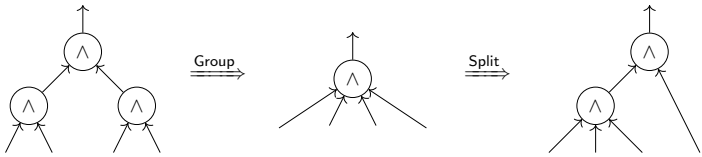
[2] Craig Gidney and N. Cody Jones. *A CCCZ Gate Performed with 6 T Gates*. 2021. arXiv: 2106.11513

Methodologies: Group-Split



- To exploit the resource-efficient execution of AND3 nodes via *group-split*:
  - Locate the AND trees in the given XAG.
  - Maximally split the AND2 nodes in each tree into pairs of concatenated ones.
  - The execution of each pair of AND2s follows the resource-efficient AND3 operation.

Methodologies: Group-Split



- To exploit the resource-efficient execution of AND3 nodes via *group-split*:
  - Locate the AND trees in the given XAG.
  - Maximally split the AND2 nodes in each tree into pairs of concatenated ones.
  - The execution of each pair of AND2s follows the resource-efficient AND3 operation.
- 1.81% fewer logical time steps and 7.47% fewer ancillary qubits.
  - How to unlock more optimization opportunities by restructuring XAGs?
  - Efficiently and effectively "massage" XAGs $\rightarrow$ AND nodes are clustered together.

Methodologies: XAG Optimization via Cut Rewriting

Cut rewriting is a peephole logic optimization technique, achieving a better logic network design by replacing some sub-networks with their optimal implementation.

[1] Mingfei Yu and Giovanni De Micheli. "Striving for Both Quality and Speed: Logic Synthesis for Practical Garbled Circuits". In: *International Conference on Computer-Aided Design*. 2023, pp. 1–9

Methodologies: XAG Optimization via Cut Rewriting

Cut rewriting is a peephole logic optimization technique, achieving a better logic network design by replacing some sub-networks with their optimal implementation.

- It features a database of optimal implementations or an exact synthesis engine.
  - A SAT formulation capable of exactly synthesizing optimal XAGs whose cost metric consists of both AND count and AND connectivity is available [1].
  - An XAG database for up to 5-variable functions is generated, with AND count set as the primary cost measure.

[1] Mingfei Yu and Giovanni De Micheli. "Striving for Both Quality and Speed: Logic Synthesis for Practical Garbled Circuits". In: *International Conference on Computer-Aided Design*. 2023, pp. 1–9

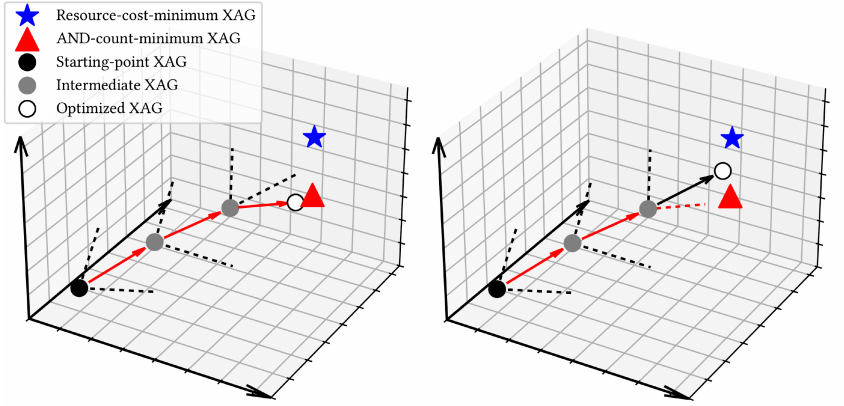
Methodologies: XAG Optimization via Cut Rewriting

Cut rewriting is a peephole logic optimization technique, achieving a better logic network design by replacing some sub-networks with their optimal implementation.

- It features a database of optimal implementations or an exact synthesis engine.
  - A SAT formulation capable of exactly synthesizing optimal XAGs whose cost metric consists of both AND count and AND connectivity is available [1].
  - An XAG database for up to 5-variable functions is generated, with AND count set as the primary cost measure.
- Assessing the plausibility of each replacement is tricky.
  - Network topology forms part of the cost metric, which requires a global view that cut rewriting lacks.
  - Rewriting a region changes the AND connectivity at the border, which may offset the obtained gain.

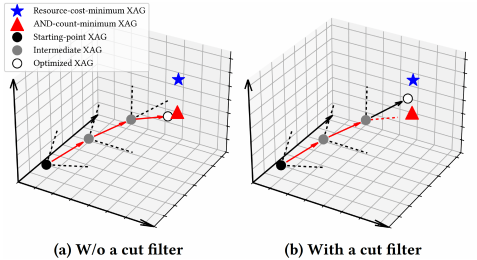
[1] Mingfei Yu and Giovanni De Micheli. "Striving for Both Quality and Speed: Logic Synthesis for Practical Garbled Circuits". In: *International Conference on Computer-Aided Design*. 2023, pp. 1–9

Methodologies: Behavioral Expectations on Cut Filters



(a) W/o a cut filter **(b) With a cut filter**

Methodologies: Cut Filter-Facilitated Cut Rewriting



Two *cut filters* are designed to introduce the required information.

- Baseline: always rewrite a region once a reduction in AND count can be achieved, regardless of structural information.
- Rigid: give up rewriting a region if any of its borders are within an AND tree.
- Voter-driven: give up if its borders within an AND tree exceed a threshold.

Experimental Evaluation

- Benchmark: the EPFL combinational benchmark suite, with state-of-the-art AND count reduction technique [1] applied.

Methodology	#T gates	#Logical time steps	#Ancillary qubits	Optimization time
Starting-point	1	1	1	-
Group-split	1	0.925	0.982	-
Baseline	0.959	0.858	0.935	1
Rigid	0.960	0.849	0.934	0.846
Voter-driven	0.955	0.851	0.930	5.485

- The power of logic restructuring: On `priority encoder`, improvements achieved by the rigid cut filter(group-and-split) are $17.65\%(0\%)$, $20.51\%(2.09\%)$, and $29.01\%(8.33\%)$.
- The voter-driven cut filter serves as a reliable cut filter.

[1] Hsiao-Lun Liu et al. "A Don't-care-based Approach to Reducing the Multiplicative Complexity in Logic Networks". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41.11 (2022), pp. 4821–4825

Conclusion and Discussion

Leveraging back-end insights to guide the front-end quantum oracle synthesis task:

- When optimizing XAGs, not only AND count, but also their connectivity matters.
- Customized XAG optimization algorithm achieves average reductions of 4.49% **in T count**, 7.00% in logical time steps, and 14.89% in ancillary qubit count.

Conclusion and Discussion

Leveraging back-end insights to guide the front-end quantum oracle synthesis task:

- When optimizing XAGs, not only AND count, but also their connectivity matters.
- Customized XAG optimization algorithm achieves average reductions of 4.49% **in T count**, 7.00% in logical time steps, and 14.89% in ancillary qubit count.

Instead of formulating a technology mapping problem over the gate set {XOR2, AND2, AND3}, our formulation

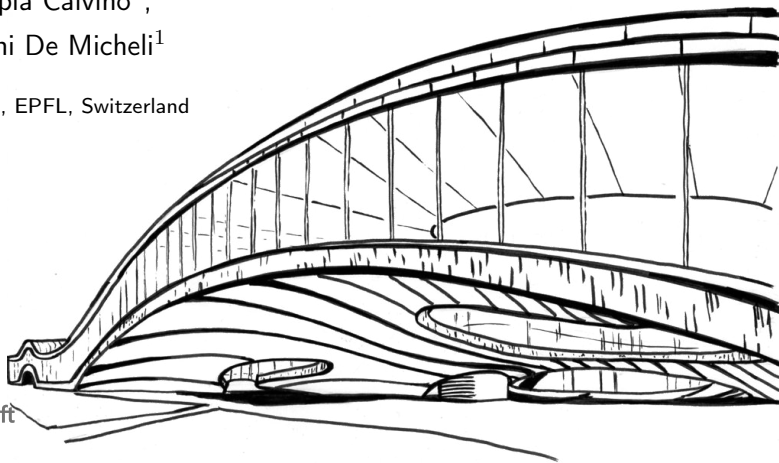
- maintains flexibility and can be easily adapted to accommodate potential T-efficient construction of multi-control Toffoli gates.
- presents a unique and technically interesting logic optimization problem, inspiring future exploration.

Back-end-aware Fault-tolerant Quantum Oracle Synthesis

Mingfei Yu[1], Alessandro Tempia Calvino[1],
Mathias Soeken[2] and Giovanni De Micheli[1]

[1]Integrated Systems Laboratory (LSI), EPFL, Switzerland
[2]Microsoft Quantum, Switzerland

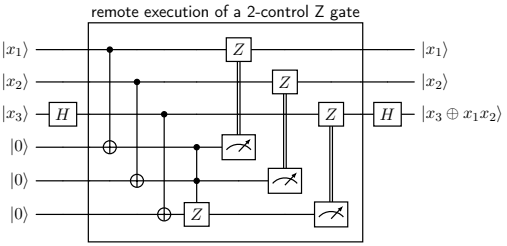
January 22, 2025, at ASP-DAC 2025



EPFL LSI Integrated Systems Laboratory Microsoft

Appx.: Leveraging Layout Insights for Efficient Quantum Oracles

Consider the *parallel synthesis sequential Pauli computation* (PSSPC) layout strategy [1]:



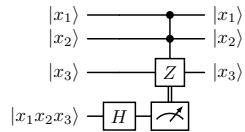
- Delegating non-Clifford gate execution (Toffoli, rotation, etc.) to "remote" qubits.
  - Parallelizing non-Clifford gate execution.
  - Synthesis qubits have easier access to the T factory.
  - An additional quality measure, *logical time steps*, is available: 4 steps per AND2.

[1] Michael E. Beverland et al. *Assessing Requirements to Scale to Practical Quantum Advantage*. 2022. arXiv: 2211.07629.

Appx.: Optimization Facilitated by Advanced Gate Construction

A T-efficient construction of a 3-control Z gate has been proposed [1].

- Making available a 6-T realization of 3-input Boolean AND operation (AND3).



- Change of resource requirement of two concatenated AND2s:
  - T count: $8(4 \times 2) \rightarrow 8(6 + 0.5 \times 4)$
  - Logical time steps: $8(4 \times 2) \rightarrow \ 7(5 + 0.5 \times 4)$
  - Ancillary qubit count: $2(1 \times 2) \rightarrow \ 1$
  - Two concatenated AND2 nodes are better than two separated ones regarding logical time steps and ancillary qubit count.
- Extend the XAG optimization problem: how to maximally benefit from this observation?

[1] Craig Gidney and N. Cody Jones. *A CCCZ Gate Performed with 6 T Gates.* 2021. arXiv: 2106.11513