The impact of compilation in the implementation of quantum computing

Software to the rescue of hardware

Simon Martiel
Researcher @ Atos Quantum
11/01/2023

When software helps hardware: quantum circuit compilation

Quantum software is a (very) broad topic:

- HPC integration
- Cloud integration
- Quantum Programming languages
- Formal methods (ZX calculus, static analysis)

In this talk we will focus on Quantum Circuit compilation/optimization

Quantum Circuit Compilation/Transpilation

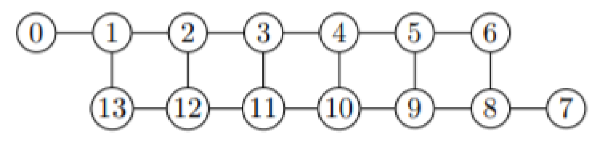
But why ?

Typical quantum circuits:

- Contain large gates (C.....CNOTs) and black-boxed primitives
- Contain all kind of weird gates

Typical quantum hardware (NISQ setting):

- Comes with gate-set limitation ({ CNOT, U3 }, { CZ, $RX(\frac{\pi}{2})$, RZ }, ….)
- (Usually) comes with connectivity restrictions
- Each operation has some (large) error rate



<u>A good compiler needs to reduce gate count/depth while matching all those constraints !</u>

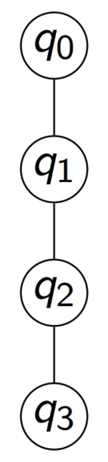
Focus on qubit routing

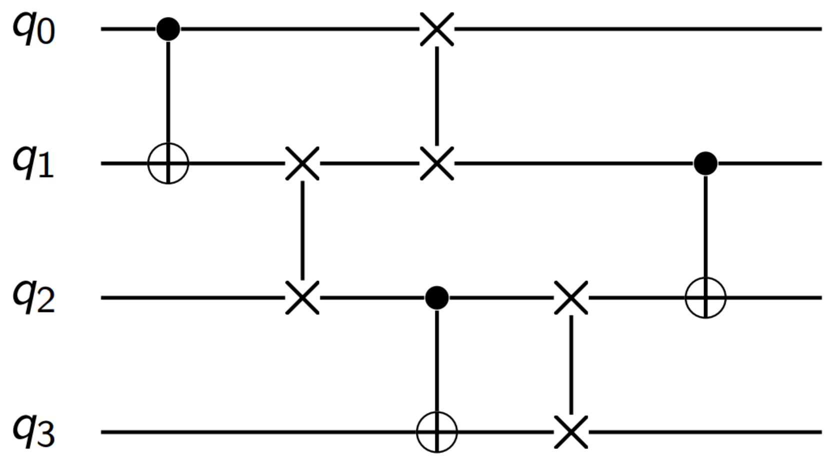
The problem

Input : Some circuit and some connectivity graph

Output : Some equivalent circuit matching the connectivity

Standard approach: SWAP insertion



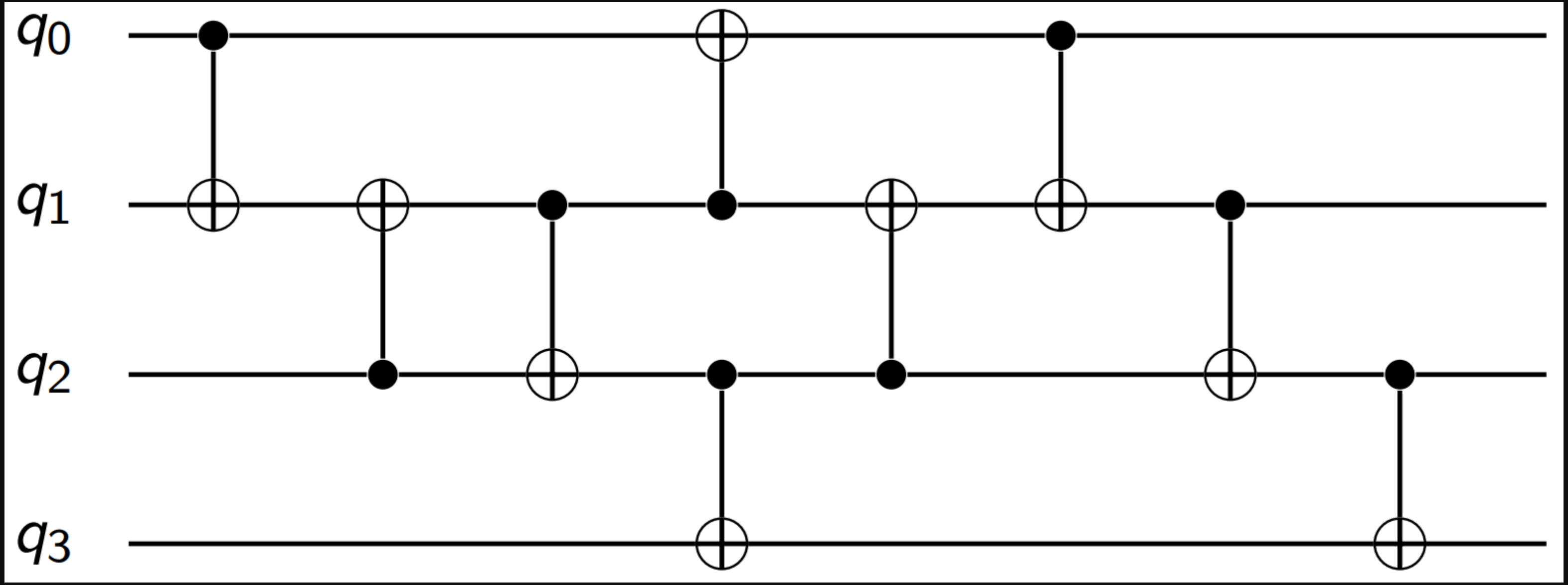


Final cost: 12 CNOTs
(3 + 3 SWAPS)

Focus on qubit routing

Ad hoc synthesis as an alternative to SWAP insertion

Can we be smarter (in that example) ?



This circuit is equivalent and contains 9 CNOTs !

[Kissinger et al. (2019)]

Focus on qubit routing

Ad hoc synthesis as an alternative to SWAP insertion

We know how to synthesize circuits for:

- Qubit permutations (SWAP circuits)
- Boolean linear maps (CNOT circuits)
- Phase Polynomials (CNOT + RZ circuits)
- Clifford (CNOT + H + S)

We don’t known how to do it for arbitrary circuits

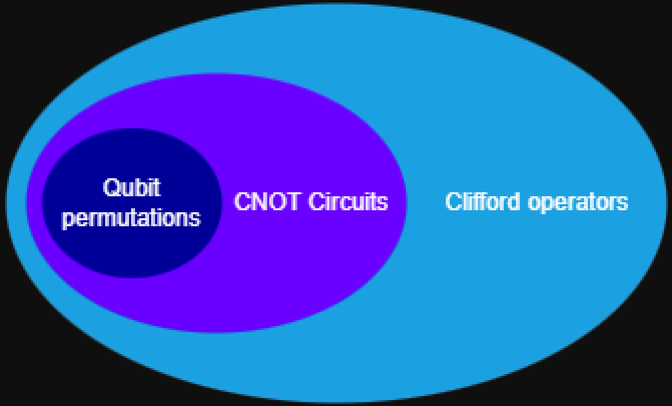
Our take on qubit routing

With Timothée Goubault de Brugière

Our take:

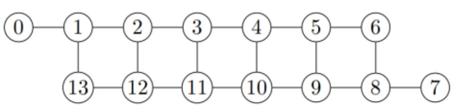
- Work with a particular set of operators
- Read circuit from left to right:
  - If the gate is in the set of operators => free cost (update current) operator
  - If not, lazily synthesize a piece of the operator using standard techniques

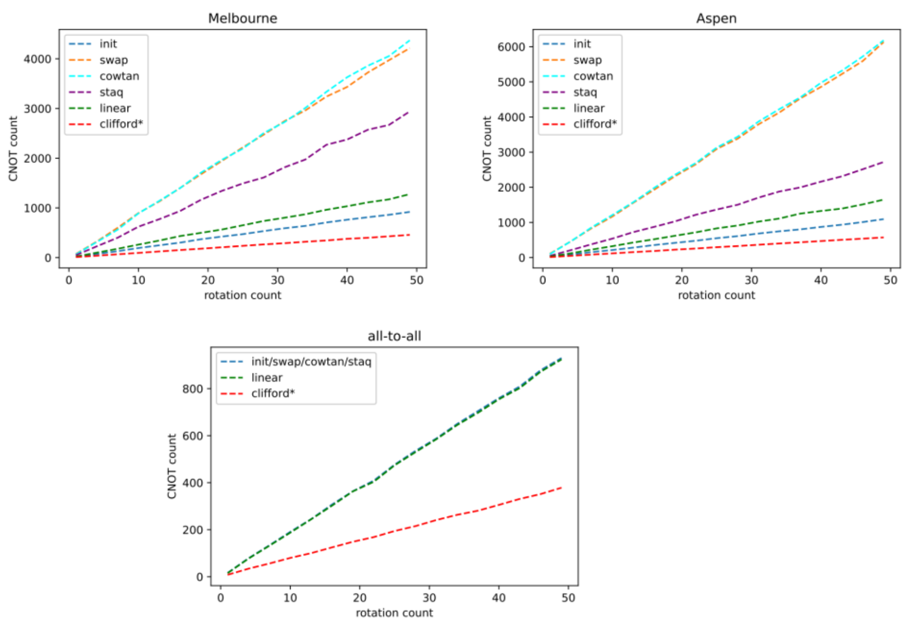
Implemented for the following operators:

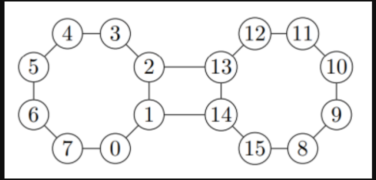


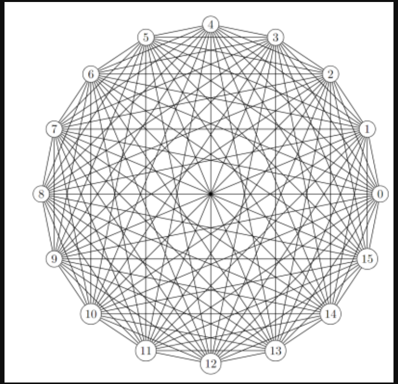
Quick benchmarks of our compiler

With Timothée Goubault de Brugière



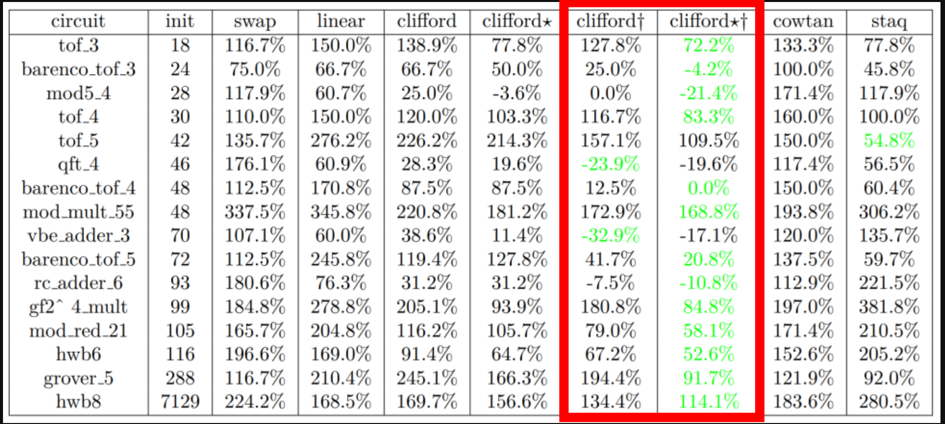




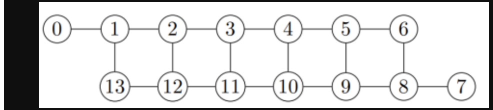


Quick benchmarks of our compiler

With Timothée Goubault de Brugière



circuit	init	swap	linear	clifford	clifford⋆	clifford†	clifford⋆†	cowtan	staq
tof_3	18	116.7%	150.0%	138.9%	77.8%	127.8%	72.2%	133.3%	77.8%
barenco_tof_3	24	75.0%	66.7%	66.7%	50.0%	25.0%	-4.2%	100.0%	45.8%
mod5_4	28	117.9%	60.7%	25.0%	-3.6%	0.0%	-21.4%	171.4%	117.9%
tof_4	30	110.0%	150.0%	120.0%	103.3%	116.7%	83.3%	160.0%	100.0%
tof_5	42	135.7%	276.2%	226.2%	214.3%	157.1%	109.5%	150.0%	54.8%
qft_4	46	176.1%	60.9%	28.3%	19.6%	-23.9%	-19.6%	117.4%	56.5%
barenco_tof_4	48	112.5%	170.8%	87.5%	87.5%	12.5%	0.0%	150.0%	60.4%
mod_mult_55	48	337.5%	345.8%	220.8%	181.2%	172.9%	168.8%	193.8%	306.2%
vbe_adder_3	70	107.1%	60.0%	38.6%	11.4%	-32.9%	-17.1%	120.0%	135.7%
barenco_tof_5	72	112.5%	245.8%	119.4%	127.8%	41.7%	20.8%	137.5%	59.7%
rc_adder_6	93	180.6%	76.3%	31.2%	31.2%	-7.5%	-10.8%	112.9%	221.5%
gf2^ 4_mult	99	184.8%	278.8%	205.1%	93.9%	180.8%	84.8%	197.0%	381.8%
mod_red_21	105	165.7%	204.8%	116.2%	105.7%	79.0%	58.1%	171.4%	210.5%
hwb6	116	196.6%	169.0%	91.4%	64.7%	67.2%	52.6%	152.6%	205.2%
grover_5	288	116.7%	210.4%	245.1%	166.3%	194.4%	91.7%	121.9%	92.0%
hwb8	7129	224.2%	168.5%	169.7%	156.6%	134.4%	114.1%	183.6%	280.5%



Lazy operator synthesis:

- A framework for quantum circuit compilation
- More than competitive for VQE like circuits
- Almost always outperforms SWAP insertion

Published in Quantum

Architecture aware compilation of quantum circuits via lazy synthesis, S. M., Timothée Goubault de Brugière, Quantum

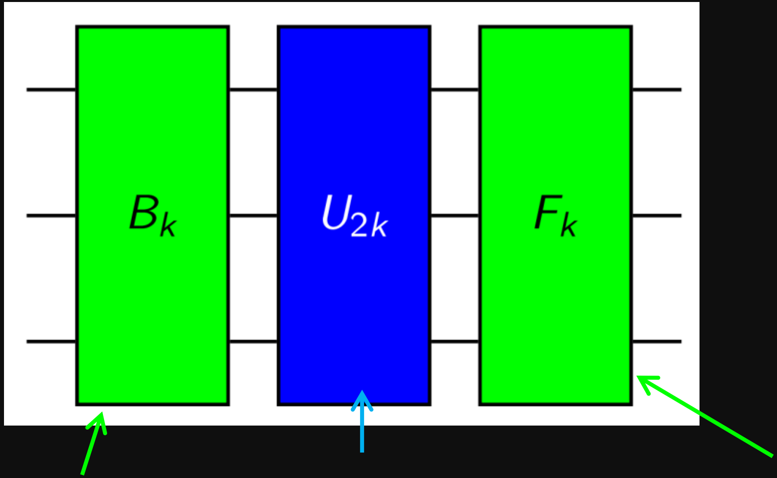
Extension to bidirectional normalization

With Arnaud Gazda, Timothée Goubault de Brugière, and Christophe Vuillot

Idea: explore Clifford basis to optimize the circuit

B_k is synthesized via
stabilizer state synthesis

F_k is synthesized via
Pauli operators co-diagonalization
+ classical post-processing



Clifford basis change

Compiled circuit

Clifford basis change

A graph-state based synthesis framework for Clifford isometries, Timothée Goubault de Brugière, S.M., Christophe Vuillot, Pre-print

A all-in-one compiler

Initial Transpilation
qat-pbo

Breaking down large gates

Rotation pruning
lazy synthesis

Bidirectional renormalization
lazy synthesis

Lazy operator synthesis (compilation & optimization)

2-qubit gate transpilation
qat-pbo

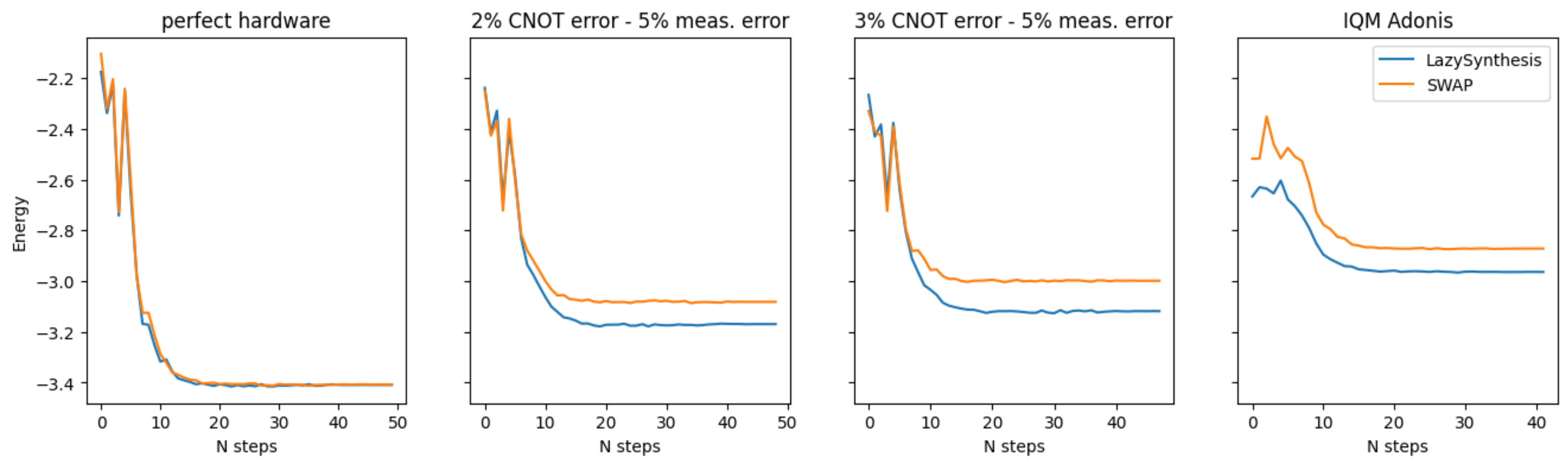
1-qubit gate compression
qat-pbo

Circuit rewriting + 1-qbit gate compression

A all-in-one compiler

Simulation and real hardware runs - Combinatorial Optimization applications

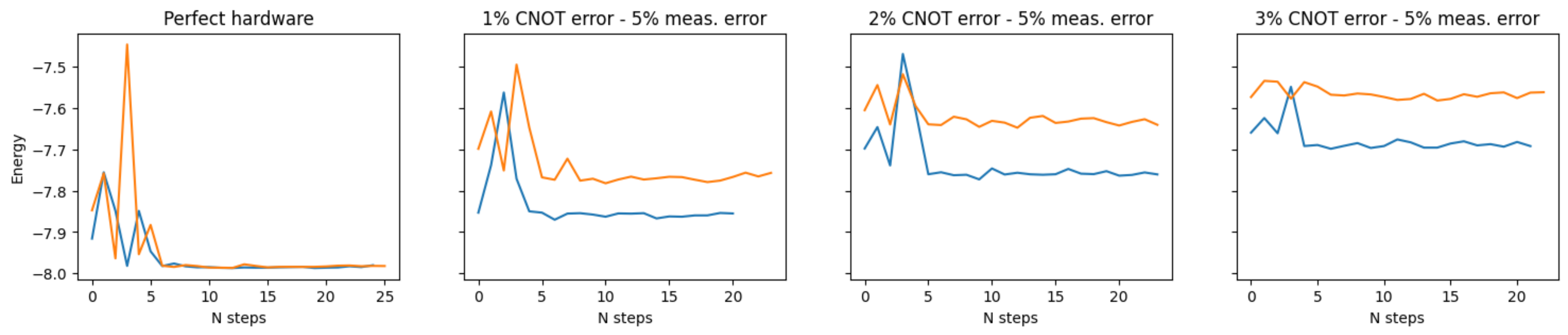
QAOA – MaxCut – $G(5,\frac{1}{2})$ – avg. over 100 runs



A all-in-one compiler

Simulation - Quantum Chemistry applications

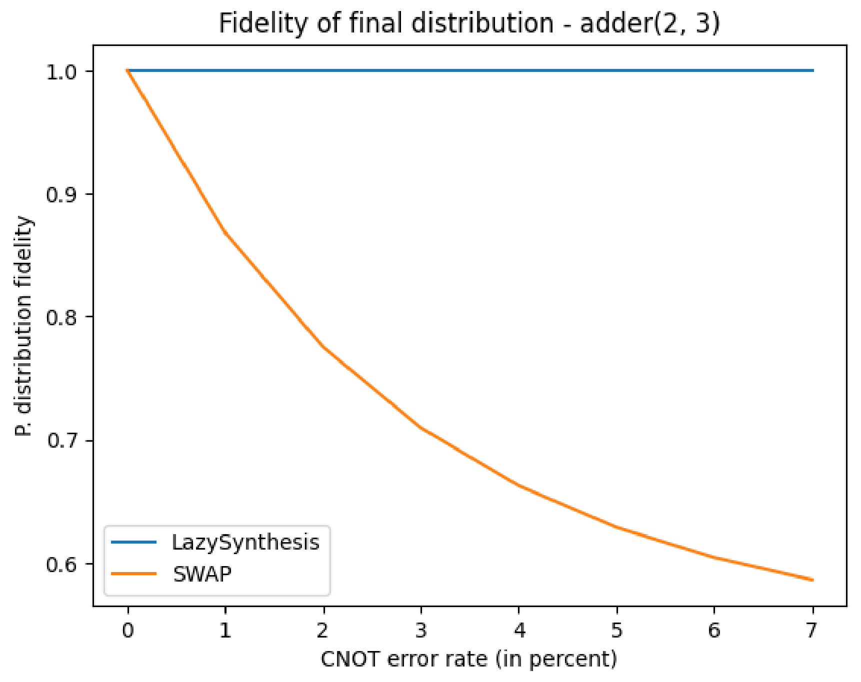
UCCSD VQE for LiH



A all-in-one compiler

Simulation - Arithmetic quantum circuit

QFT-based adder(2,3)



Conclusion and perspectives

We presented:

- A generic compilation framework
- And its embedding in an all-in-one compiler that covers most usages

Benchmarks show that the compiler does *increase the algorithmic performances* of the QPU

Available in the QLM framework : NISQCompiler plugin

Further work:

- A more subtle target metric (here we minimized the gate count)
- Scalability improvements (we are limited to less than 50 qbits)

Thank you ⍰

simon.martiel@atos.net

Atos