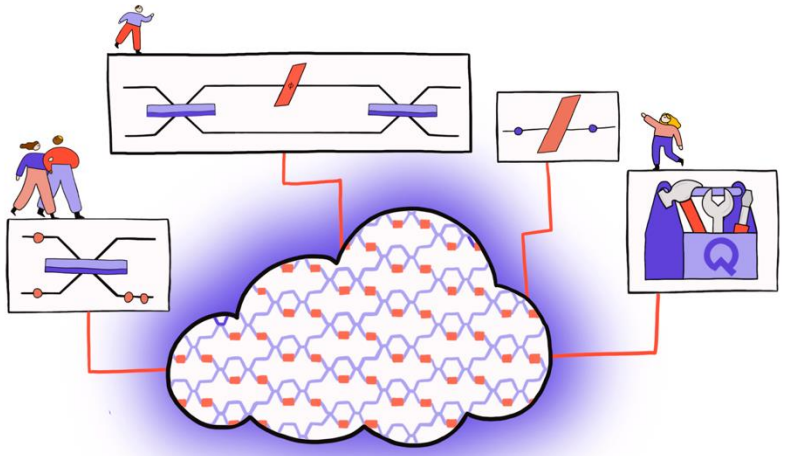

QUANDELA
Driven by quantum,
Empowered by Quandela
www.quandela.com



# What we cover today

## Linear Optics

- Fock States and Fock Space
- Linear Optics Components and Circuits
- Example: HOM effect
- Boson Sampling
- Qubit encoding and gates

## Hardware

- Single Photon Sources
- Noise and Errors
- Example: Ascella

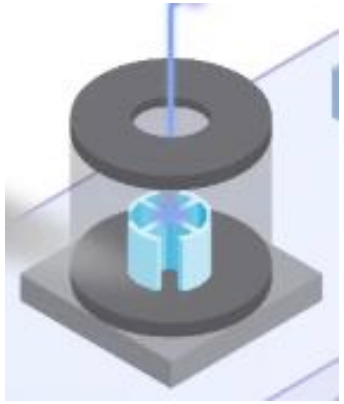
## Software

- Perceval
- Compilation and Transpilation
- Example: Hello World

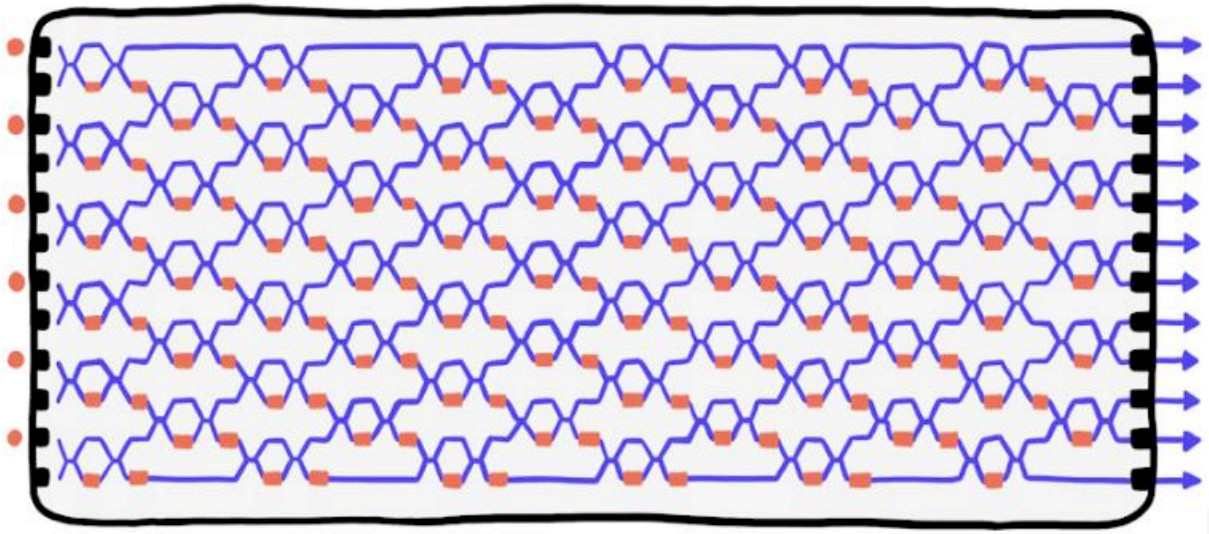
01.

# Linear Optic Quantum Computing

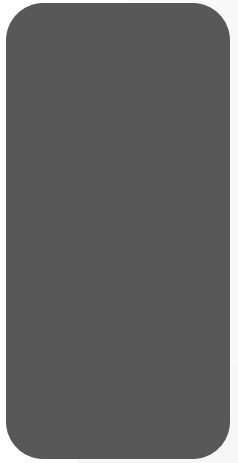
# Manipulating Single Photons using Linear Optics



Single Photon Source (SPS)

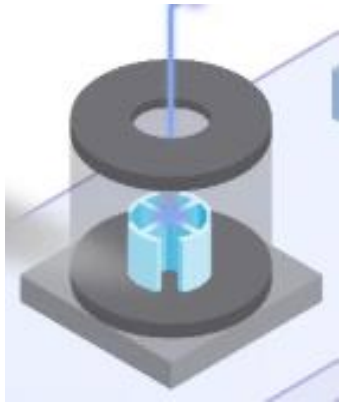


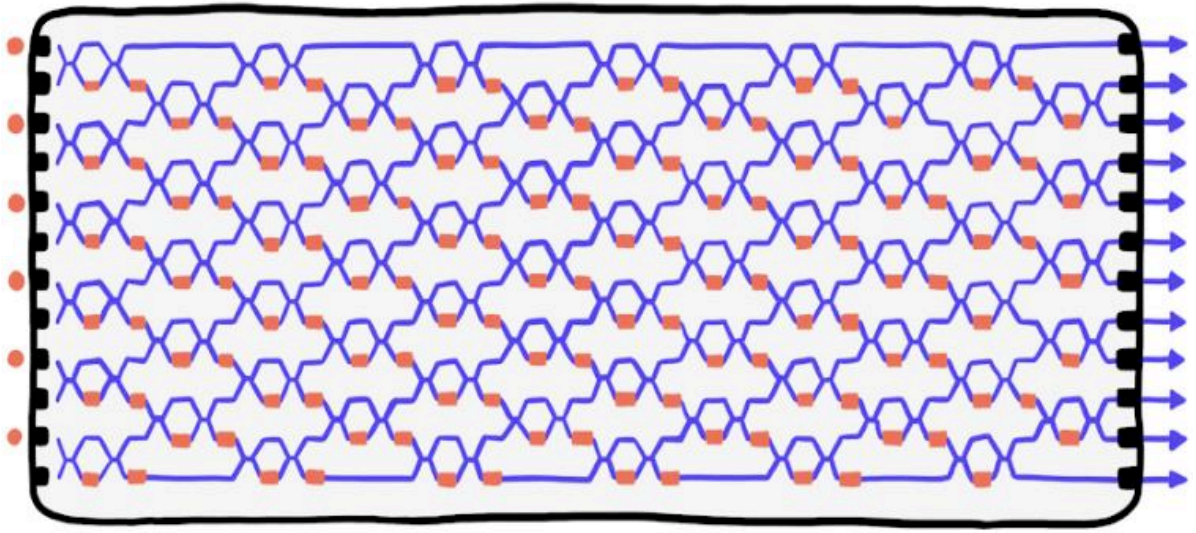
Photonic Integrated Chip (PIC)

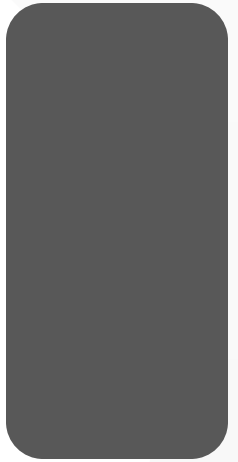


Superconducting nanowire single-photon detectors (SNSPD)

# Manipulating Single Photons using Linear Optics







**Single Photon Source (SPS)**

Generate a stream of single photons

**Photonic Integrated Chip (PIC)**

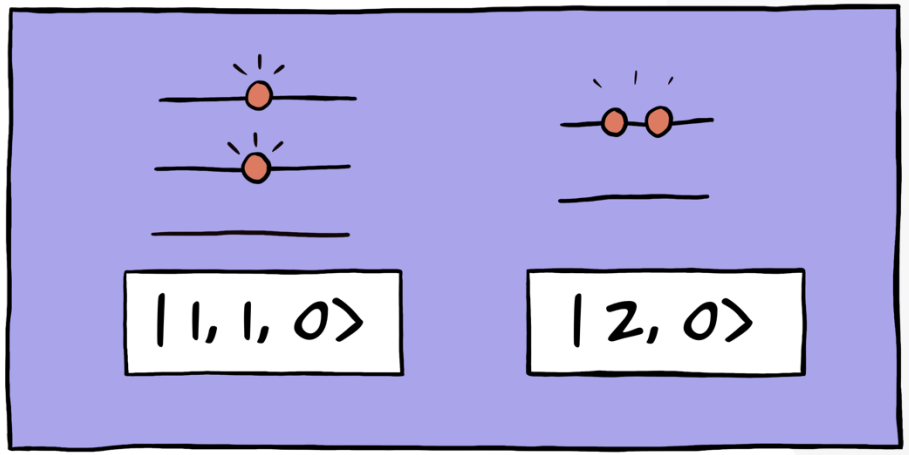
Insert multiple photons into the chip

**Superconducting nanowire single-photon detectors (SNSPD)**

Detect where the photons exit the chip

# Fock States

A **Fock state** is a specific configuration of **n** photons in **m** modes.




Two different Fock states with 2 photons and 3 and 2 modes respectively.

## Test your understanding

1 Go to wooclap.com

2 Enter the event code in the top banner

Event code
HRDMYU

# Fock Space

| # photons (n) | # optical modes (m) | Fock Space Dimension |
|---|---|---|
| 2 | 2 | 3 |
| 6 | 12 | |
| 12 | 24 | |
| 24 | 48 | |

# Fock Space

| # photons (n) | # optical modes (m) | Fock Space Dimension |
|---|---|---|
| 2 | 2 | 3 |
| 6 | 12 | 12.376 |
| 12 | 24 | |
| 24 | 48 | |

# Fock Space

| # photons (n) | # optical modes (m) | Fock Space Dimension |
|---|---|---|
| 2 | 2 | 3 |
| 6 | 12 | 12.376 |
| 12 | 24 | 834.451.800 |
| 24 | 48 | |

# Fock Space

| # photons (n) | # optical modes (m) | Fock Space Dimension |
|---|---|---|
| 2 | 2 | 3 |
| 6 | 12 | 12.376 |
| 12 | 24 | 834.451.800 |
| 24 | 48 | $5{,}3 \times 10^{18}$ |

$$\Phi_{Fock} = \binom{n+m-1}{n}$$

# Fock Space

| # photons (n) | # optical modes (m) | Fock Space Dimension |
|---|---|---|
| 2 | 2 | 3 |
| 6 | 12 | 12.376 |
| 12 | 24 | 834.451.800 |
| 24 | 48 | $5{,}3 \times 10^{18}$ |

$$\Phi_{Fock} = \binom{n+m-1}{n}$$

? ? ? ? ? ?

# Fock Space

| # photons (n) | # optical modes (m) | Fock Space Dimension |
|---|---|---|
| 2 | 2 | 3 |
| 6 | 12 | 12.376 |
| 12 | 24 | 834.451.800 |
| 24 | 48 | $5{,}3 \times 10^{18}$ |

$$\Phi_{Fock} = \binom{n+m-1}{n}$$

?  ?   ?

# Fock Space

| # photons (n) | # modes (m) | Fock Space Dimension |
|---|---|---|
| 2 | 2 | 3 |
| 6 | 12 | 12.376 |
| 12 | 24 | 834.451.800 |
| 24 | 48 | $5{,}3 \times 10^{18}$ |

$$\Phi_{Fock} = \binom{n+m-1}{n}$$

# Fock Space

| # photons (n) | # optical modes (m) | Fock Space Dimension |
|---|---|---|
| 2 | 2 | 3 |
| 6 | 12 | 12.376 |
| 12 | 24 | 834.451.800 |
| 24 | 48 | $5{,}3 \times 10^{18}$ |

$$\Phi_{Fock} = \binom{n+m-1}{n}$$

$$| \star | \star \star | = |0,1,2,0\rangle$$

# Fock Space

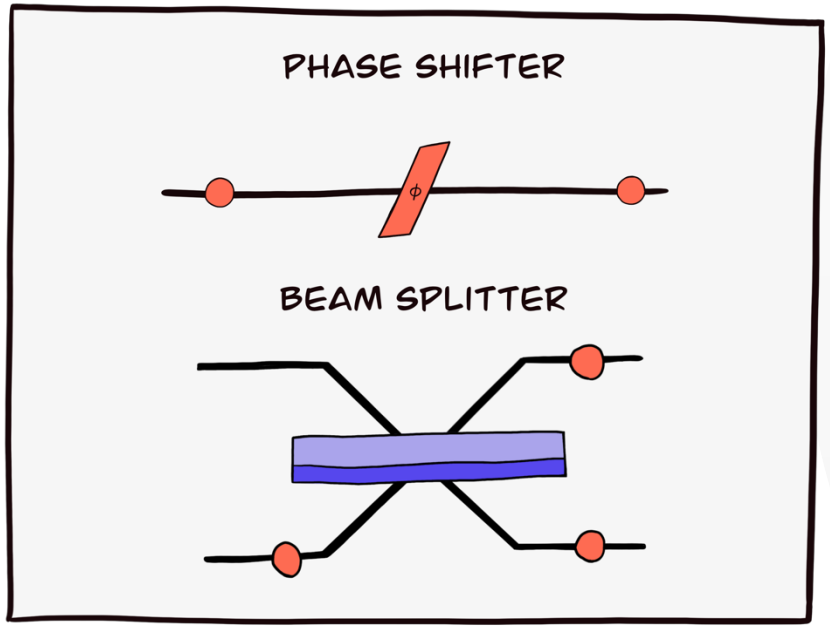
| # photons/qubits (n) | # optical modes (m) | Fock Space Dimension |
|---|---|---|
| 2 | 2 | 3 |
| 6 | 12 | 12.376 |
| 12 | 24 | 834.451.800 |
| 24 | 48 | $5{,}3 \times 10^{18}$ |

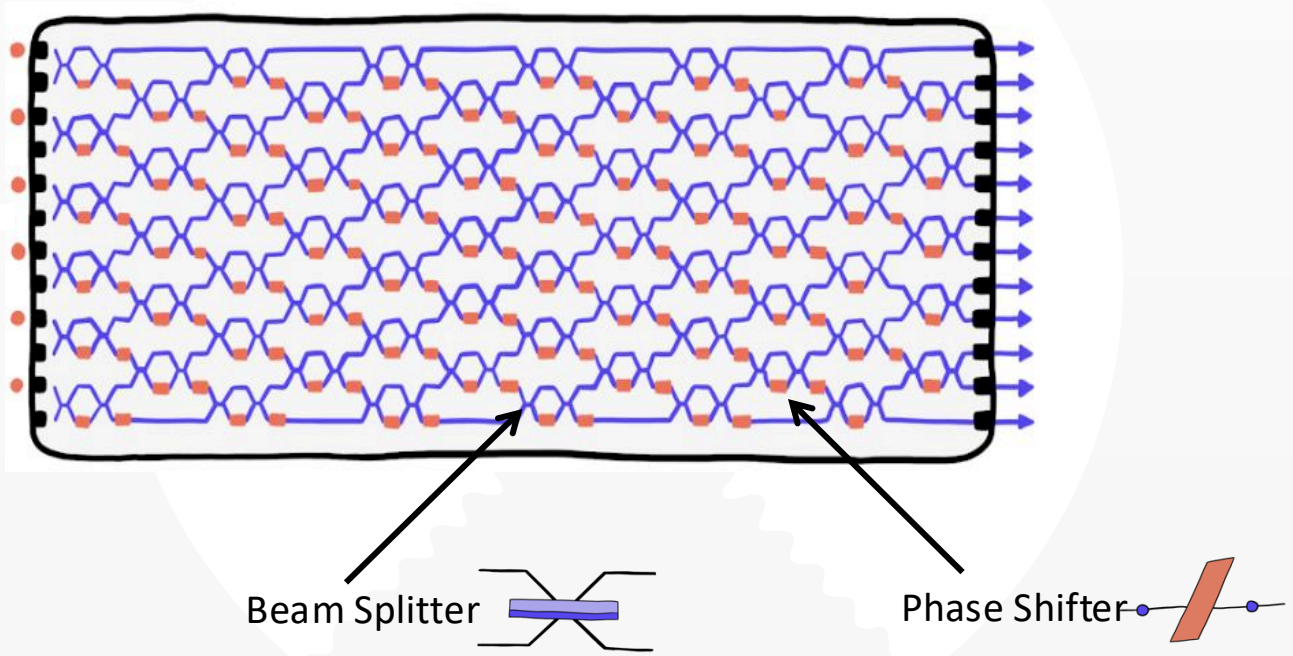
| Qubit Space Dimension |
|---|
| 4 |
| 64 |
| 4.096 |
| 16.77.216 |

$$\Phi_{Fock} = \binom{n+m-1}{n}$$

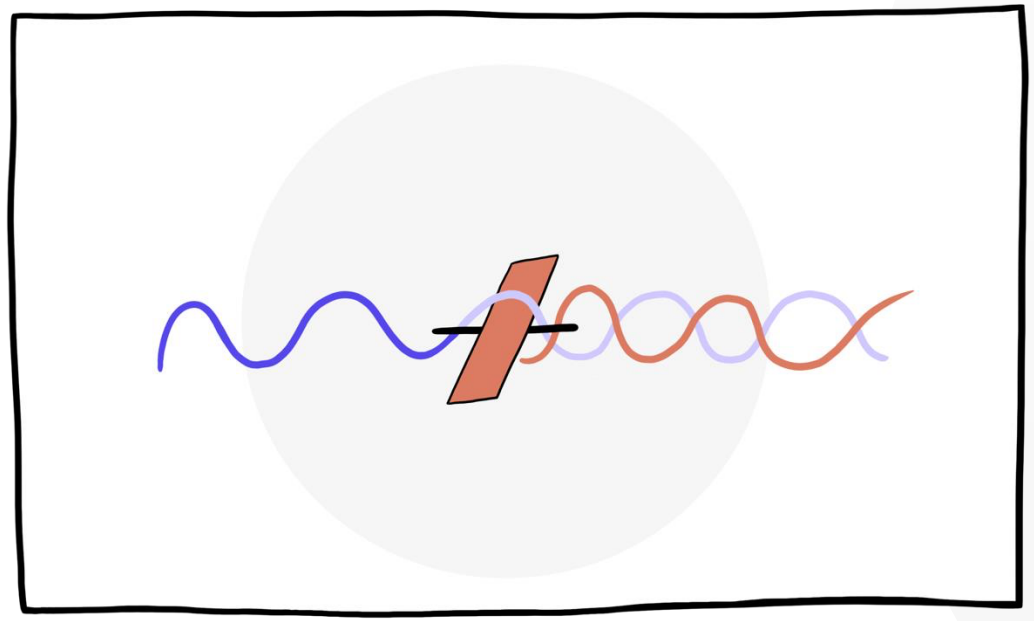
$$\Phi_{Qubit} = 2^n$$

# Linear Optical Components and Circuits



# Linear Optical Components

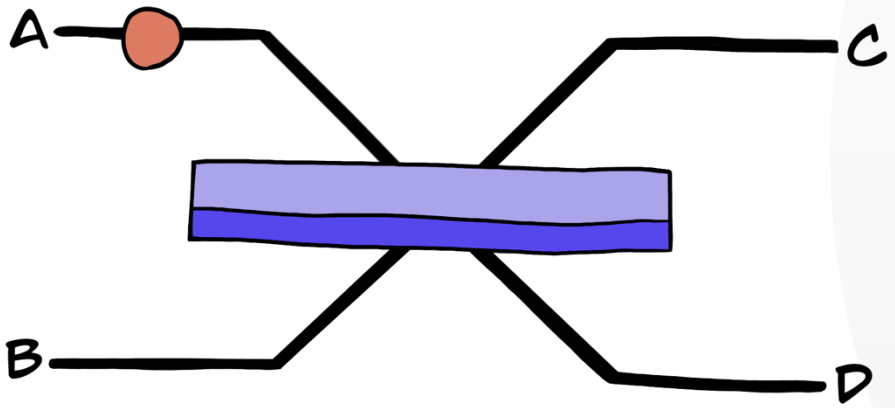


The **Phase Shifter** changes the phase of the photon.

Matrix representation:

$$PS(\phi) = e^{i\phi} \quad \text{with } \phi \in [0,2\pi]$$

# Linear Optical Components



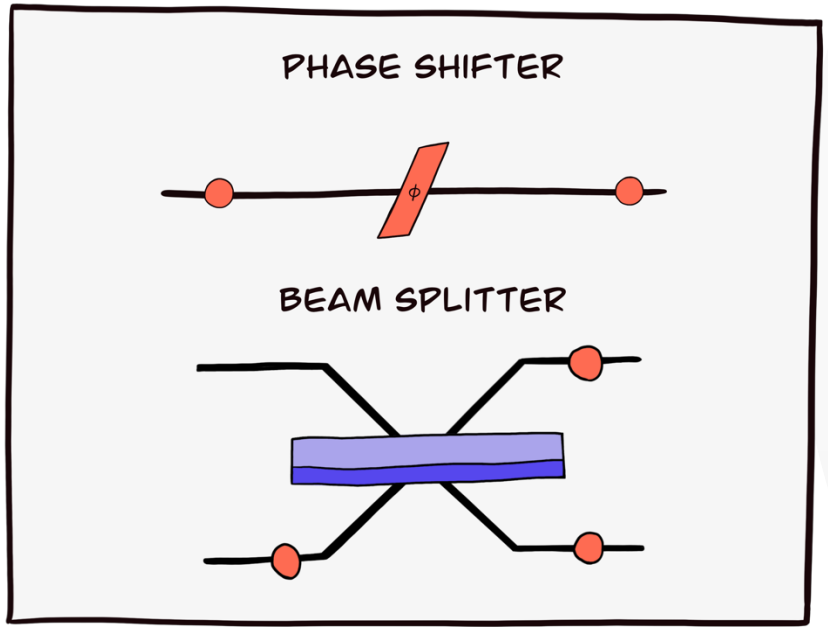

The **Beam Splitter** allows two photons to interact.

Matrix representation of a balanced beam splitter:

$$\mathrm{BS} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} \hat{a}^\dagger \\ \hat{b}^\dagger \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} \hat{a}^\dagger + \hat{b}^\dagger \\ \hat{a}^\dagger - \hat{b}^\dagger \end{pmatrix}$$

# Linear Optical Components

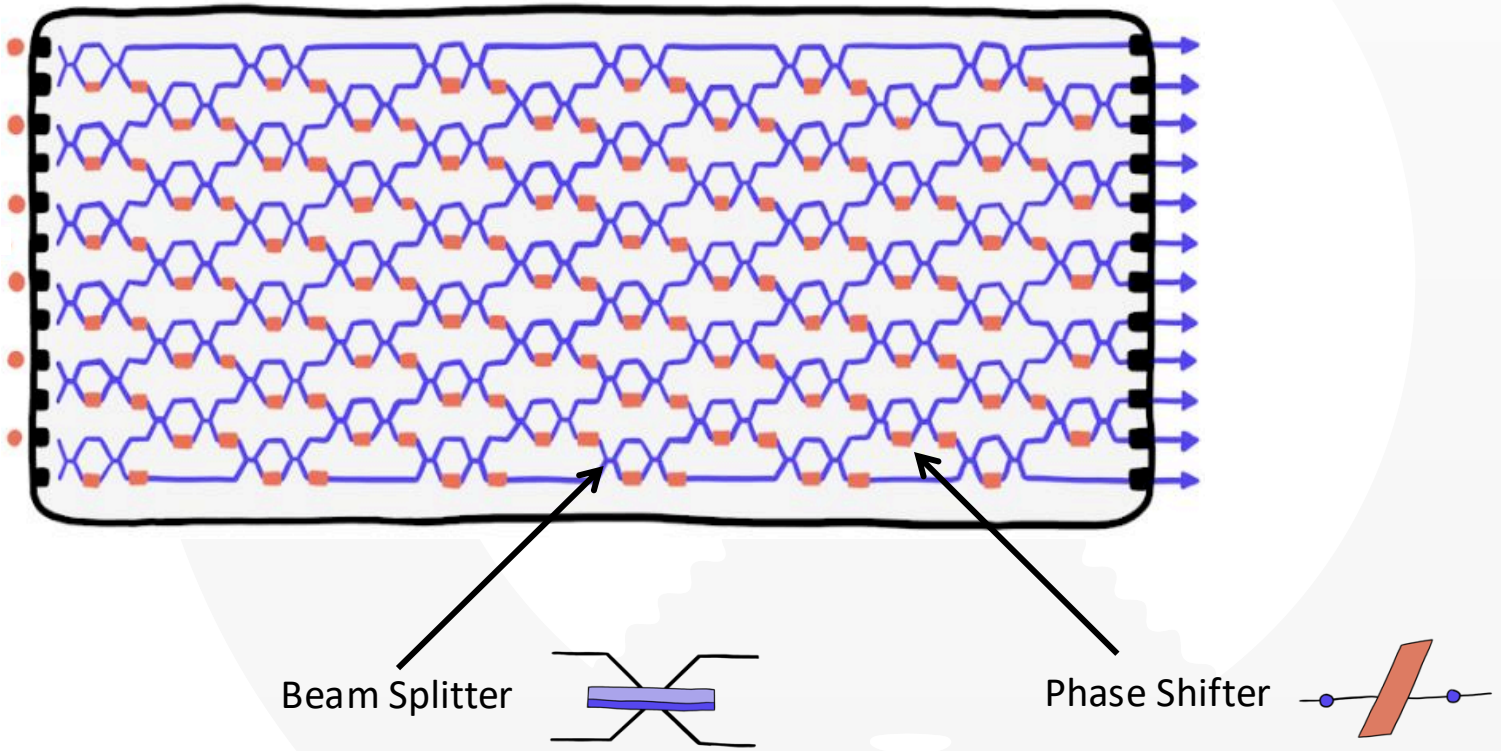



Using only **beam splitters** and **phase shifters** any **unitary transformation** can be achieved.

# Universal Linear Optic Interferometer

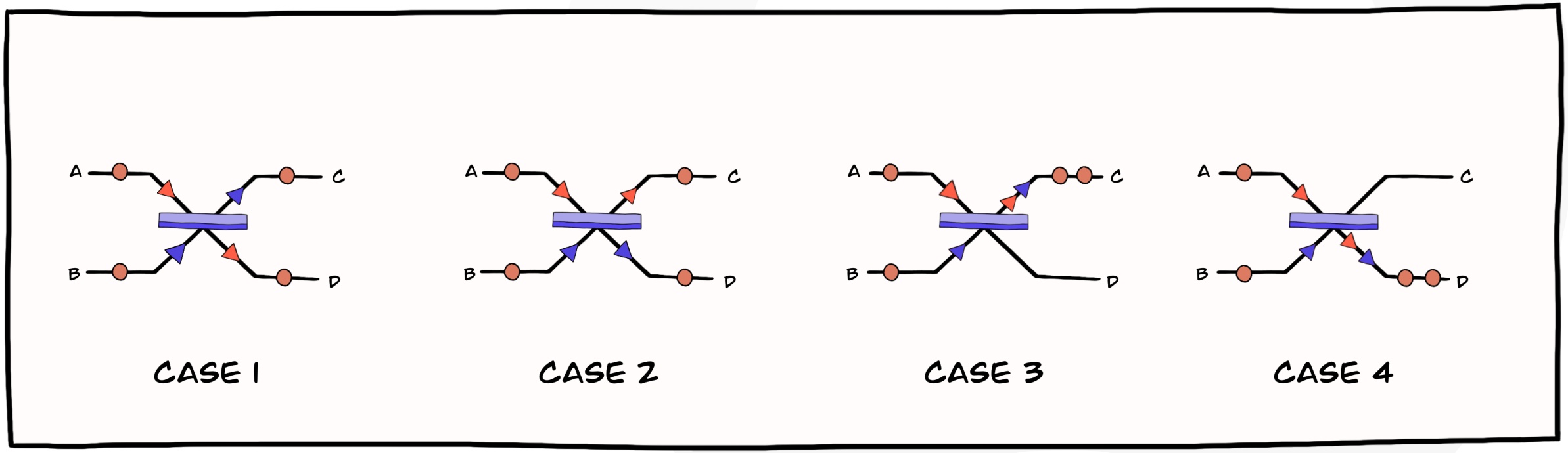
Any linear optical circuit can be executed by the universal interferometer.

Only the phase parameter of the phase shifters have to be adjusted.



# Hong-Ou-Mandel (HOM) effect

Two photons passing through a balanced beam splitter.



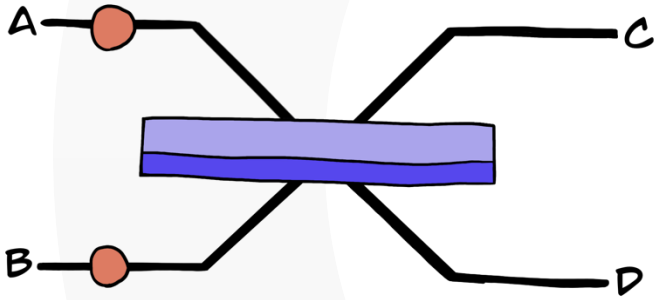
# Hong-Ou-Mandel (HOM) effect

Input Fock State

Balanced Beam Splitter

Output Fock State

$$\hat{a}^\dagger \hat{b}^\dagger |0_a, 0_b\rangle = |1_a, 1_b\rangle$$




$$\text{BS} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

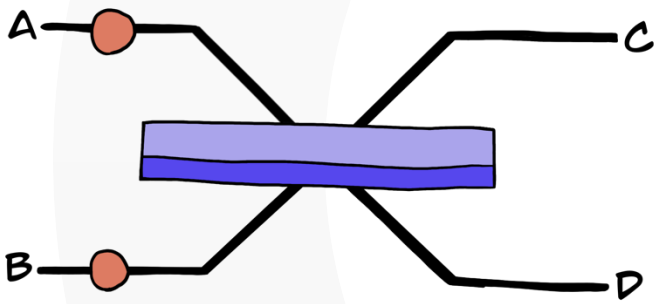
$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} \hat{a}^\dagger \\ \hat{b}^\dagger \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} \hat{a}^\dagger + \hat{b}^\dagger \\ \hat{a}^\dagger - \hat{b}^\dagger \end{pmatrix}$$

# Hong-Ou-Mandel (HOM) effect

Input Fock State

$$\hat{a}^\dagger \hat{b}^\dagger |0_a, 0_b\rangle = |1_a, 1_b\rangle$$

Balanced Beam Splitter




$$\text{BS} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} \hat{a}^\dagger \\ \hat{b}^\dagger \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} \hat{a}^\dagger + \hat{b}^\dagger \\ \hat{a}^\dagger - \hat{b}^\dagger \end{pmatrix}$$

$$\Rightarrow \begin{matrix} \hat{a}^\dagger \to \left(\frac{1}{\sqrt{2}}\hat{c}^\dagger + \hat{d}^\dagger\right) \\ \hat{b}^\dagger \to \left(\frac{1}{\sqrt{2}}\hat{c}^\dagger - \hat{d}^\dagger\right) \end{matrix}$$

Output Fock State

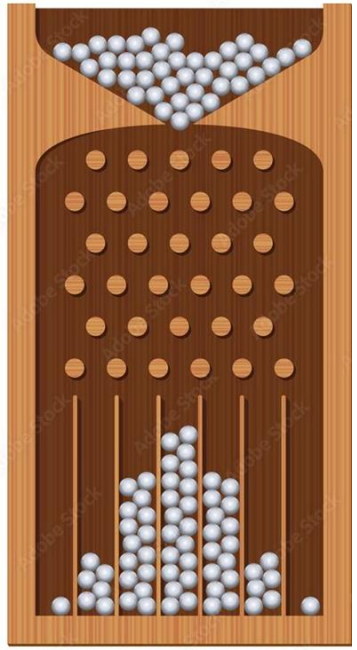
$$\frac{1}{\sqrt{2}}(|1_c\rangle + |1_d\rangle)(|1_c\rangle - |1_d\rangle)$$

$$= \frac{1}{\sqrt{2}}(|1_c\rangle|1_c\rangle - |1_c\rangle|1_d\rangle + |1_d\rangle|1_c\rangle - |1_d\rangle|1_d\rangle)$$

$$= \frac{1}{\sqrt{2}}(|1_c\rangle|1_c\rangle - |1_d\rangle|1_d\rangle)$$

$$= \frac{1}{\sqrt{2}}(|2_c, 0\rangle - |0, 2_d\rangle)$$

# Boson Sampling



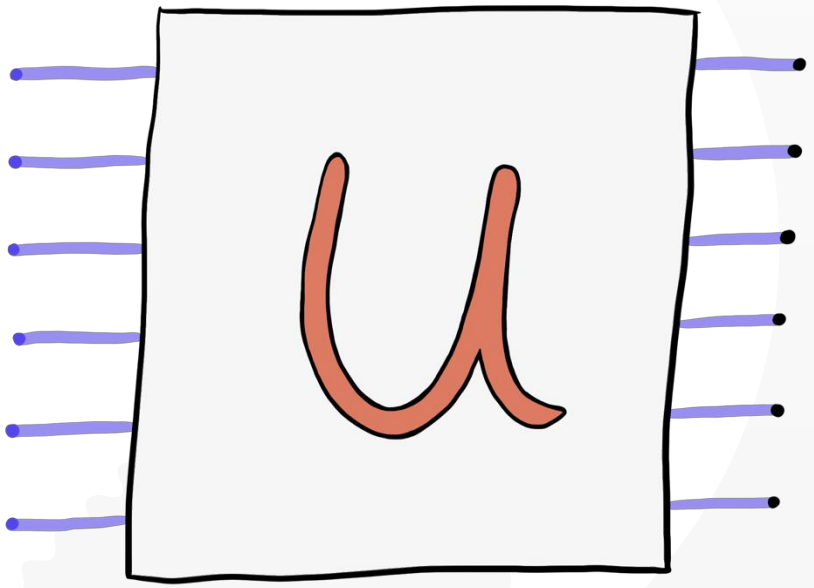
**Boson Sampling** is a quantum computational model in which **indistinguishable photons** pass through a **linear optical interferometer**, and their output distribution—governed by the **permanent** of a unitary matrix—is sampled.

# The permanent of a Matrix

$$per(U) = \sum_{\sigma \in S(n)} \prod_{i=1}^{n} a_{i,\sigma(i)}$$

$$U = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

$$per(U) = aei + ahf + dbi + dhc + gbf + gec$$



The classical limit for simulating boson sampling is around 40 photons.

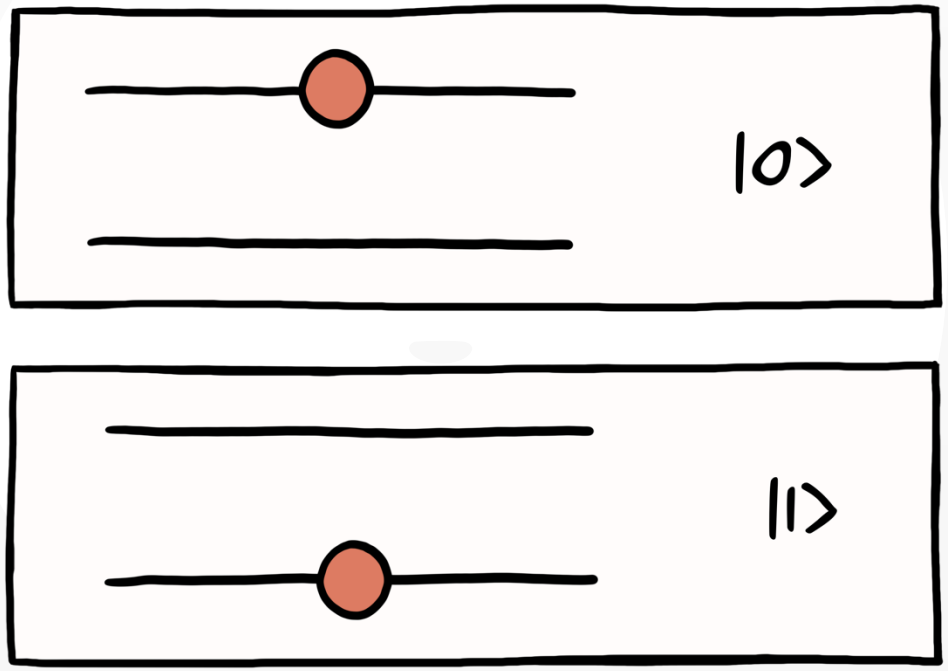
# Qubit encoding

**Dual rail encoding:**

**One qubit** is in one photon and two modes encoded.

**Qudit encoding:**

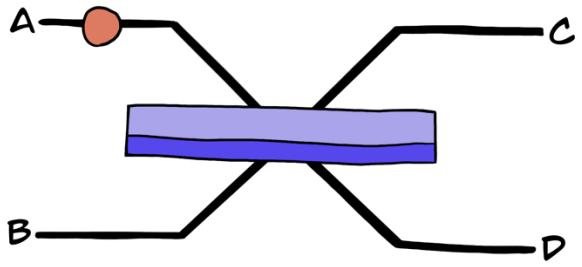
**n** qubits are in **one** photon and $2^n$ modes encoded.



# Qubit gates

## Deterministic 1-qubit gates

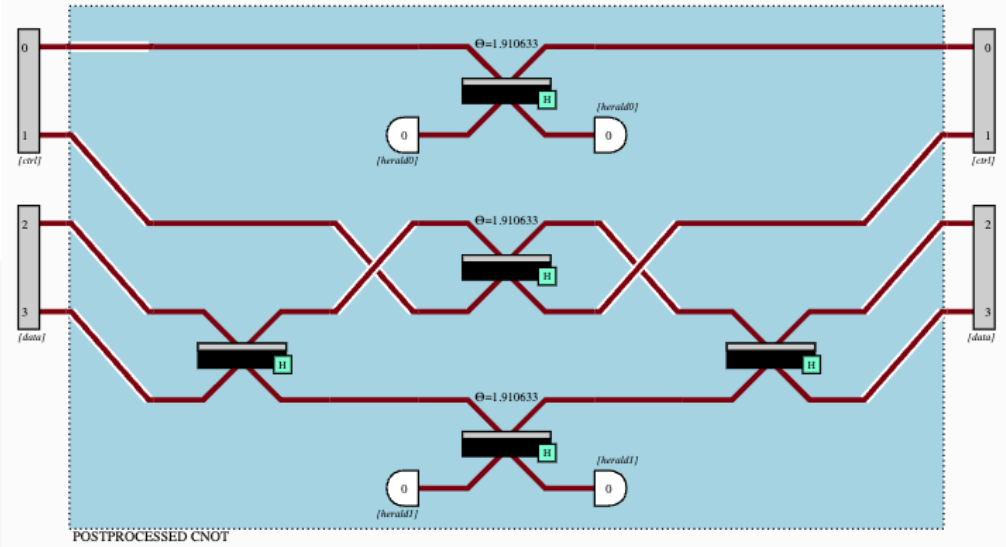
Hadamard Gate




$$|0\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

## Probabilistic 2-qubit gates

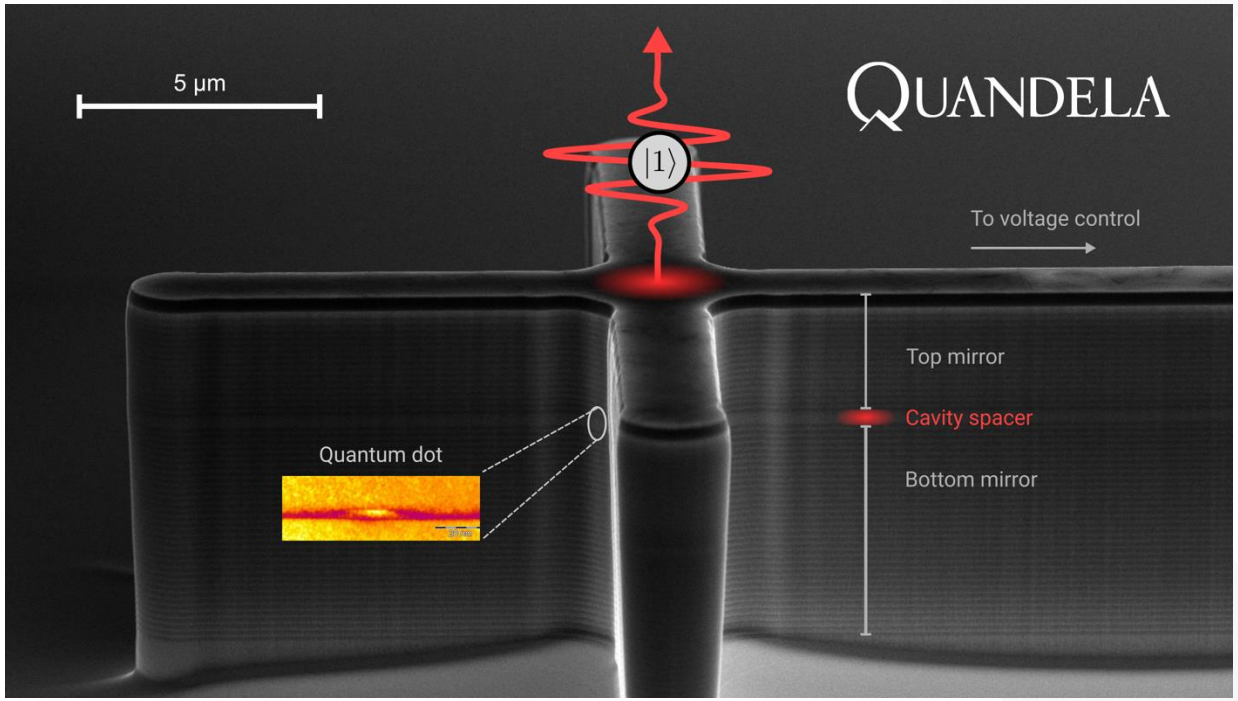
CNOT Gate
Success probability = $^1/_9$



02.

# Hardware

# Generating Single Photons



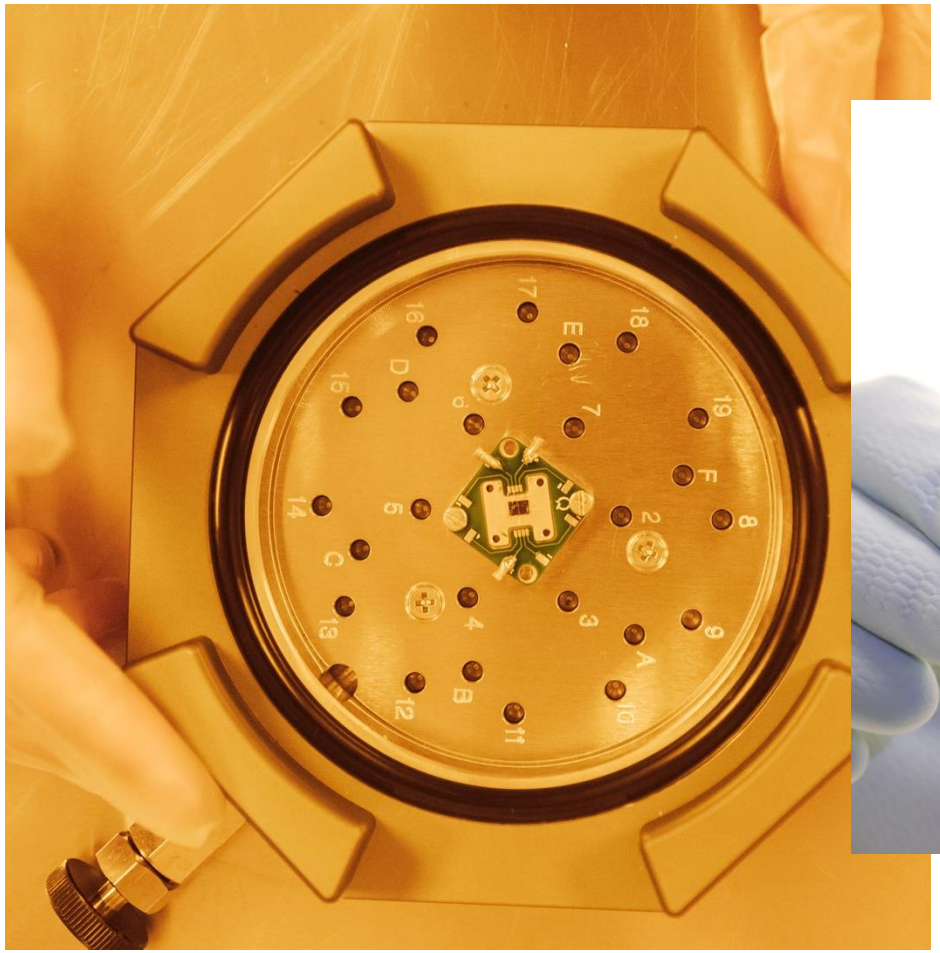

Quandela's single photon source is a **Quantum dot**

The founding technology of Quandela

Only deterministic single photon source

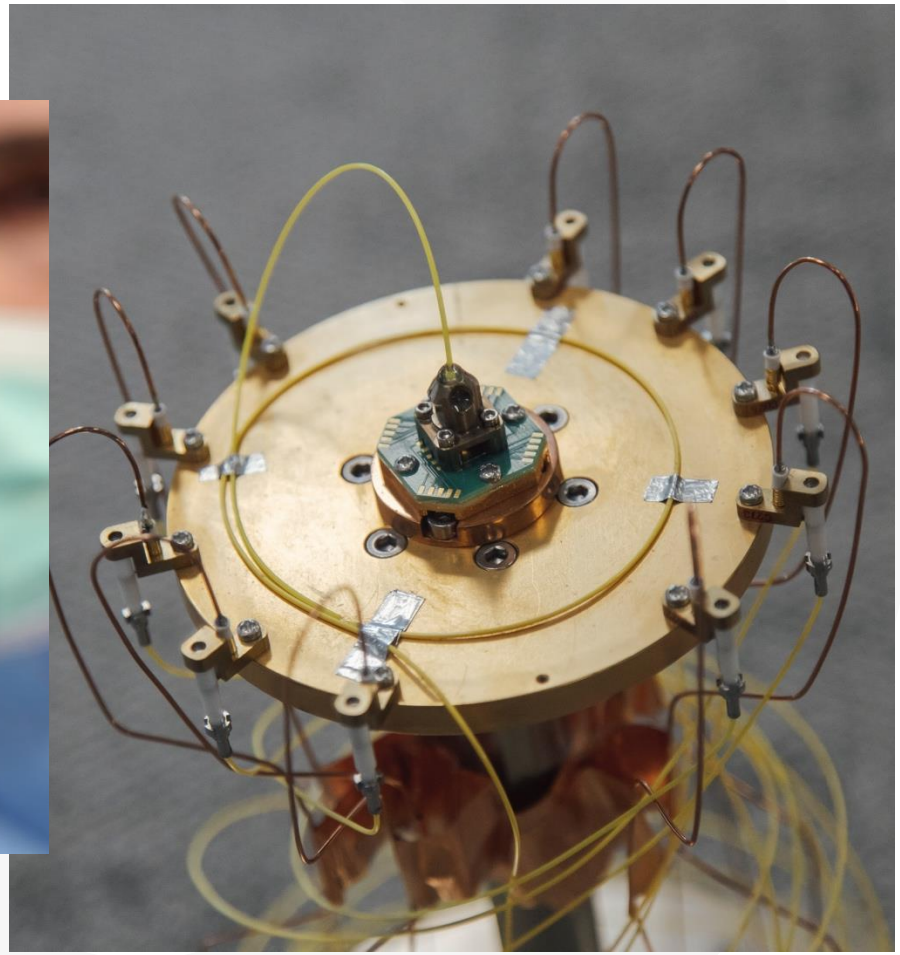
Generates high quality photons

# The Quantum dot



The single photon source in the lab

The single photon source connected to an optical fibre



The single photon source connected to the cryostat

# Generating Single Photons - brightness

The ideal case:

In reality:

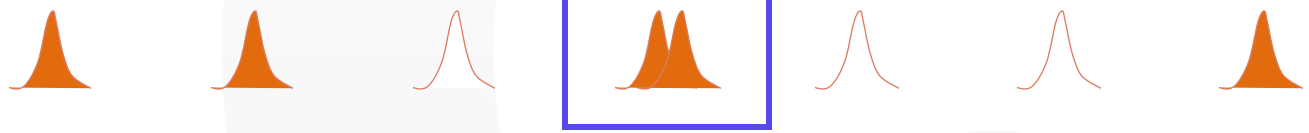
The **brightness** of the single photon source is the ratio of photons generated over the clock speed.

# Generating Single Photons - purity

The ideal case:

Two photons generated

In reality:



The **purity** of the single photon source is the probability of generating more then one photon.

State of the Art: 99%

# Generating Single Photons - indistinguishability

The ideal case:

Distinguishable photons generated

In reality:

The **indistinguishability** of the single photon source measures how often two generated photons are indistinguishable.

State of the Art: 94%

# Noise and Errors in the whole System

## Transmittance

The transmittance is defined as the percentage of **emitted photons** that are **detected**.

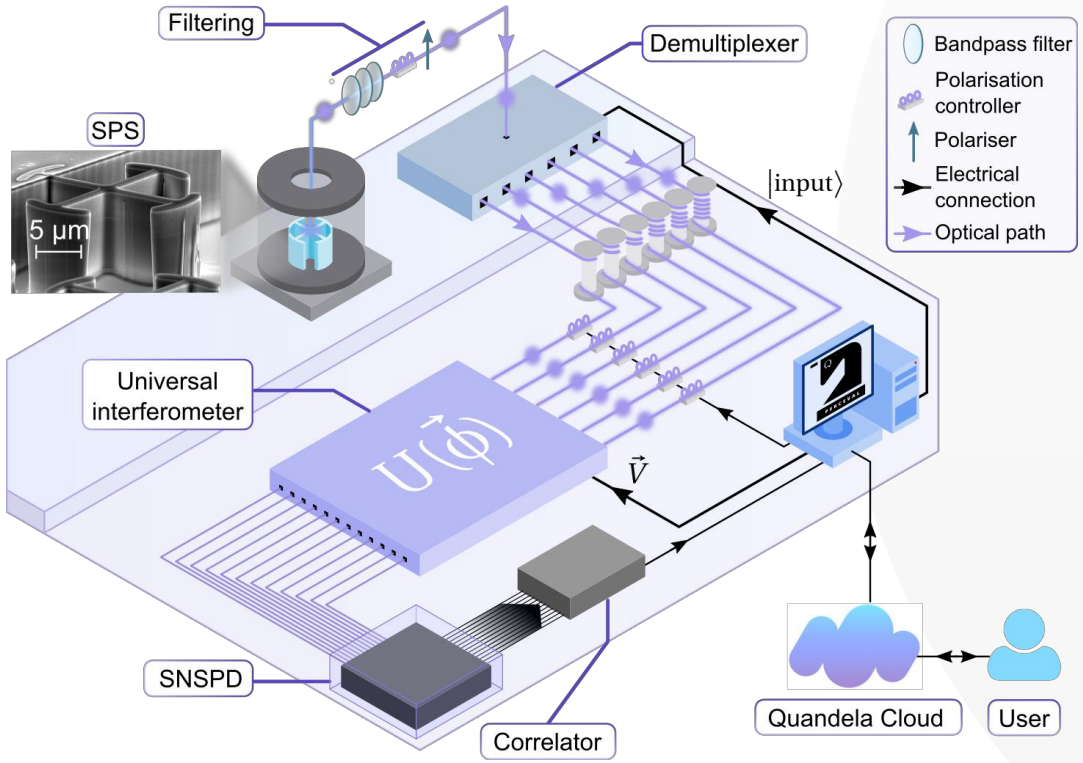
State of the Art: 8%

## Phase Imprecision

Each **phase shifter** varies slightly from the desired value.

State of the Art: 50 mrad

# Example - Ascella




A versatile single-photon-based quantum computing platform (2024)

03.

# Software

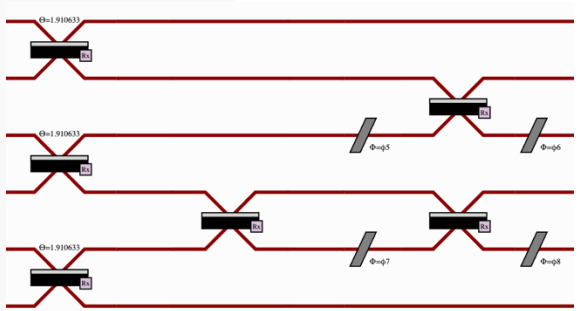
# Perceval

## Simulation Backends in C++

**Strong simulation:**
Calculating the whole probability distribution. (SLOS)

**Weak simulation:**
Generating samples of the probability distribution. (Clifford)

## Circuit building



**Noise modelling:**
Simulates the presence of noise such as photon loss, indistinguishability, etc…

## Cloud access

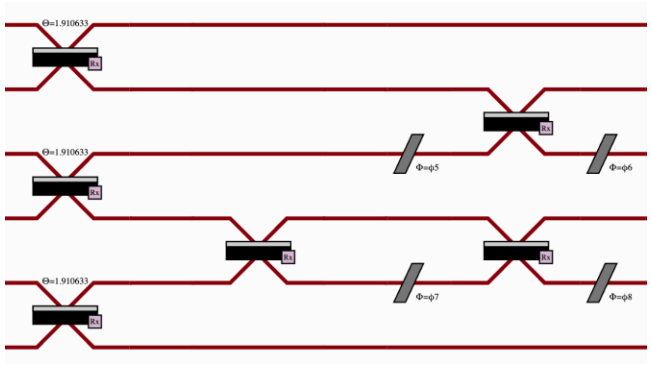
**QPU access:**
Runs your circuit on real hardware.

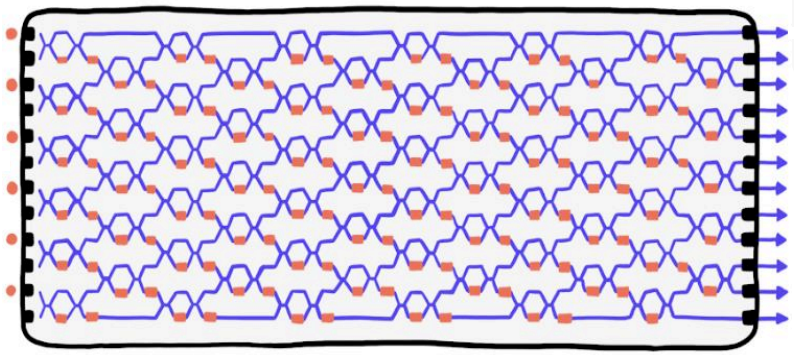
**GPU access:**
Allows for larger simulations.

**Quantum Toolbox:**
Pre-programmed algorithms ready to use.

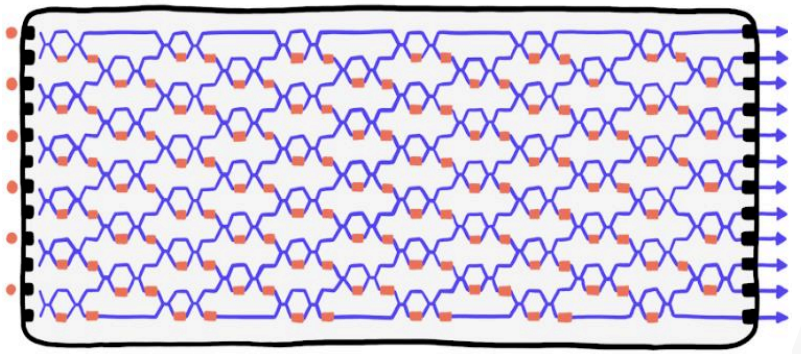
# From Perceval to QPU

Compilation





Transpilation



# Perceval “Hello world”

```
import perceval as pcvl
```

# Perceval “Hello world”

Create an input Fock state

```
import perceval as pcvl

input_state = pcvl.BasicState("|1,1>")
```

# Perceval “Hello world”

Create an input Fock state

Define your circuit

```
import perceval as pcvl

input_state = pcvl.BasicState("|1,1>")

circuit = pcvl.Circuit(2) // pcvl.BS()
```

# Perceval “Hello world”

Create an input Fock state

Define your circuit

Select a platform

```
import perceval as pcvl

input_state = pcvl.BasicState("|1,1>")

circuit = pcvl.Circuit(2) // pcvl.BS()

processor = pcvl.Processor("SLOS")
processor.set_circuit(circuit)
processor.min_detected_photons_filter(2)
processor.with_input(input_state)
```

# Perceval “Hello world”

Create an input Fock state

Define your circuit

Select a platform

Get your results

```
import perceval as pcvl

input_state = pcvl.BasicState("|1,1>")

circuit = pcvl.Circuit(2) // pcvl.BS()

processor = pcvl.Processor("SLOS")
processor.set_circuit(circuit)
processor.min_detected_photons_filter(2)
processor.with_input(input_state)

sampler = pcvl.algorithm.Sampler(processor)
probs = sampler.probs()['results']
print(f”Probabilities: {probs}")
```

# Perceval “Hello world”

Create an input Fock state

Define your circuit

Select a platform

Get your results

```
import perceval as pcvl

input_state = pcvl.BasicState("|1,1>")

circuit = pcvl.Circuit(2) // pcvl.BS()

processor = pcvl.Processor("SLOS")
processor.set_circuit(circuit)
processor.min_detected_photons_filter(2)
processor.with_input(input_state)

sampler = pcvl.algorithm.Sampler(processor)
probs = sampler.probs()['results']
print(f”Probabilities: {probs}")


_____________________________________________
Probabilities: {
                                |2,0⟩: 0.5
                                |0, 2⟩: 0.5
}
```

# Perceval “Hello world”

Create an input Fock state

Define your circuit

Select a platform

Get your results

```
import perceval as pcvl

input_state = pcvl.BasicState(“|1,1>”)

circuit = pcvl.Circuit(2) // pcvl.BS()
noise_model = pcvl.NoiseModel(brightness=1,
g2=0, indistinguishability=1)

processor = pcvl.Processor(“SLOS”,
noise=noise_model)
processor.set_circuit(circuit)
processor.min_detected_photons_filter(2)
processor.with_input(input_state)

sampler = pcvl.algorithm.Sampler(processor)
probs = sampler.probs()['results']
print(f”Probabilities: {probs}”)


_____________________________________________
Probabilities: {
                                |2,0⟩: 0.5
                                |0, 2⟩: 0.5
}
```


# QUESTIONS AND ANSWERS

samuel.horsch@quandela.com

Quandela.com