PIQUASSO

Zoltán Zimborás

Photonic Quantum Simulator Software

A framework for optical quantum computer programming and simulation

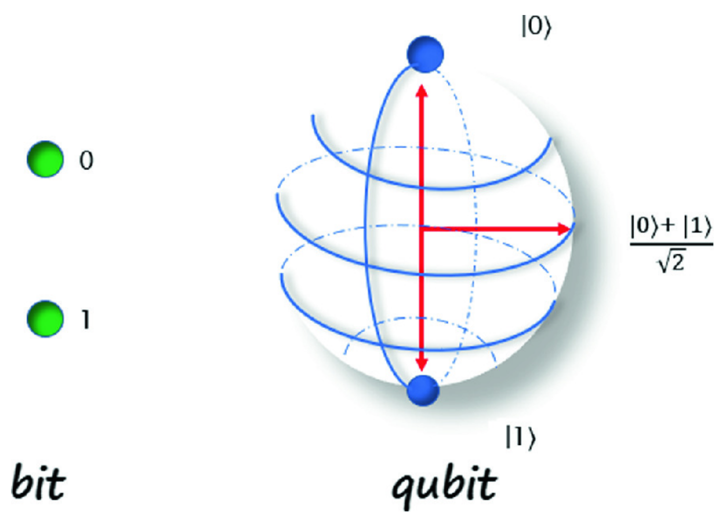
GPU Days

21 June 2022

Quantum Computing architectures in our focus

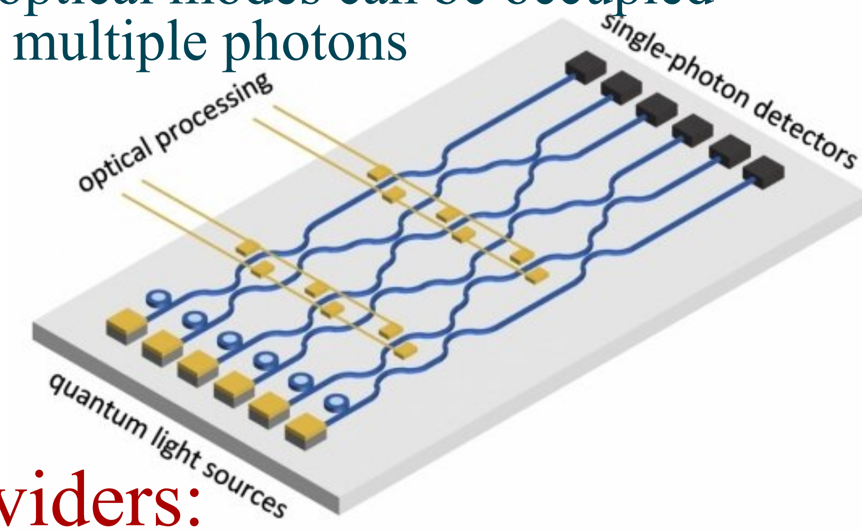
Quantum Information National Laboratory HUNGARY

Qubit based
(quantum bit)



Q-mode based
(photonic quantum computing)

The optical modes can be occupied with multiple photons



Hardware developers/vendors/cloud providers:

IBM, Google, Alibaba, PsiQuantum, IonQ, Microsoft, Rigetti, Photonic, Honeywell Quantum Solutions, Quix, Xanadu, Quandela, PhotoQ, Amazon, Atos, 1Qbit, ORCA, Huawei,

WIGNER

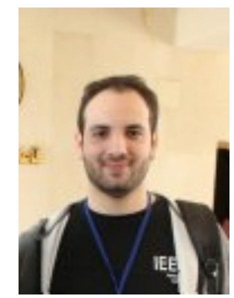
The Piquasso project

Quantum Information
National Laboratory
HUNGARY

Gregory Morse

Zoltán Zimborás
Wigner

Poór Boldizsár

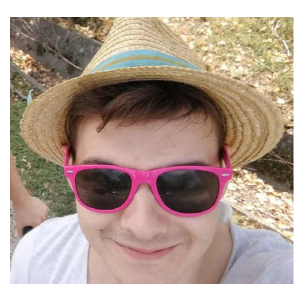


Kareem El Safty

Kozsik tamás

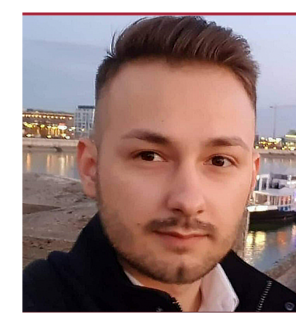
Michał Oszmaniec

Tomasz Rybotycki



Kolarovszki Zoltán

Kaposi Ágoston



Jóczik Szabolcs

Supported by

ERICSSON

PIQUASSO

Quantum Information National Laboratory HUNGARY

universal photonic quantum computer programming framework

PIQUASSO.com

Open source project developed by:

WIGNER

ELTE EÖTVÖS LORÁND UNIVERSITY

ERICSSON

GitHub https://github.com/Budapest-Quantum-Computing-Group

Piquasso
High level Python programming interface

Piquasso Boost
High performance C++ computing engines

Piquasso UI
Graphical user interface for online access

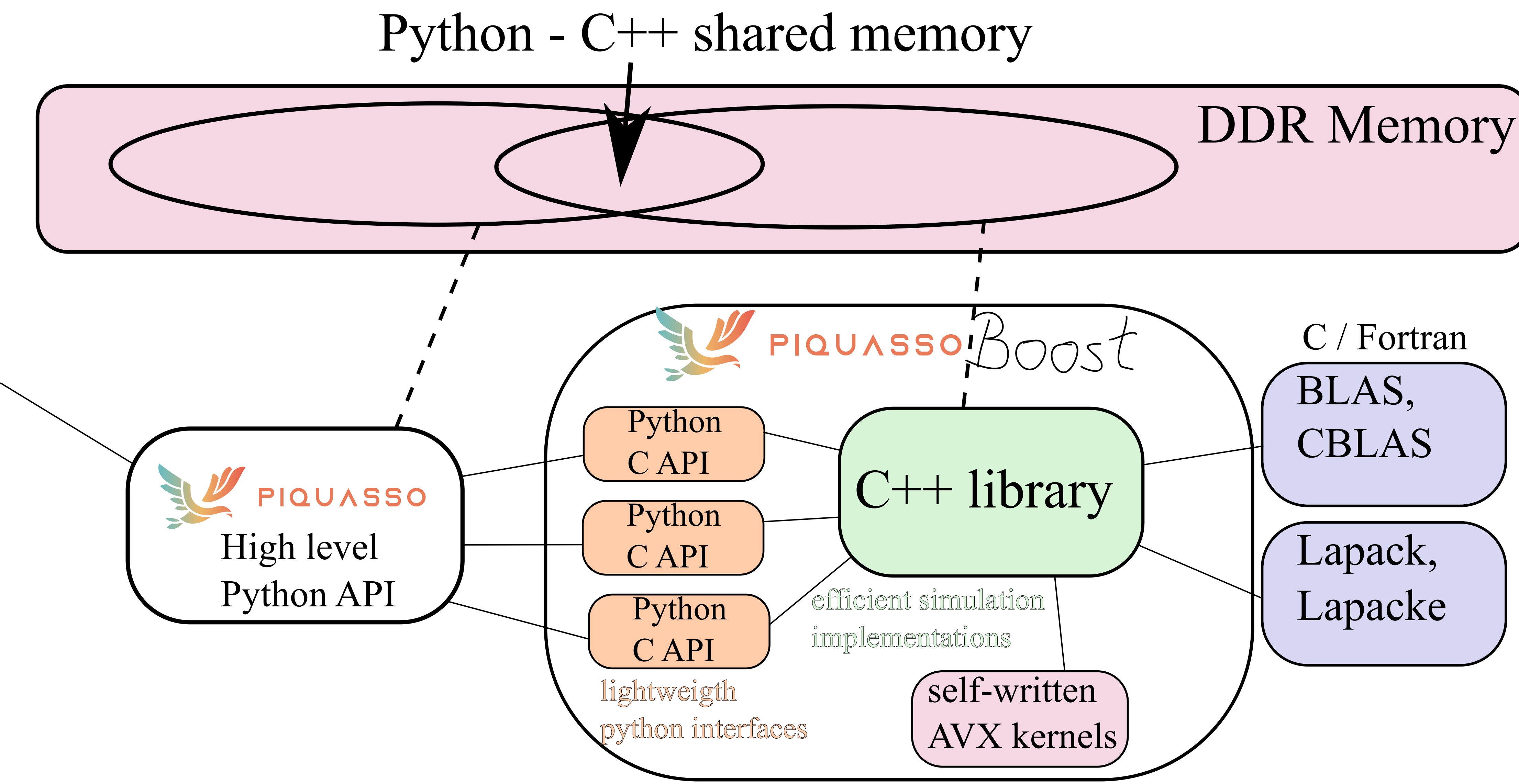
piquasso.com

New module (unpublished):

Piquasso DFE
Data-flow engine support for QC simulations

MAXELER Technologies Maximum Performance Computing

PIQUASSO Boost: simulation engine



Python C garbage collector

compatibility

thread-safe, automatic garbage collector for numeric arrays (+ thread-safe containers)

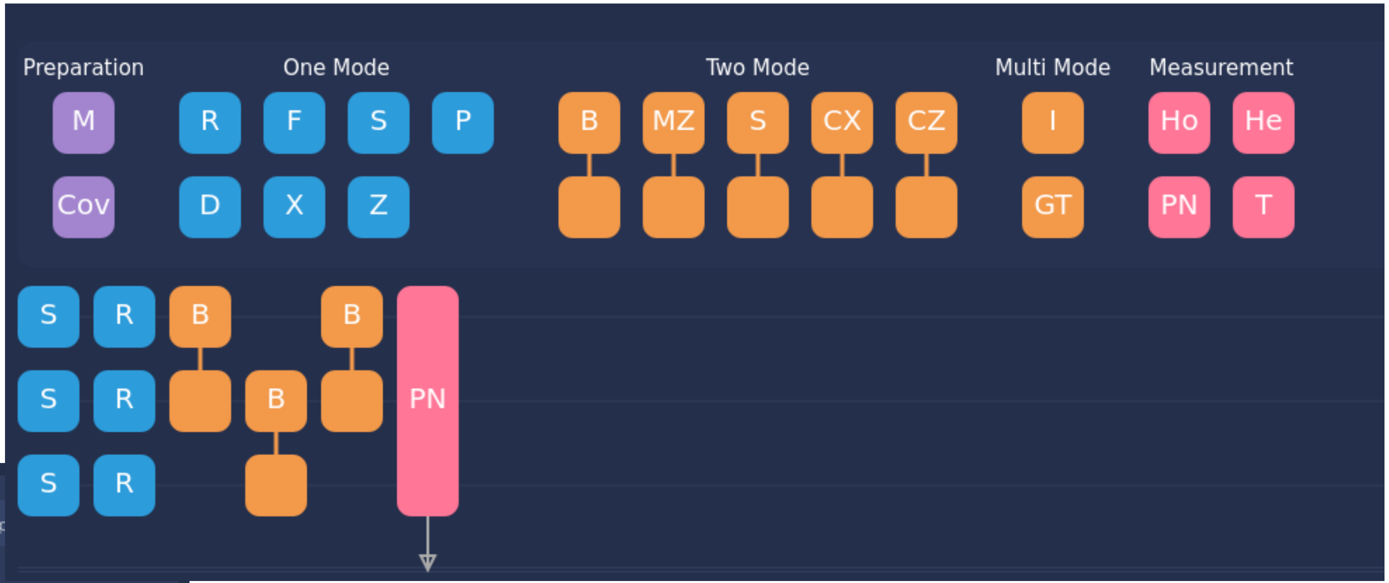
Stable memory management without leaks.

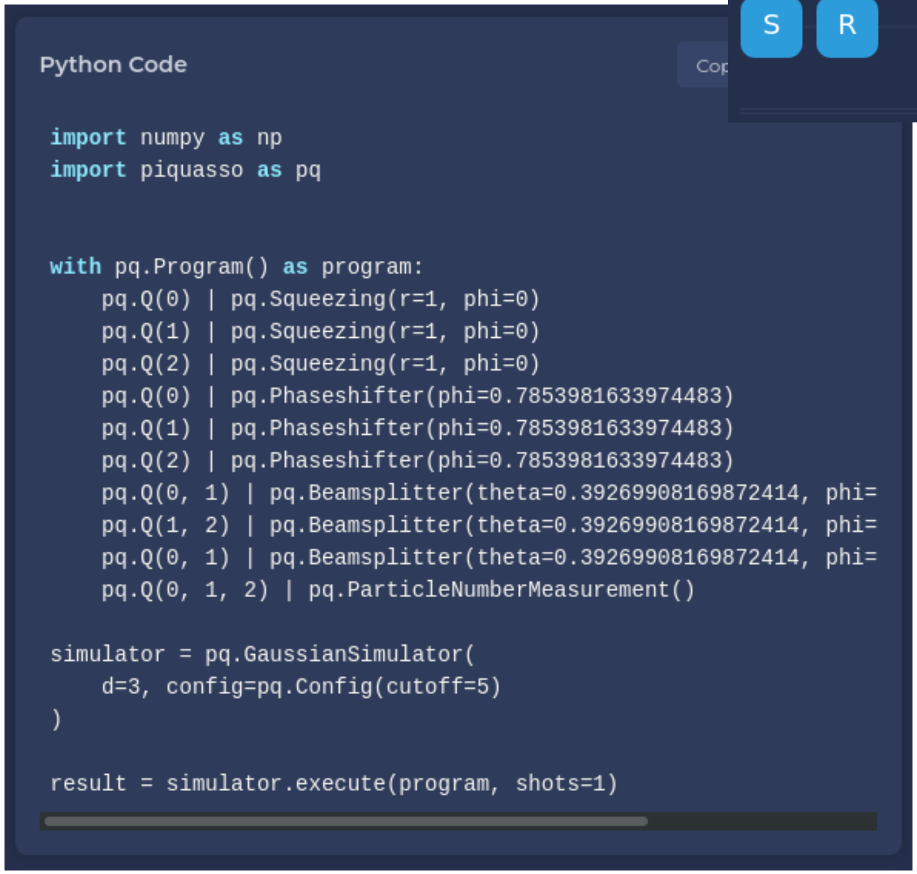
PIQUASSO UI

Quantum Information National Laboratory HUNGARY

PIQUASSO.com

Interactive circuit construction

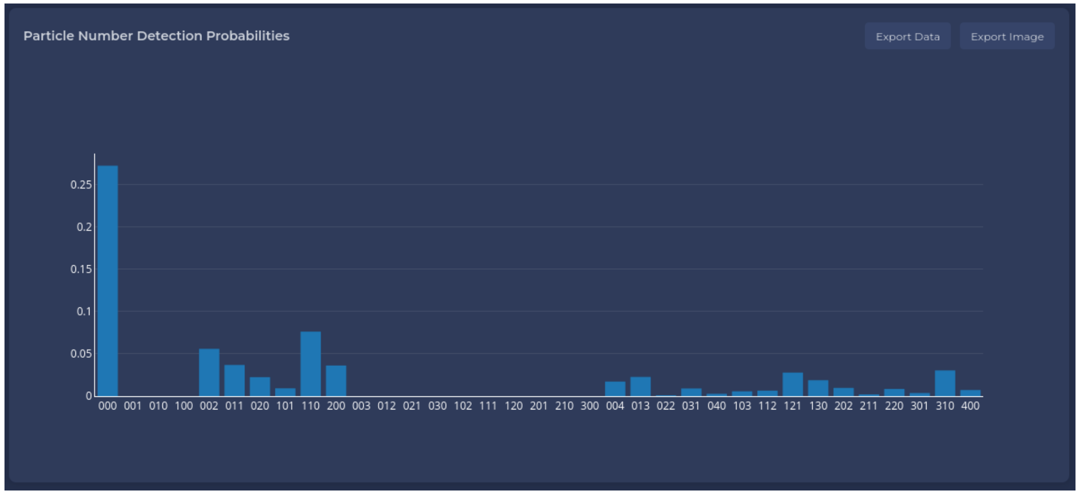





```
import numpy as np
import piquasso as pq


with pq.Program() as program:
    pq.Q(0) | pq.Squeezing(r=1, phi=0)
    pq.Q(1) | pq.Squeezing(r=1, phi=0)
    pq.Q(2) | pq.Squeezing(r=1, phi=0)
    pq.Q(0) | pq.Phaseshifter(phi=0.7853981633974483)
    pq.Q(1) | pq.Phaseshifter(phi=0.7853981633974483)
    pq.Q(2) | pq.Phaseshifter(phi=0.7853981633974483)
    pq.Q(0, 1) | pq.Beamsplitter(theta=0.39269908169872414, phi=
    pq.Q(1, 2) | pq.Beamsplitter(theta=0.39269908169872414, phi=
    pq.Q(0, 1) | pq.Beamsplitter(theta=0.39269908169872414, phi=
    pq.Q(0, 1, 2) | pq.ParticleNumberMeasurement()

simulator = pq.GaussianSimulator(
    d=3, config=pq.Config(cutoff=5)
)

result = simulator.execute(program, shots=1)
```

automatic Python code generation

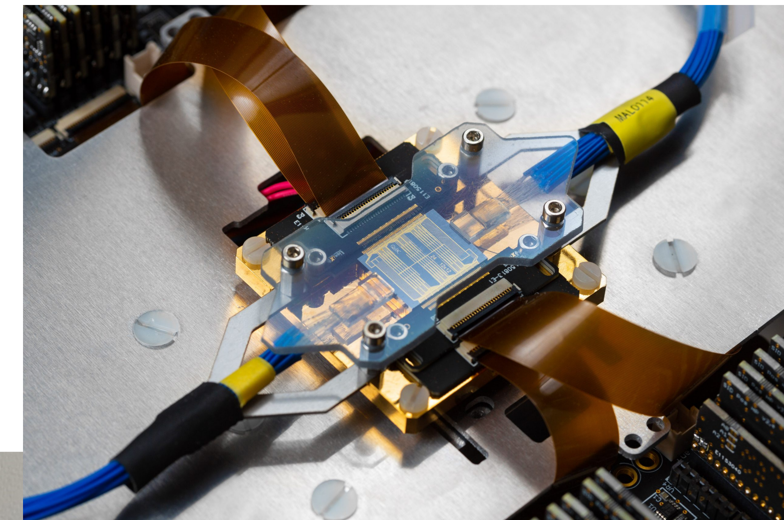
result visualization



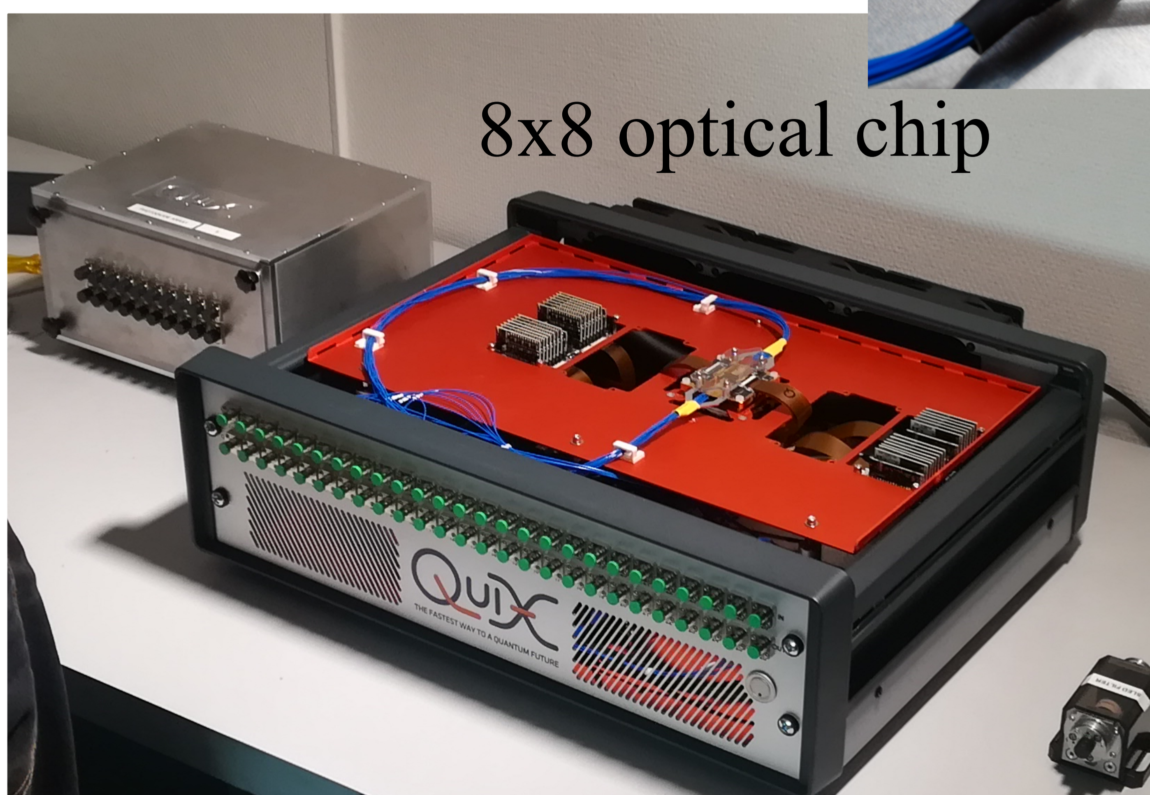
PIQUASSO:
integration with quantum hardware

Quantum Information
National Laboratory
HUNGARY

PIQUASSO.com



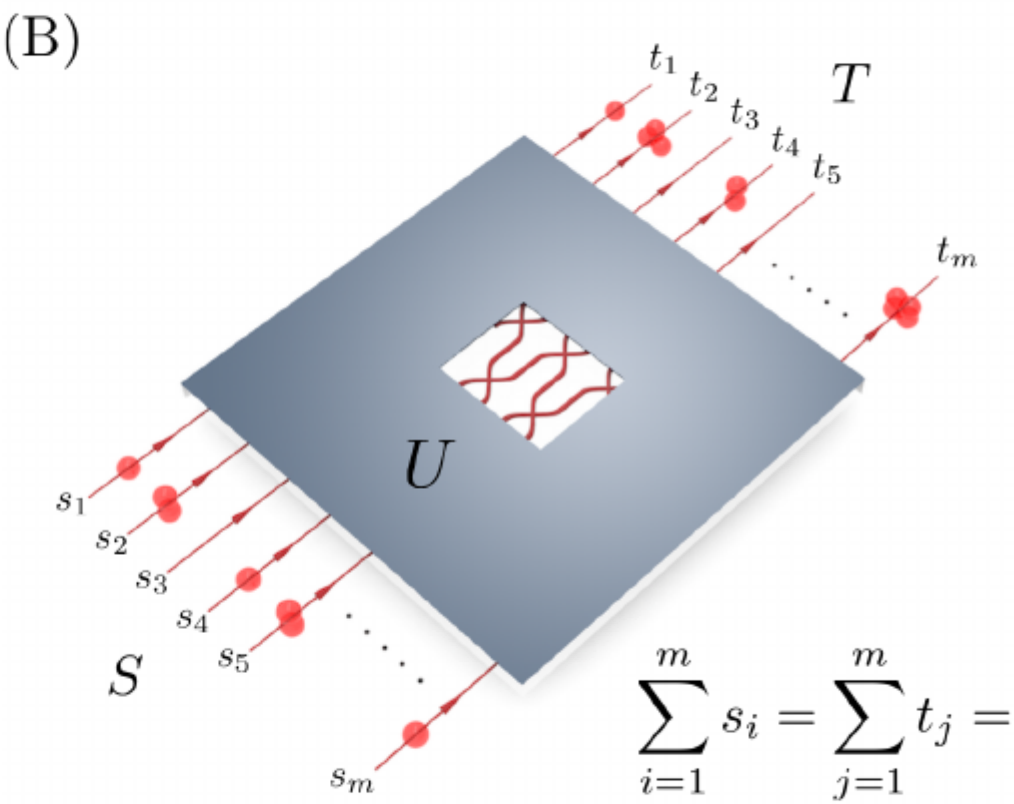
8x8 optical chip



ELTE
EÖTVÖS LORÁND
UNIVERSITY

Gábor Vattay

The problem of boson sampling



$$\sum_{i=1}^{m} s_i = \sum_{j=1}^{m} t_j = n$$

To simulate boson sampling one needs to evaluate:

The **permanent**: $\mathrm{Per}\,(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i\sigma(i)},$

The **torontonian**: $\mathrm{Tor}(\mathbf{A}) = \sum_{Z \in P_N} \frac{(-1)^{N/2-|Z|}}{\sqrt{|\det(\mathbf{I} - \mathbf{A_Z})|}}$

The **hafnian**: $\mathrm{haf}(\boldsymbol{A}) = \sum_{M \in \mathrm{PMP}(n)} \prod_{(i,j) \in M} A_{i,j}$

Each of them having exponential complexity!

tight relation with graph problems:

number of perfect matchings in graphs (bipartite graphs)

Quantum Information
National Laboratory
HUNGARY

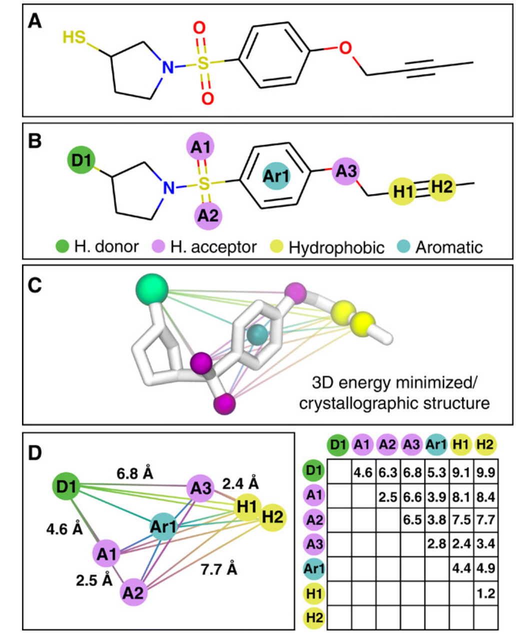
Boson Sampling to solve problems

Quantum Information National Laboratory HUNGARY

Molecular docking with Gaussian Boson Sampling

LEONARDO BANCHI, MARK FINGERHUTH, TOMAS BABEJ, CHRISTOPHER ING, AND, JUAN MIGUEL ARRAZOLA Authors Info & Affiliations

SCIENCE ADVANCES • 5 Jun 2020 • Vol 6, Issue 23 • DOI: 10.1126/sciadv.aax1950



	D1	A1	A2	A3	Ar1	H1	H2
D1		4.6	6.3	6.8	5.3	9.1	9.9
A1			2.5	6.6	3.9	8.1	8.4
A2				6.5	3.8	7.5	7.7
A3					2.8	2.4	3.4
Ar1						4.4	4.9
H1							1.2
H2							

Published: 24 August 2015

Boson sampling for molecular vibronic spectra

Joonsuk Huh, Gian Giacomo Guerreschi, Borja Peropadre, Jarrod R. McClean & Alán Aspuru-Guzik

Nature Photonics 9, 615–620 (2015) | Cite this article

8139 Accesses | 133 Citations | 91 Altmetric | Metrics

Graph isomorphism and Gaussian boson sampling

K. Bradler, S. Friedland, +2 authors D. Su • Published 2018 • Mathematics, Physics • Special Matrices

CERTAIN PROPERTIES AND APPLICATIONS OF SHALLOW BOSONIC CIRCUITS

to solve QUBO problems

arXiv > quant-ph > arXiv:2112.09766

KAMIL BRÁDLER AND HUGO WALLNER

ORCA Computing

WIGNER

Boson Sampling

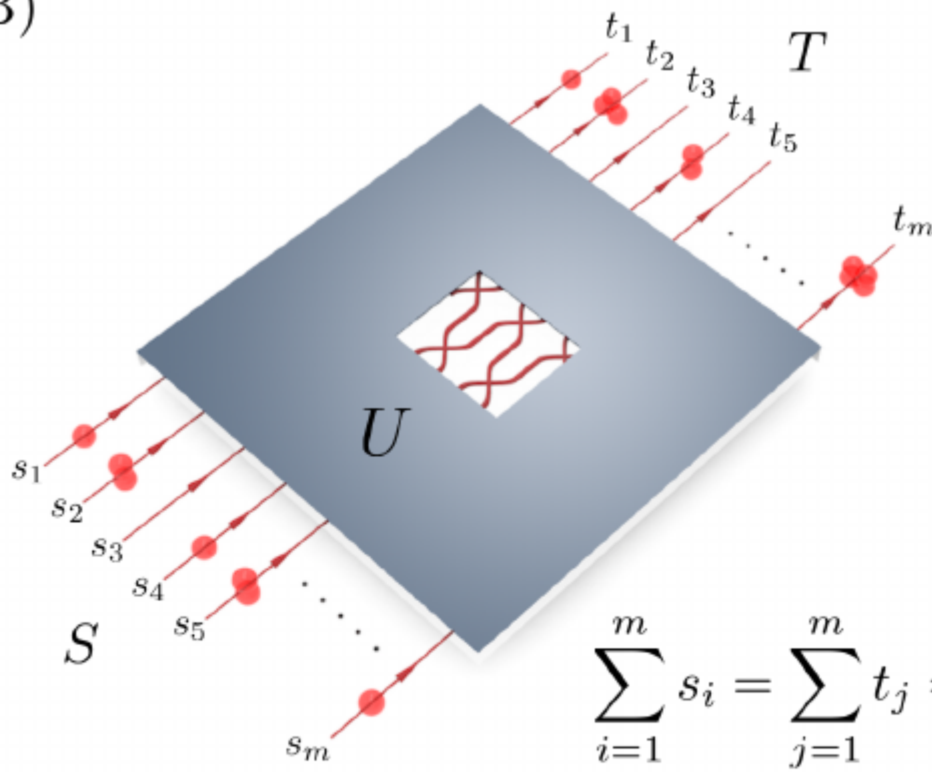
Quantum Information National Laboratory HUNGARY

PIQUASSO

The Computational Complexity of Linear Optics

Scott Aaronson* Alex Arkhipov†

(B)



$$P[S \to T] = \frac{|Per(U_{S,T})|^2}{s_1! \dots s_m! t_1! \dots t_m!}$$

$$\mathrm{Per}\,(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i\sigma(i)},$$

Set of permutations of (1,2,3 .... n)

$\mathcal{O}((n-1) \cdot n!)$

$\mathcal{O}(n^2 \cdot 2^n)$

$$\sum_{i=1}^{m} s_i = \sum_{j=1}^{m} t_j = n$$

Ryser's formula: $$\mathrm{Per}(A) = (-1)^n \sum_{S \subseteq \{1,2,\dots,n\}} (-1)^{|S|} \prod_{i=1}^{n} \sum_{j \in S} a_{ij}$$

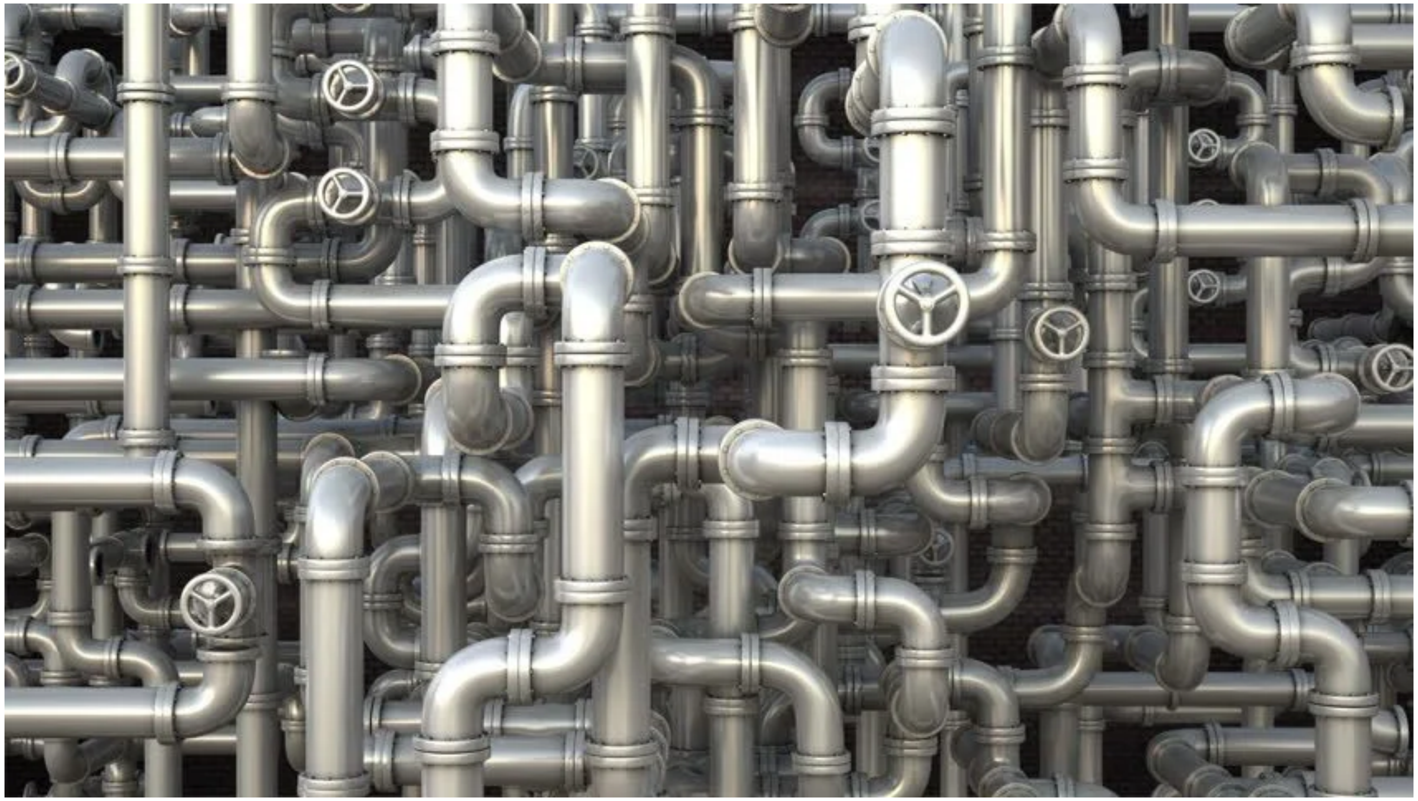
BB/FG's formula: $$\mathrm{Per}(A) = \frac{\sum_\delta \left(\prod_{k=1}^n \delta_k\right) \prod_{i=1}^n \sum_{j=1}^n \delta_j a_{ij}}{2^{n-1}}$$

$\delta = \{\delta_1, \delta_2, \dots, \delta_n\}$ $\delta_1 = 1$ and $\delta_i \in \{-1, 1\}$ for $2 \le i \le n$

FPGA based data-flow engines (DFE's)

Quantum Information National Laboratory HUNGARY

MAXELER Technologies
Maximum Performance Computing



Data streams flowing through the FPGA chip

automatized time and space constraints

FPGA hardware + data-flow programming model = DFE

supported by Xilinx University Program

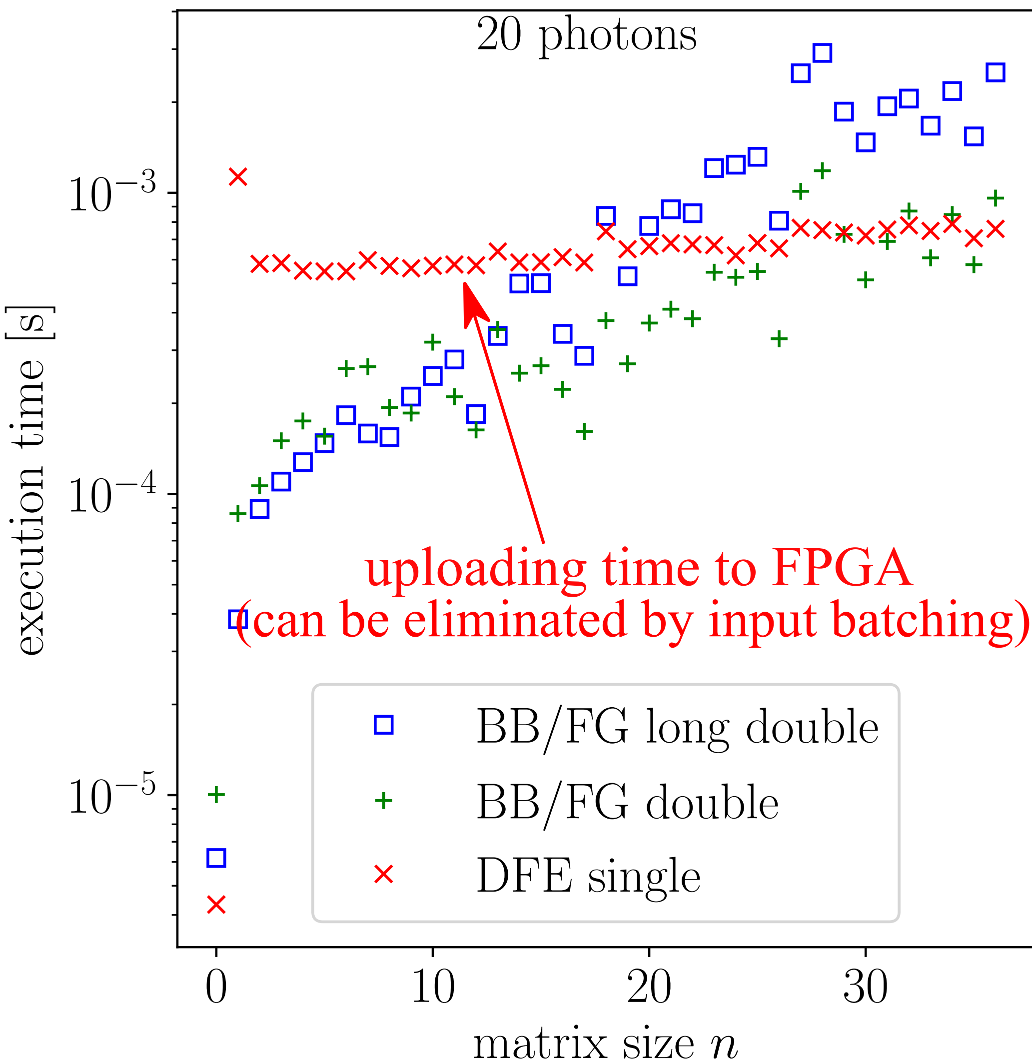
ELTE EÖTVÖS LORÁND UNIVERSITY

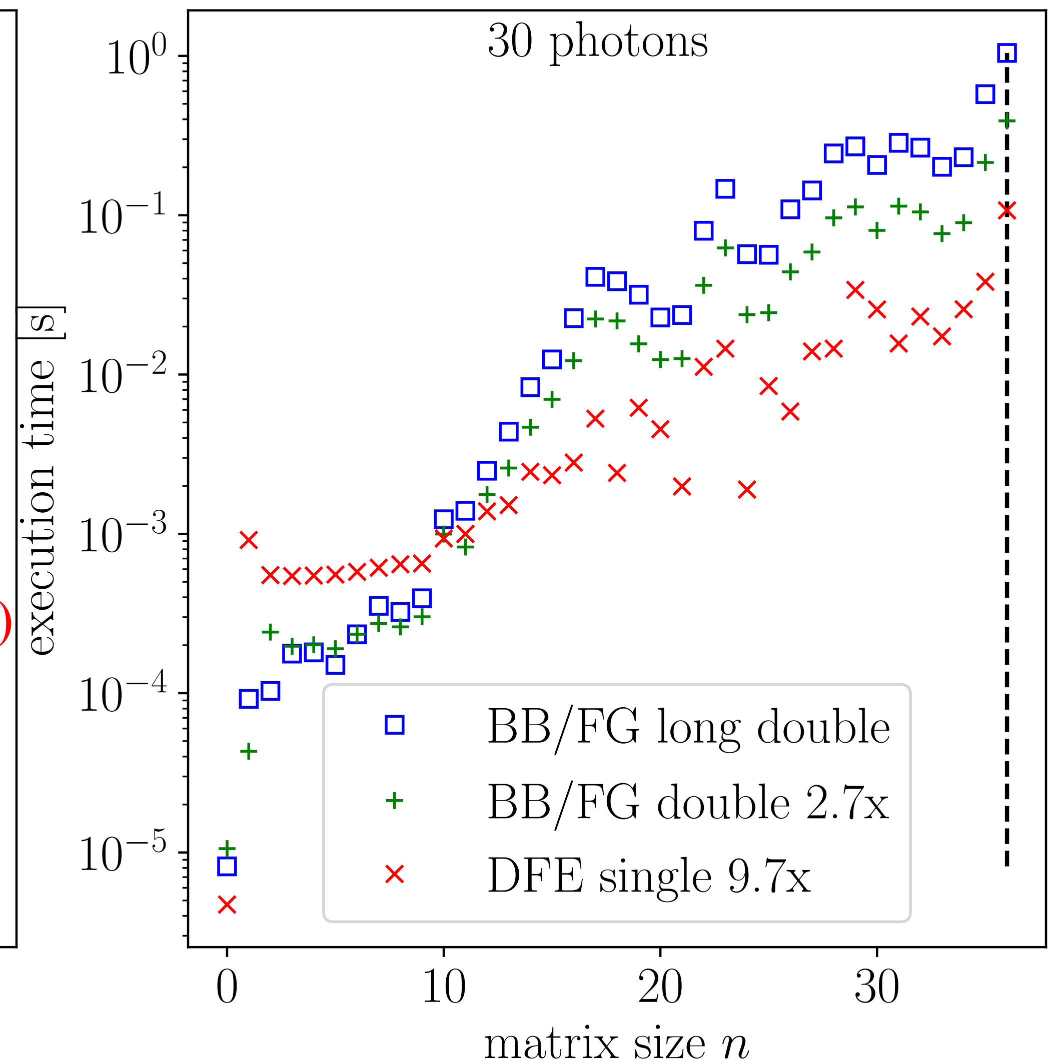
WIGNER

DFE Permanent benchmark

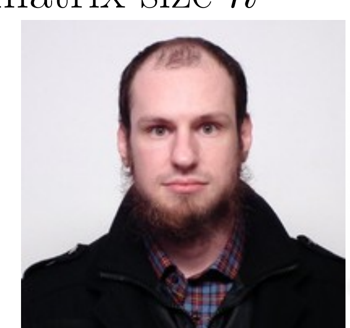
CPU: AMD EPYC 7542 32-Core Processor, 64 threads

Quantum Information
National Laboratory
HUNGARY





ELTE
EÖTVÖS LORÁND
UNIVERSITY



Gregory Morse

FPGA

128 bit fixed point
number representation

MAXELER
Technologies
Maximum Performance Computing

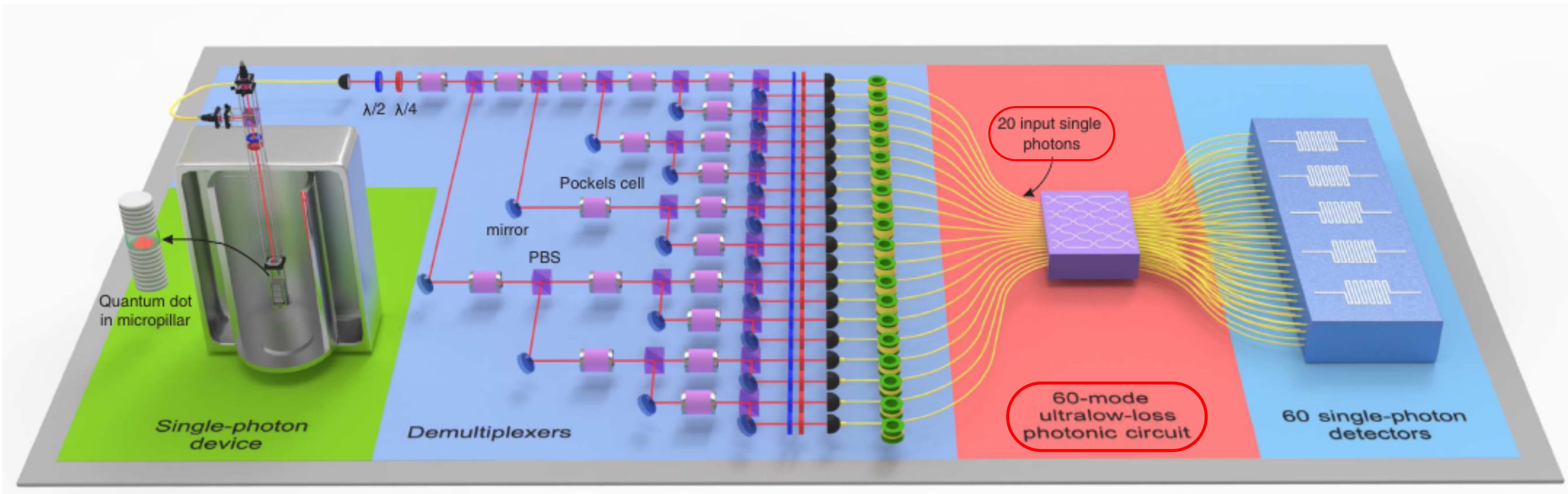
WIGNER

Boson Sampling benchmark

Quantum Information National Laboratory HUNGARY

Experiment

PHYSICAL REVIEW LETTERS **123**, 250503 (2019)

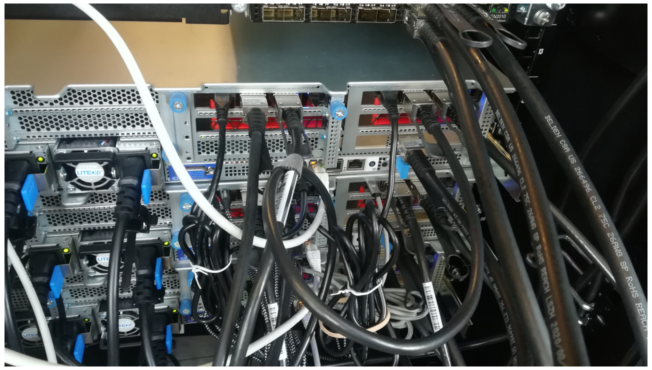


Simulation:

Peter Clifford, Raphaël Clifford: arXiv:1706.01260, arXiv:2005.04214

challenges:

- multiple photons on the output modes
  (reduces permanent calculation complexity)
- port "repeated row" logic to FPGA
- process multiple matrices on FPGA in one shot



ELTE EÖTVÖS LORÁND UNIVERSITY

BS Simulation benchmarks on DFE quantum computer simulator with 60 modes:

20 photons: ~0.01 sec/sample
30 photons: ~1 sec/sample
36 photons: ~40 sec/sample

Particle resolved Gaussian Boson Sampling

Quantum Information National Laboratory HUNGARY

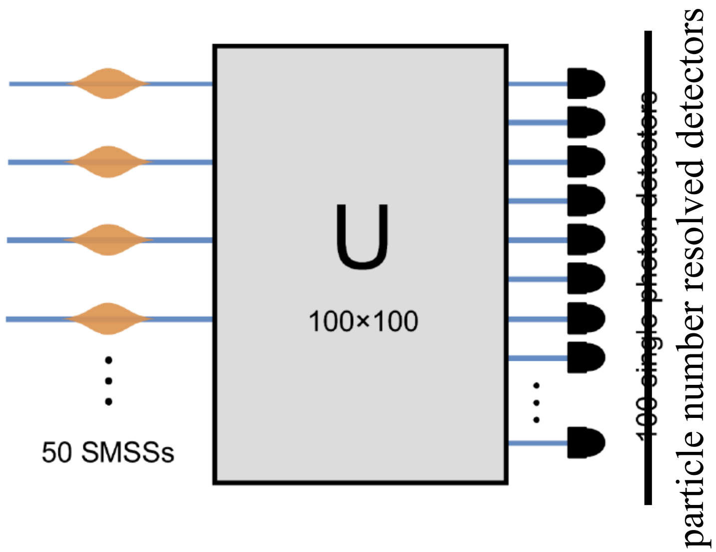
PIQUASSO Boost

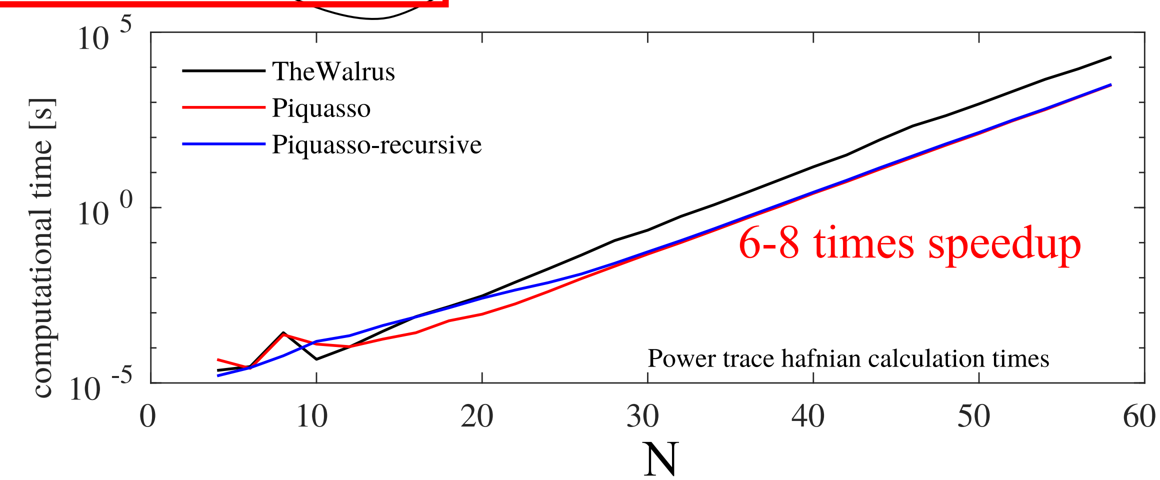
The probability of output S = (s_1,s_2,s_3,s_4,s_5,s_6,s_7,s_8,s_9,s_{10},s_{11} ...)

expectation value of: $\hat{\Pi}^{(n)}_{s_n} = |s_n\rangle\langle s_n|$

$$p(S) = \frac{1}{\sqrt{\det(\boldsymbol{Q})}} \frac{\mathrm{haf}(\boldsymbol{A}_S)}{s_1! \cdots s_m!}$$

Derived from the covariance matrix of the Gaussian state





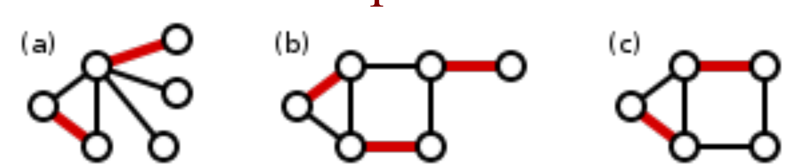
Sampling algorithm again via the chain rule of probability

s_i = 0,1,2,3,4,5,6, ... cut off

The Hafnian function:

$$\mathrm{haf}(\boldsymbol{A}) = \sum_{M \in \mathrm{PMP}(n)} \prod_{(i,j)\in M} A_{i,j}$$

PMP: set of perfect matches



Andreas Björklund, Brajesh Gupt, Nicolás Quesada,
ACM Journal of Experimental Algorithmics Volume 242019 Article No.: 1.11