GHENT
UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

DEPARTMENT OF INFORMATION TECHNOLOGY
PHOTONICS RESEARCH GROUP

A SOFTWARE LANGUAGE APPROACH FOR DESCRIBING AND PROGRAMMING PHOTONICS HARDWARE

Master's thesis defence - Sébastien d'Herbais de Thun - 29th of June 2023

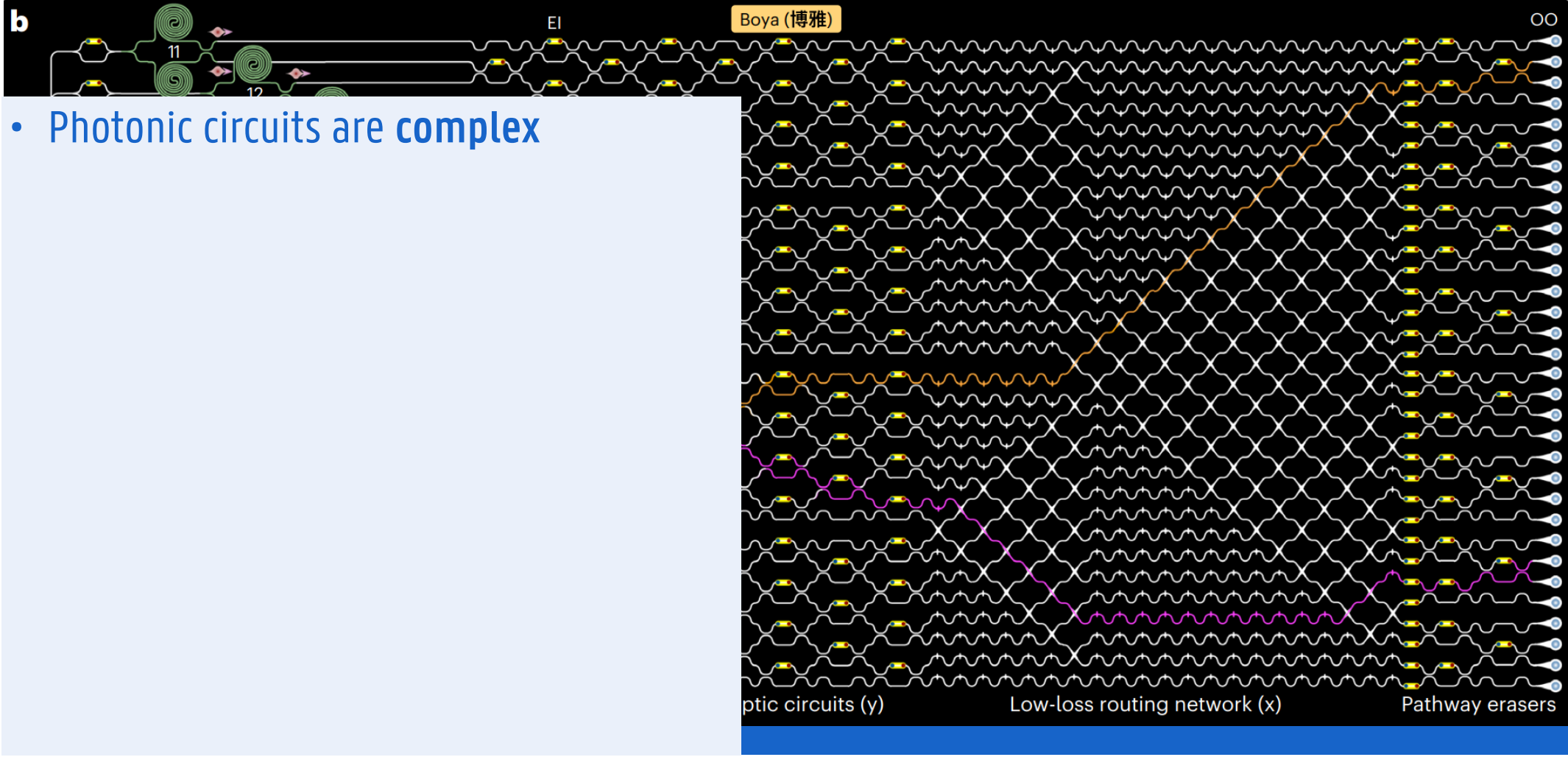
Promoters: Prof. dr. ir. Wim Bogaerts, Prof. dr. ir. Dirk Stroobandt

GHENT UNIVERSITY

The elevator pitch

GHENT
UNIVERSITY

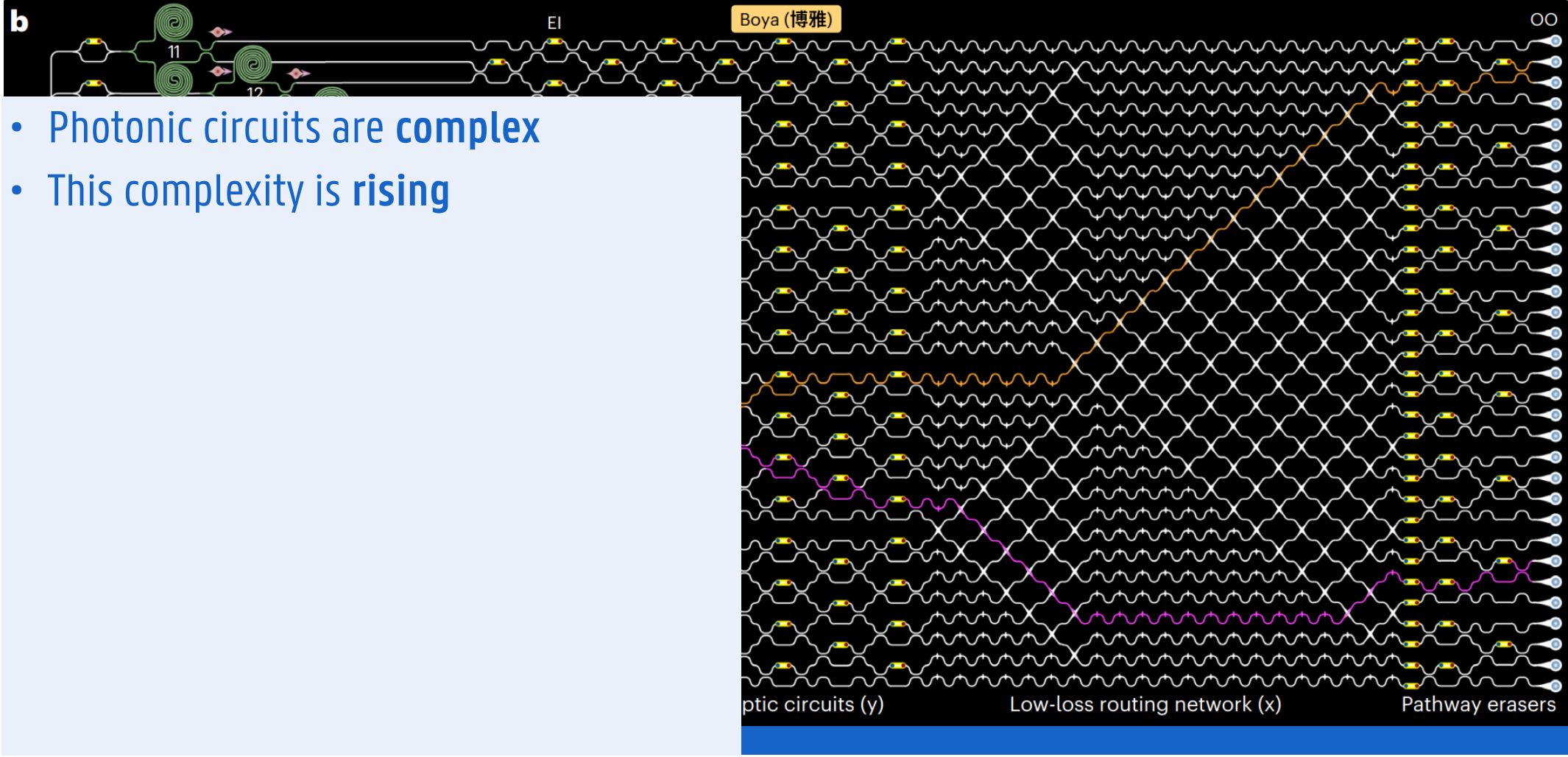
- Photonic circuits are **complex**



GHENT UNIVERSITY

Picture: *Bao Jueming, et al.* [1]

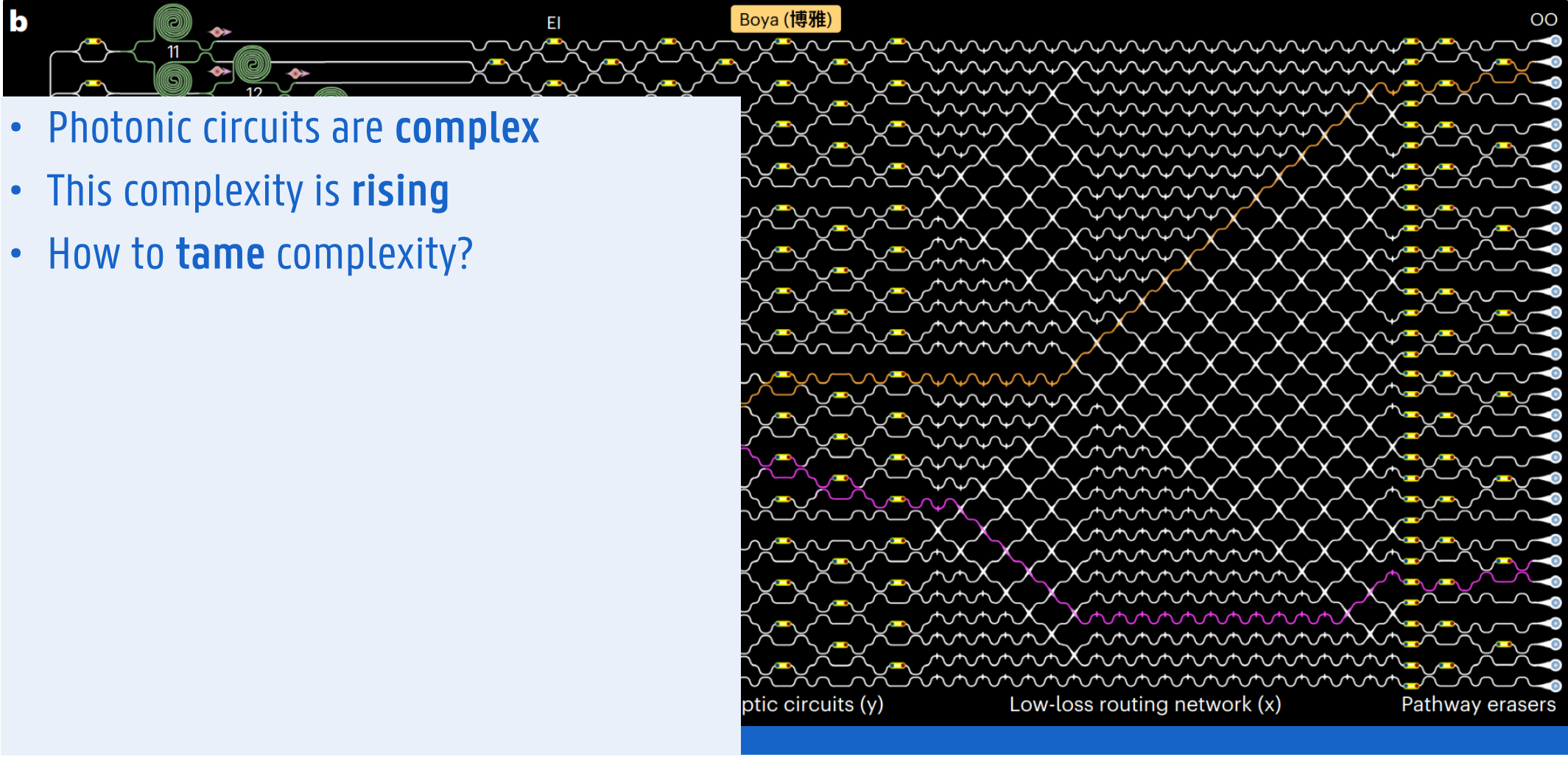
- Photonic circuits are **complex**
- This complexity is **rising**



GHENT UNIVERSITY

Picture: *Bao Jueming, et al.* [1]

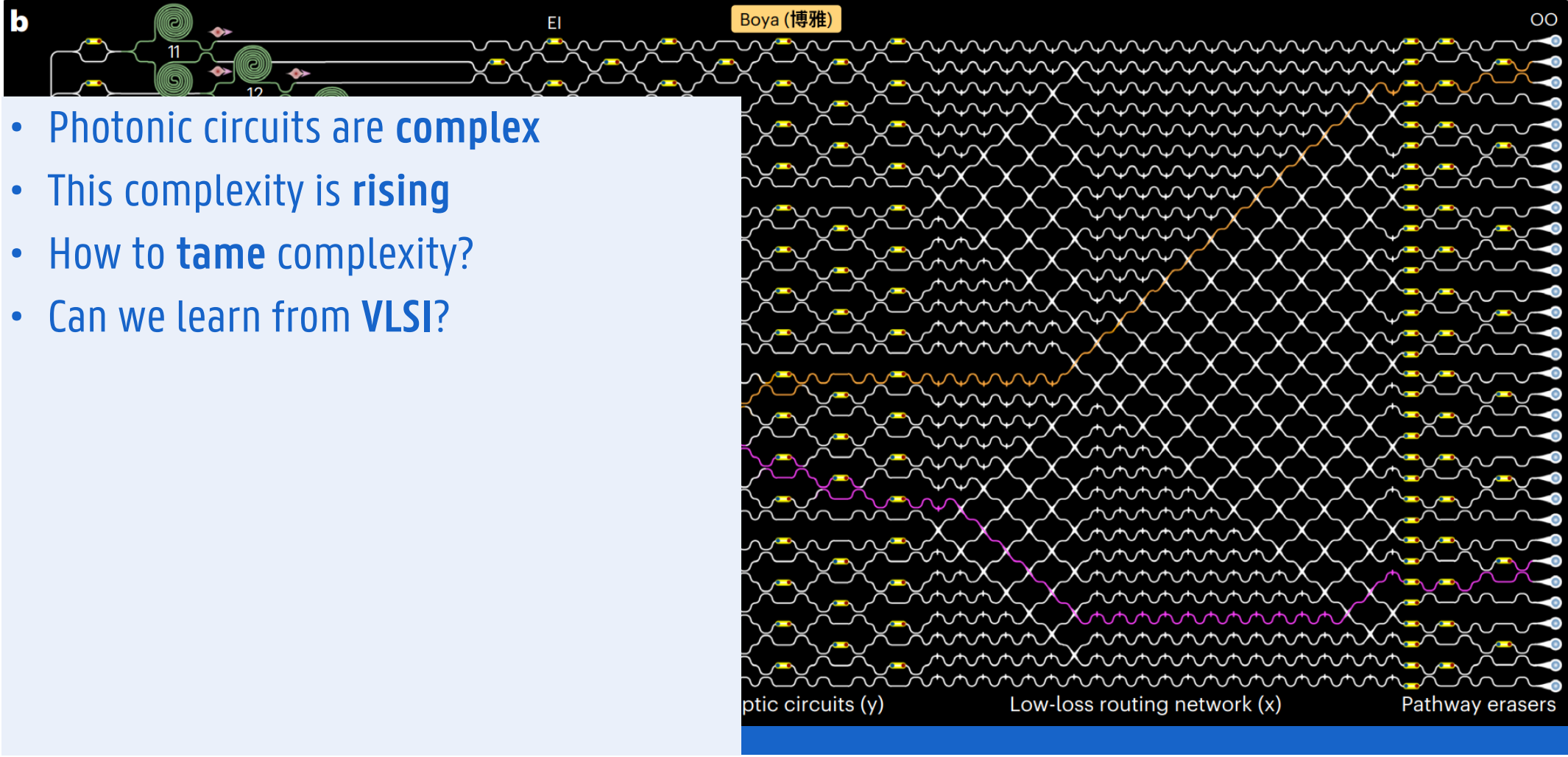
- Photonic circuits are **complex**
- This complexity is **rising**
- How to **tame** complexity?



GHENT UNIVERSITY

Picture: *Bao Jueming, et al.* [1]

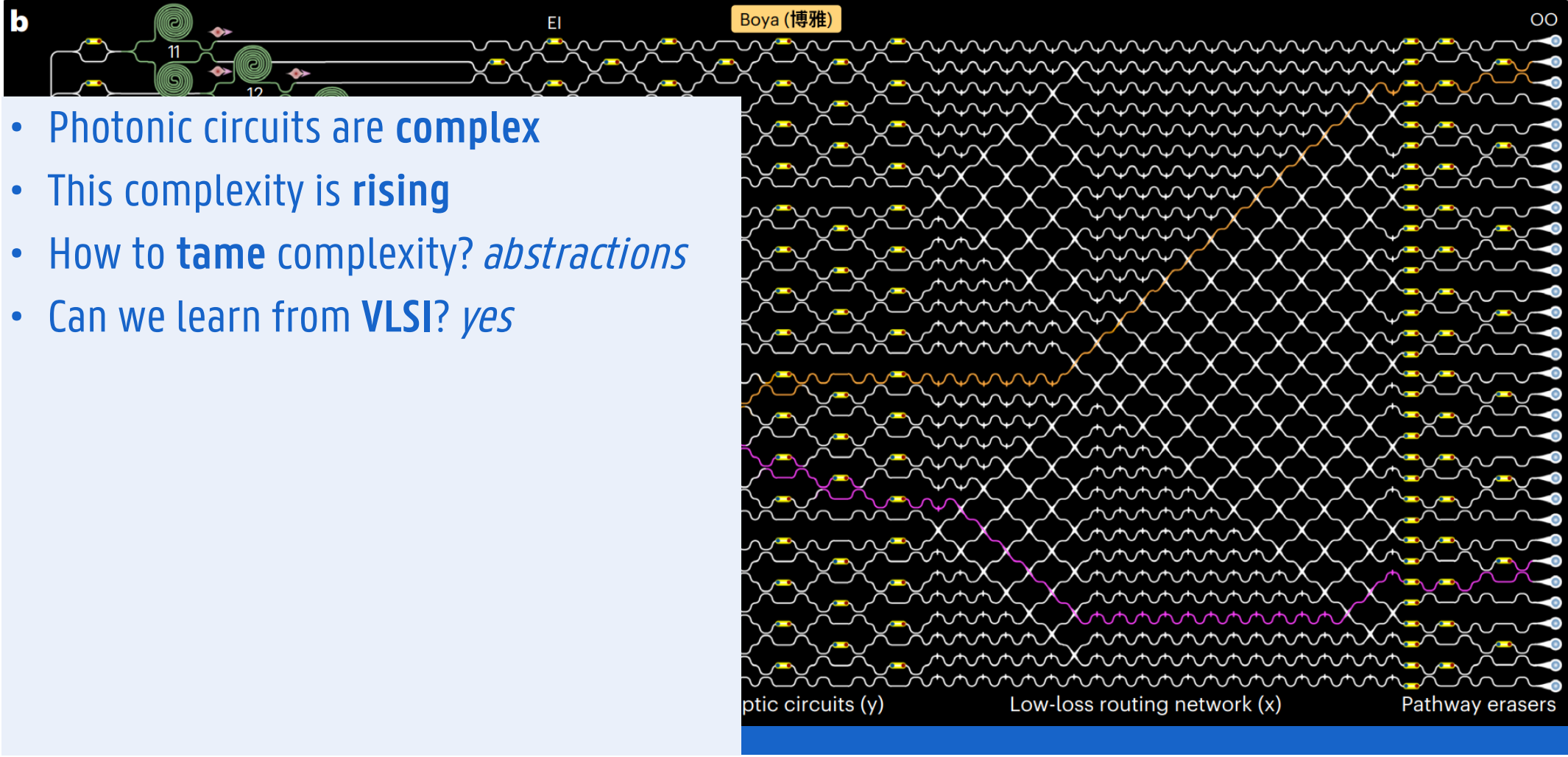
- Photonic circuits are **complex**
- This complexity is **rising**
- How to **tame** complexity?
- Can we learn from **VLSI**?



GHENT UNIVERSITY

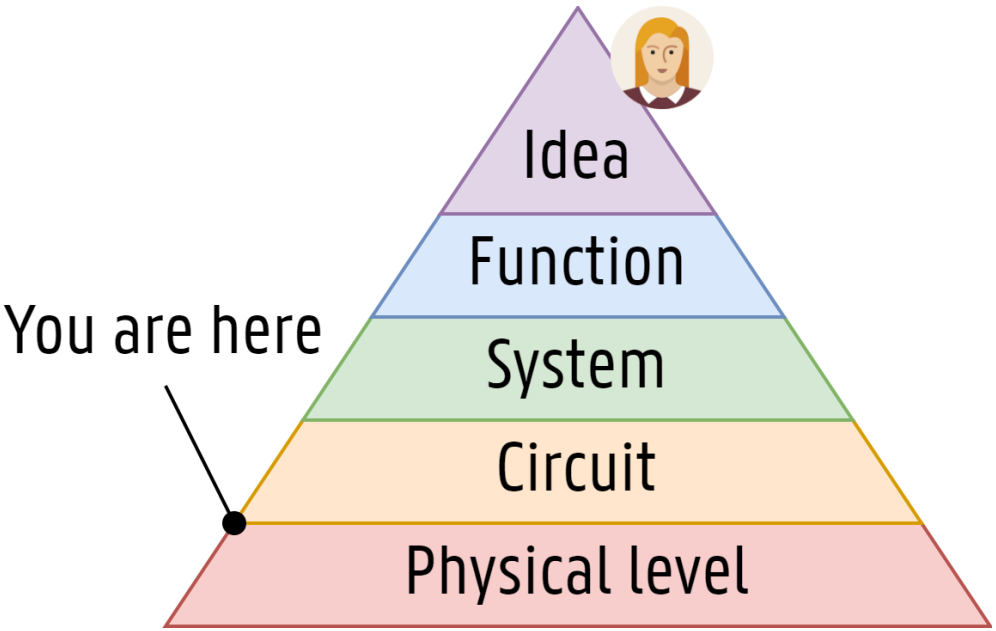
Picture: *Bao Jueming, et al.* [1]

- Photonic circuits are **complex**
- This complexity is **rising**
- How to **tame** complexity? *abstractions*
- Can we learn from **VLSI**? *yes*



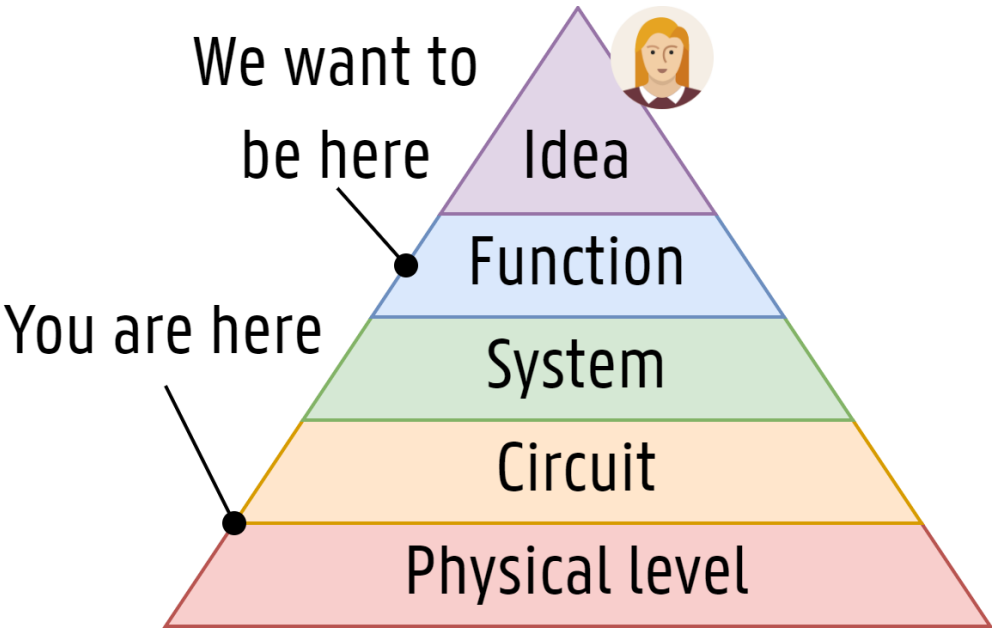
Levels of abstraction

- Currently low



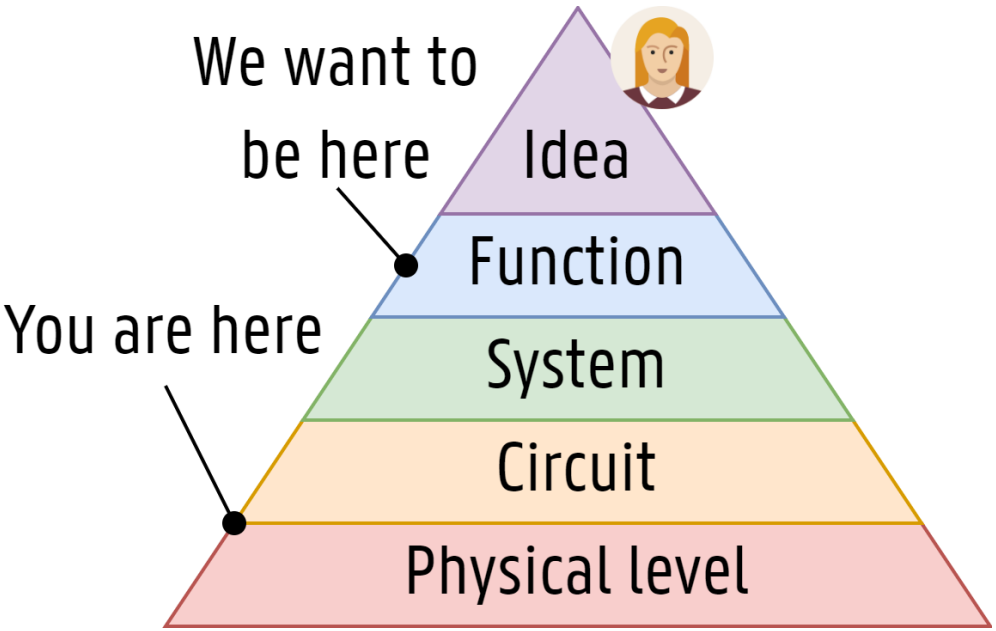
Levels of abstraction

- Currently low
- We want to go higher



Levels of abstraction

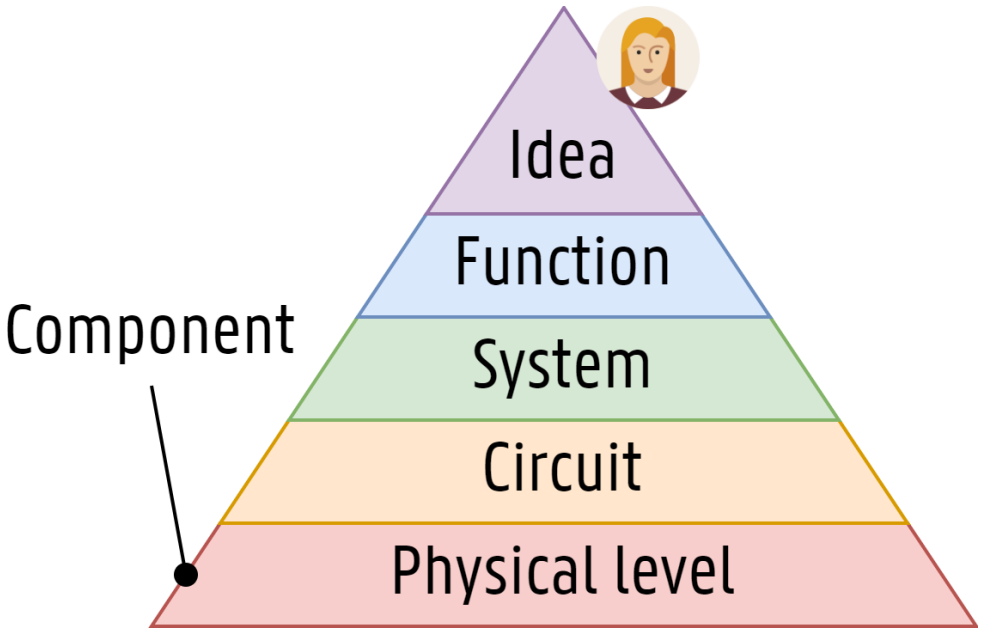
- Currently low
- We want to go higher
- We want to go **much** higher



GHENT
UNIVERSITY

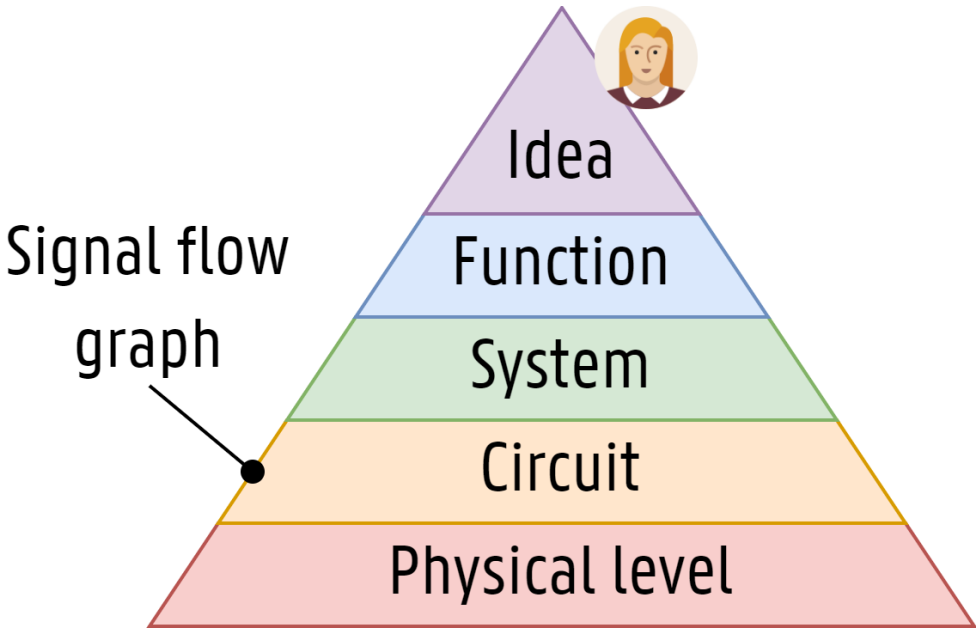
Levels of abstraction

- Currently low
- We want to go higher
- We want to go **much** higher
- We need to build abstractions
  - Components (parametric)



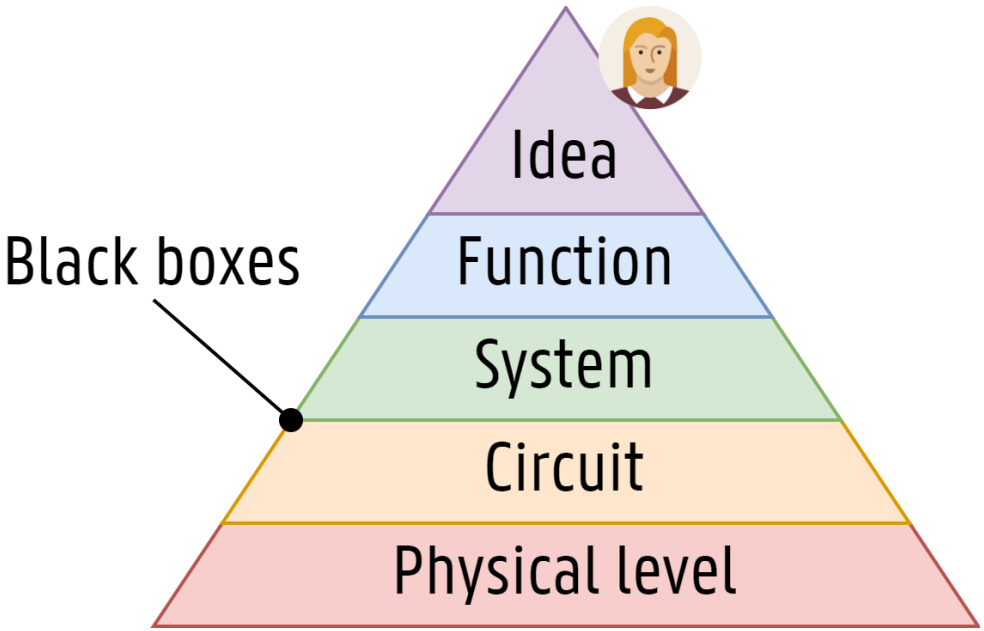
Levels of abstraction

- Currently low
- We want to go higher
- We want to go **much** higher
- We need to build abstractions
  - Components (parametric)
  - Signal flow graphs



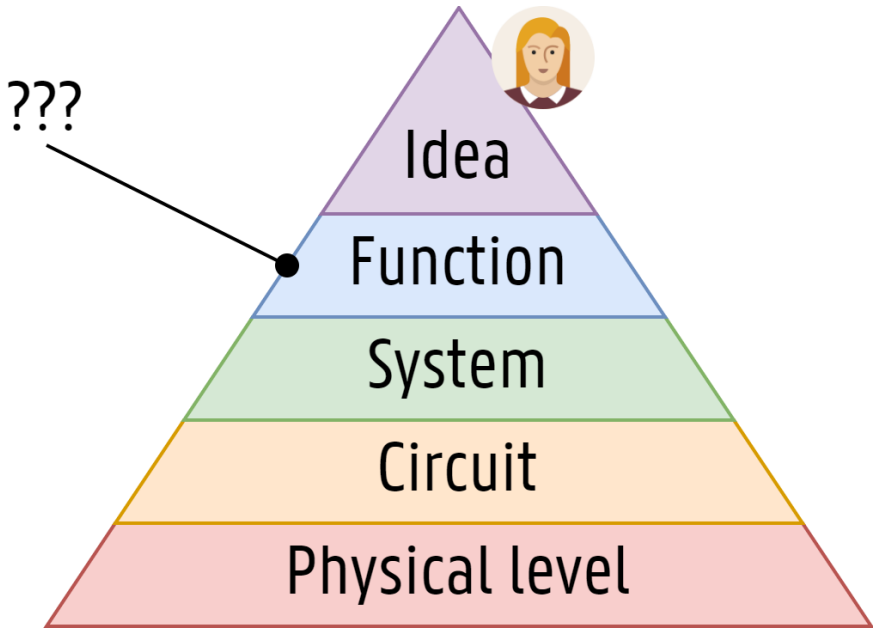
Levels of abstraction

- Currently low
- We want to go higher
- We want to go **much** higher
- We need to build abstractions
  - Components (parametric)
  - Signal flow graphs
  - Black boxes



Levels of abstraction

- Currently low
- We want to go higher
- We want to go **much** higher
- We need to build abstractions
  - Components (parametric)
  - Signal flow graphs
  - Black boxes
  - ???



Introducing PHÔS

- PHÔS is a **new language** and the result of this thesis

Introducing PHÔS

- PHÔS is a **new language** and the result of this thesis
- PHÔS is a **domain-specific language**

Introducing PHÔS

- PHÔS is a **new language** and the result of this thesis
- PHÔS is a **domain-specific language**
- PHÔS describes **photonic circuits**

Introducing PHÔS

- PHÔS is a **new language** and the result of this thesis
- PHÔS is a **domain-specific language**
- PHÔS describes **photonic circuits**
- PHÔS is **declarative**

Introducing PHÔS

- PHÔS is a **new language** and the result of this thesis
- PHÔS is a **domain-specific language**
- PHÔS describes **photonic circuits**
- PHÔS is **declarative**
- PHÔS is **parametric**

Introducing PHÔS

- PHÔS is a **new language** and the result of this thesis
- PHÔS is a **domain-specific language**
- PHÔS describes **photonic circuits**
- PHÔS is **declarative**
- PHÔS is **parametric**
- PHÔS is **expressive**

Introducing PHÔS

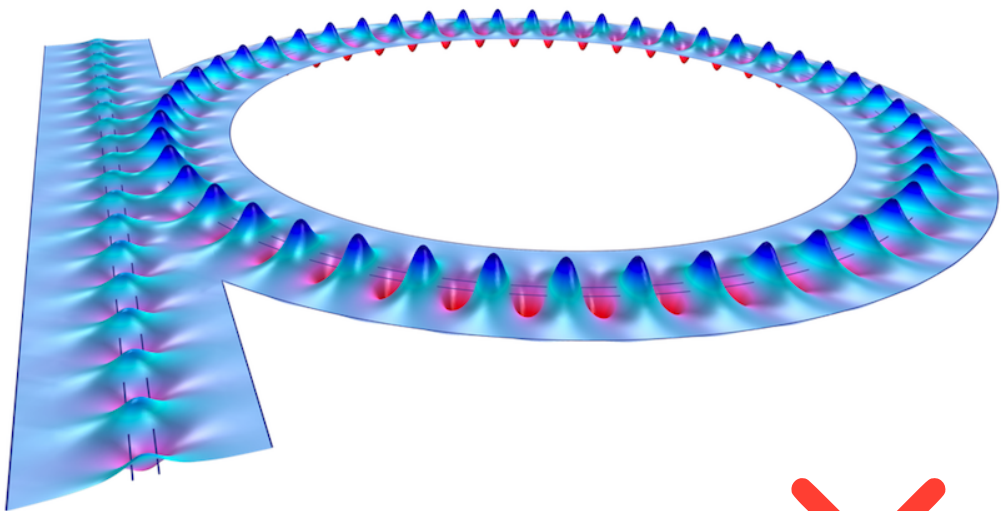
- PHÔS is a **new language** and the result of this thesis
- PHÔS is a **domain-specific language**
- PHÔS describes **photonic circuits**
- PHÔS is **declarative**
- PHÔS is **parametric**
- PHÔS is **expressive**
- PHÔS is **extensible**

Introducing PHÔS

- PHÔS is a **new language** and the result of this thesis
- PHÔS is a **domain-specific language**
- PHÔS describes **photonic circuits**
- PHÔS is **declarative**
- PHÔS is **parametric**
- PHÔS is **expressive**
- PHÔS is **extensible**
- PHÔS is **not finished** nor **perfect**
  - We need **you** to make it better!

- PHÔS is **not** at the component level
  - ~~Component design~~
  - ~~Component simulation~~
  - ~~Component optimization~~

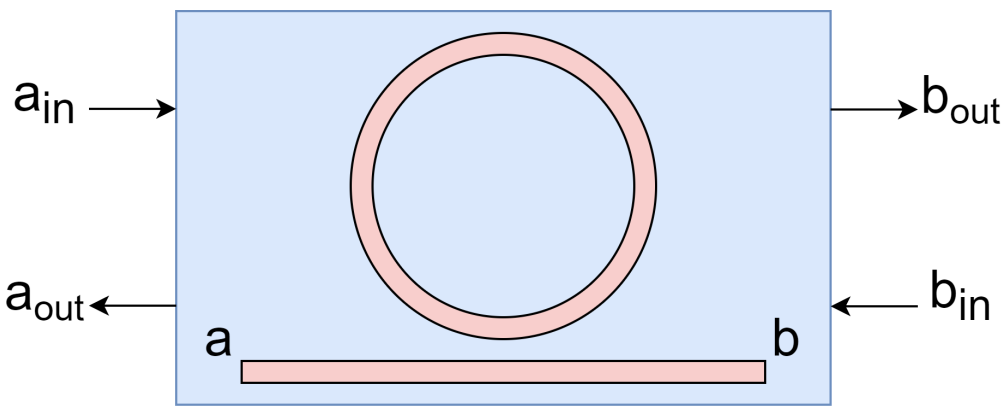
Introducing PHÔS



- PHÔS is **not** at the component level
  - ~~Component design~~
  - ~~Component simulation~~
  - ~~Component optimization~~
- PHÔS is the **function** and **system** levels
  - Filter synthesis
  - Signal flow graph generation
  - Component modeling & instantiation
  - Reconfigurability & tunability
  - Optimization

Picture: *Brianne Christopher* [2]

Introducing PHÔS



- PHÔS is **not** at the component level
  - ~~Component design~~
  - ~~Component simulation~~
  - ~~Component optimization~~
- PHÔS is the **function** and **system** levels
  - Filter synthesis
  - Signal flow graph generation
  - Component modeling & instantiation
  - Reconfigurability & tunability
  - Optimization

GHENT UNIVERSITY

About this presentation

- Elevator pitch
- Programmatic description: an overview
- Example: 16-QAM modulator
- Example: Lattice filter
- Conclusion
- Future work

Programmatic description: an overview

GHENT
UNIVERSITY

Translation of intent

- How do we tell the computer what we want?
- What do we want the computer to do for us?
- How does the computer do it?

Translation of intent

- How do we tell the computer what we want? **Programming!**
- What do we want the computer to do for us?
- How does the computer do it?

Translation of intent

- How do we tell the computer what we want? **Programming!**
- What do we want the computer to do for us? **As much as possible!**
- How does the computer do it?

Translation of intent

- How do we tell the computer what we want? **Programming!**
- What do we want the computer to do for us? **As much as possible!**
- How does the computer do it? **Compilation, Evaluation, and Synthesis!**

How do you describe photonic circuit?

- Scaling graphical circuits is **really** hard
- Scaling code is **really** easy

How do you describe photonic circuit?

- Scaling graphical circuits is **really** hard
- Graphical circuits are **inflexible**

- Scaling code is **really** easy
- Code is **flexible**

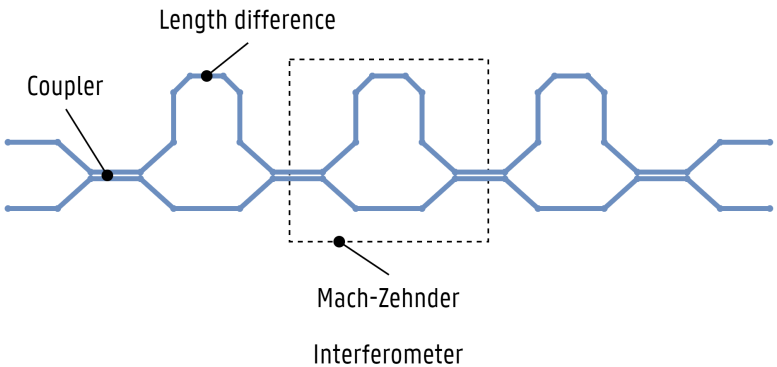
How do you describe photonic circuit?

- Scaling graphical circuits is **really** hard
- Graphical circuits are **inflexible**
- Graphical circuits are not **reusable**

- Scaling code is **really** easy
- Code is **flexible**
- Code is easily **reusable**

How do you describe photonic circuit?

- Scaling graphical circuits is **really** hard
- Graphical circuits are **inflexible**
- Graphical circuits are not **reusable**
- Graphical circuits are not **expressive**



- Scaling code is **really** easy
- Code is **flexible**
- Code is easily **reusable**
- Code is **expressive**

```
filter_kind_coefficients(filter_kind)
  |> fold((a, b), |acc, (coeff, phase)| {
    acc |> coupler(coeff)
        |> constrain(d_phase = phase)
  })
```

PHÔS

GHENT UNIVERSITY

Yes, but why a new language?

- Existing languages **do not** works for photonics
  - Hardware description languages: ~~VHDL~~, ~~MyHDL~~
  - High-level synthesis languages: ~~SystemC~~
  - Analog modeling languages: ~~Verilog-AMS~~, ~~SPICE~~
  - Traditional programming languages: ~~Python~~, ~~Rust~~

Yes, but why a new language?

- Existing languages **do not** works for photonics
  - Hardware description languages: ~~VHDL~~, ~~MyHDL~~
  - High-level synthesis languages: ~~SystemC~~
  - Analog modeling languages: ~~Verilog-AMS~~, ~~SPICE~~
  - Traditional programming languages: ~~Python~~, ~~Rust~~
- Libraries are **not expressive** enough

Yes, but why a new language?

- Existing languages **do not** works for photonics
  - Hardware description languages: ~~VHDL~~, ~~MyHDL~~
  - High-level synthesis languages: ~~SystemC~~
  - Analog modeling languages: ~~Verilog-AMS~~, ~~SPICE~~
  - Traditional programming languages: ~~Python~~, ~~Rust~~
- Libraries are **not expressive** enough
- Why? **Because photonics is different**
  - ~~Sequential~~ Continuous
  - ~~Digital~~ Analog

Yes, but why a new language?

- Existing languages **do not** works for photonics
  - Hardware description languages: ~~VHDL~~, ~~MyHDL~~
  - High-level synthesis languages: ~~SystemC~~
  - Analog modeling languages: ~~Verilog-AMS~~, ~~SPICE~~
  - Traditional programming languages: ~~Python~~, ~~Rust~~
- Libraries are **not expressive** enough
- Why? **Because photonics is different**
  - ~~Sequential~~ Continuous
  - ~~Digital~~ Analog
- We need a **domain-specific language**

What do we want the computer to do for us?

- **Ideal behaviour**: feedback loops, calibration

What do we want the computer to do for us?

- **Ideal behaviour**: feedback loops, calibration
- **Simulation**: simulator, interface with existing ones

What do we want the computer to do for us?

- **Ideal behaviour**: feedback loops, calibration
- **Simulation**: simulator, interface with existing ones
- **Platform independence**: process, foundry, processor architecture

What do we want the computer to do for us?

- **Ideal behaviour**: feedback loops, calibration
- **Simulation**: simulator, interface with existing ones
- **Platform independence**: process, foundry, processor architecture
- **Visualization**: signal flow graphs, circuit diagrams

What do we want the computer to do for us?

- **Ideal behaviour**: feedback loops, calibration
- **Simulation**: simulator, interface with existing ones
- **Platform independence**: process, foundry, processor architecture
- **Visualization**: signal flow graphs, circuit diagrams
- **Reconfigurability**: reconfigurability through branching

What do we want the computer to do for us?

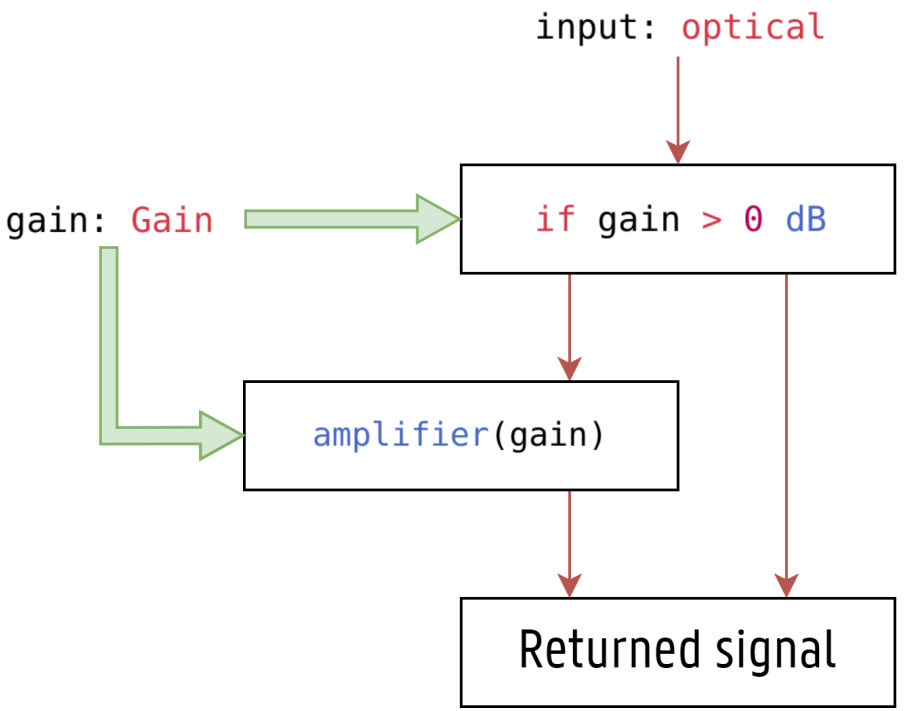
- **Ideal behaviour**: feedback loops, calibration
- **Simulation**: simulator, interface with existing ones
- **Platform independence**: process, foundry, processor architecture
- **Visualization**: signal flow graphs, circuit diagrams
- **Reconfigurability**: reconfigurability through branching
- **Tunability**: implicit tunability

What do we want the computer to do for us?

- **Ideal behaviour**: feedback loops, calibration
- **Simulation**: simulator, interface with existing ones
- **Platform independence**: process, foundry, processor architecture
- **Visualization**: signal flow graphs, circuit diagrams
- **Reconfigurability**: reconfigurability through branching
- **Tunability**: implicit tunability
- **Programmability**: hardware abstraction layer (HAL)

Reconfigurability and Tunability

```
1   syn my_circuit(                          PHÔS
2     input: optical,
3     gain: Gain
4   ) -> optical {
5     if gain > 0 dB {
6       input |> amplifier(gain)
7     } else {
8       input
9     }
10  }
```



GHENT UNIVERSITY

Tying it all together

- Express **constraints** on the signals and values

Tying it all together

- Express **constraints** on the signals and values
- Used for **verification** and **optimization**

Tying it all together

- Express **constraints** on the signals and values
- Used for **verification** and **optimization**
- Used to **reduce** reconfigurability space

Tying it all together

- Express **constraints** on the signals and values
- Used for **verification** and **optimization**
- Used to **reduce** reconfigurability space
- Used to **simulate** the circuit

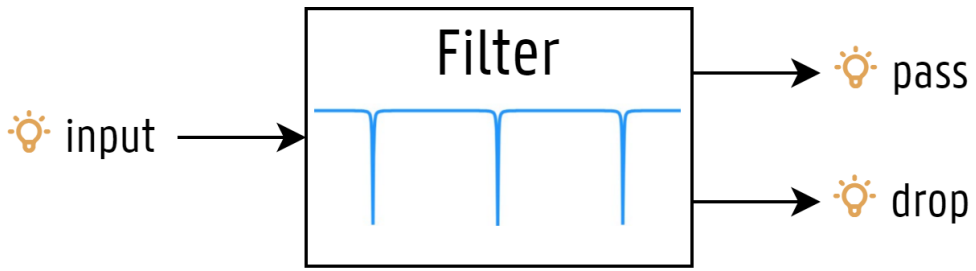
Tying it all together

- Express **constraints** on the signals and values
- Used for **verification** and **optimization**
- Used to **reduce** reconfigurability space
- Used to **simulate** the circuit

```
1  syn amplifier(                                    PHÔS
2      @power(max(0 dBm - gain))
3      input: optical,
4
5      @max(10 dB)
6      gain: Gain,
7  ) -> @power(input + gain) optical {
8      ...
9  }
```

What is a circuit made of?

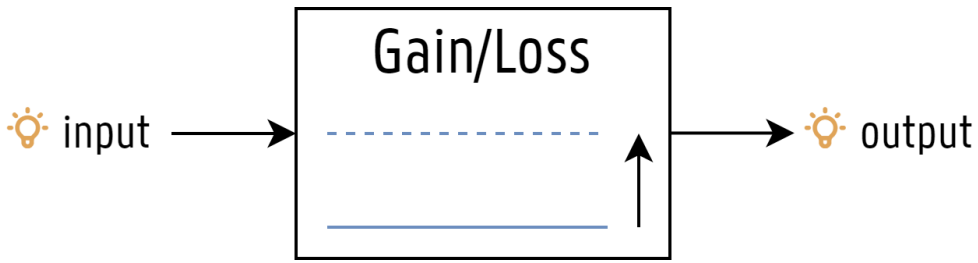
- **Filters**



Wavelength constraint

What is a circuit made of?

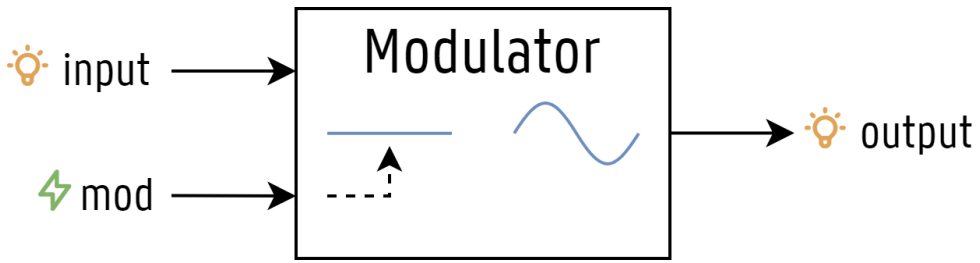
- **Filters**
- **Gain** and **loss** elements



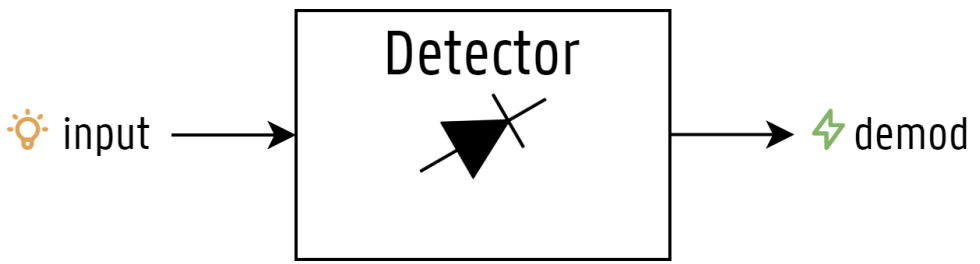
Power constraint

What is a circuit made of?

- **Filters**
- **Gain** and **loss** elements
- **Modulators** and **detectors**



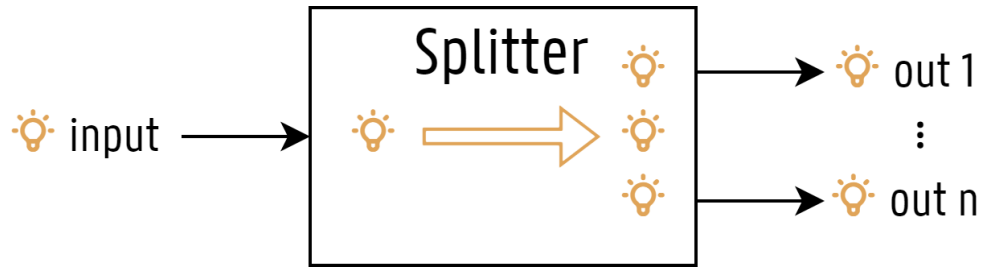
(no constraint)



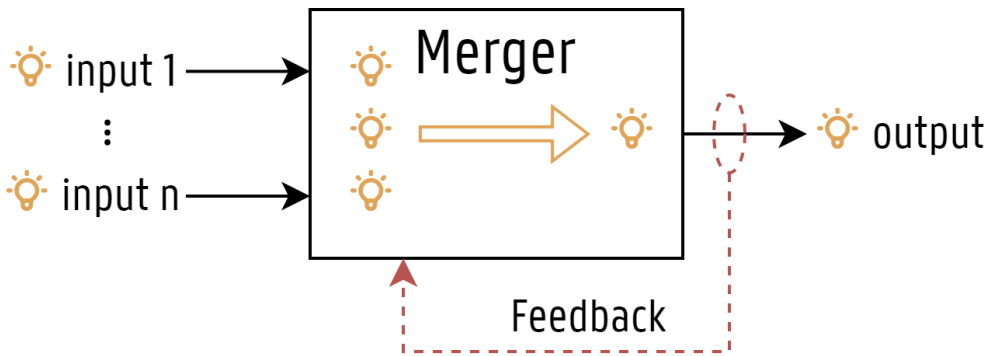
(no constraint)

What is a circuit made of?

- **Filters**
- **Gain** and **loss** elements
- **Modulators** and **detectors**
- **Splitters** and **combiners**



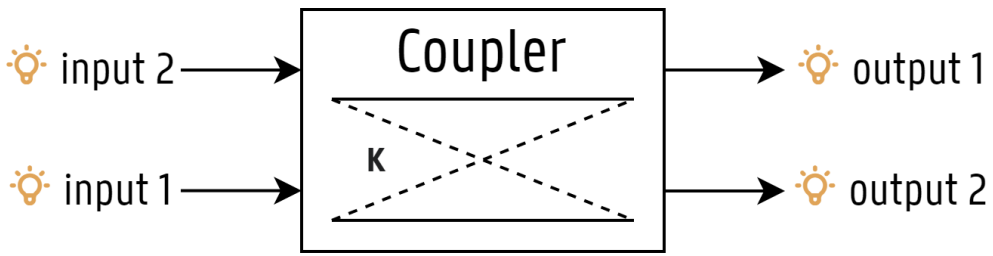
Power constraint



Power constraint

What is a circuit made of?

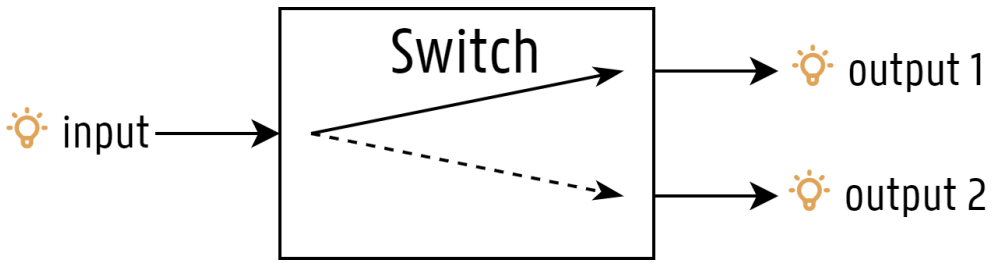
- **Filters**
- **Gain** and **loss** elements
- **Modulators** and **detectors**
- **Splitters** and **combiners**
- **Couplers**



(no constraint)

What is a circuit made of?

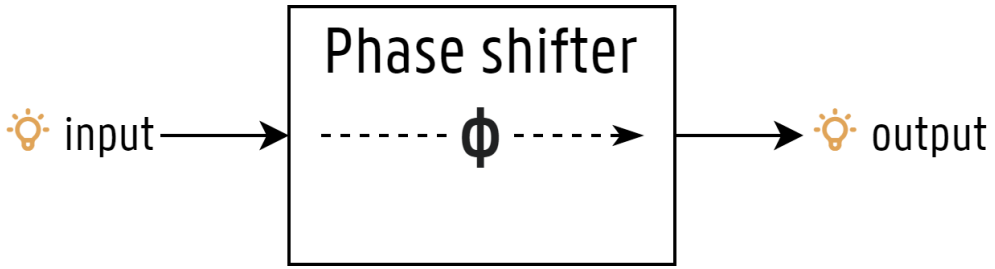
- **Filters**
- **Gain** and **loss** elements
- **Modulators** and **detectors**
- **Splitters** and **combiners**
- **Couplers**
- **Switches**



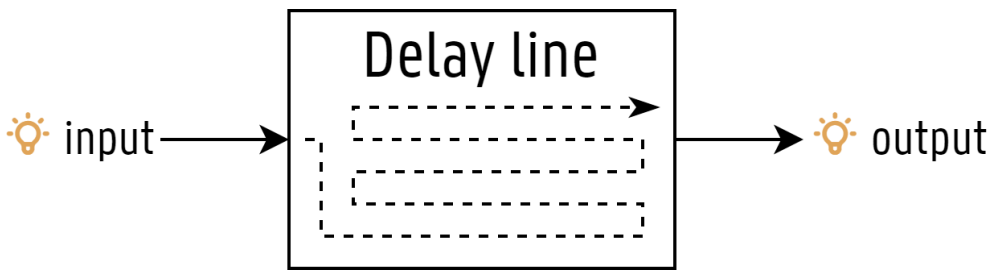
(no constraint)

What is a circuit made of?

- **Filters**
- **Gain** and **loss** elements
- **Modulators** and **detectors**
- **Splitters** and **combiners**
- **Couplers**
- **Switches**
- **Phase shifters** and **delay lines**



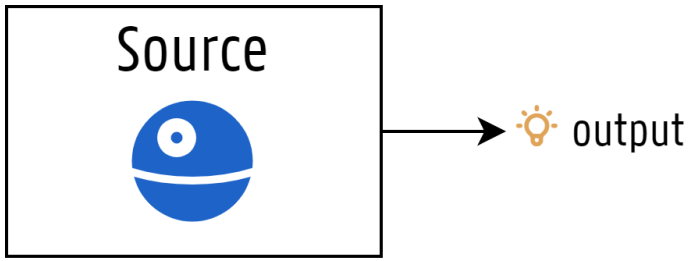
Phase constraint



Delay constraint

What is a circuit made of?

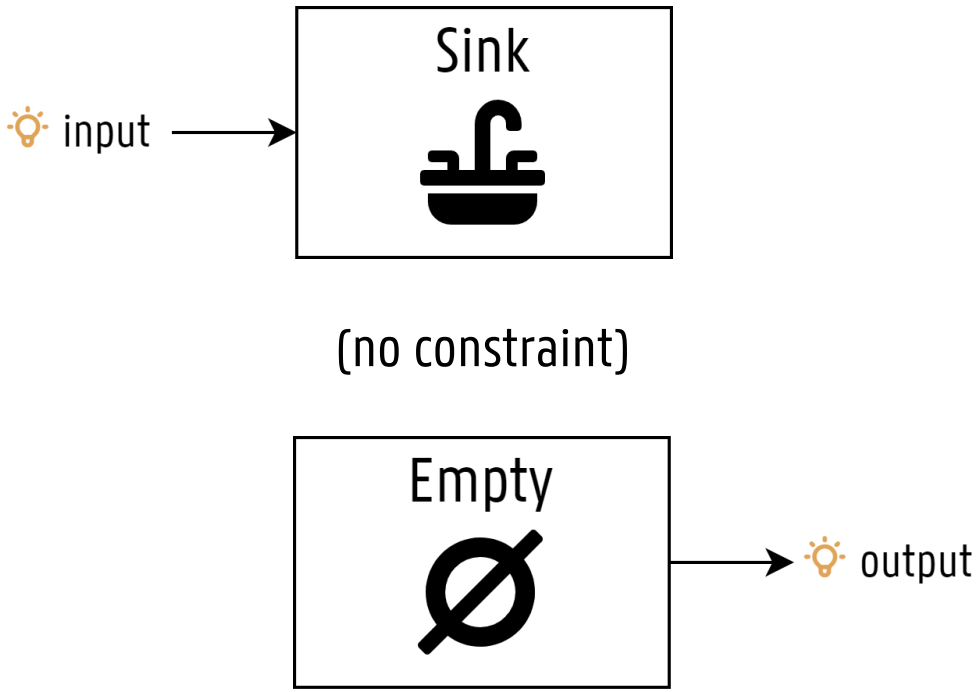
- **Filters**
- **Gain** and **loss** elements
- **Modulators** and **detectors**
- **Splitters** and **combiners**
- **Couplers**
- **Switches**
- **Phase shifters** and **delay lines**
- **Sources**



Power and **Wavelength** constraint

What is a circuit made of?

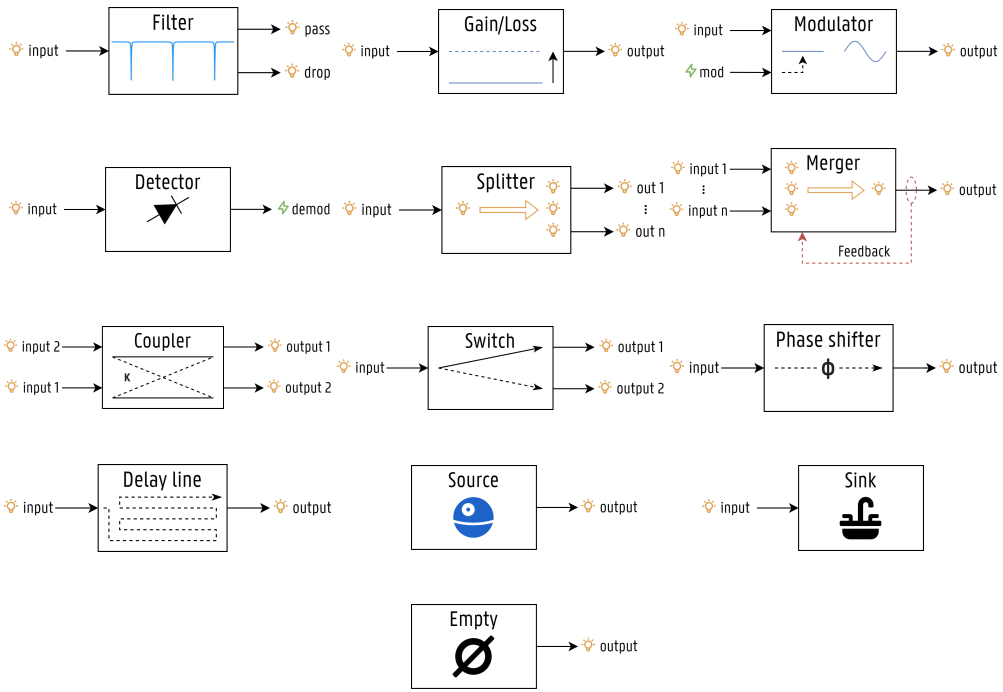
- **Filters**
- **Gain** and **loss** elements
- **Modulators** and **detectors**
- **Splitters** and **combiners**
- **Couplers**
- **Switches**
- **Phase shifters** and **delay lines**
- **Sources**
- **Sinks**, and **empty** signals



Power and **Wavelength** constraint

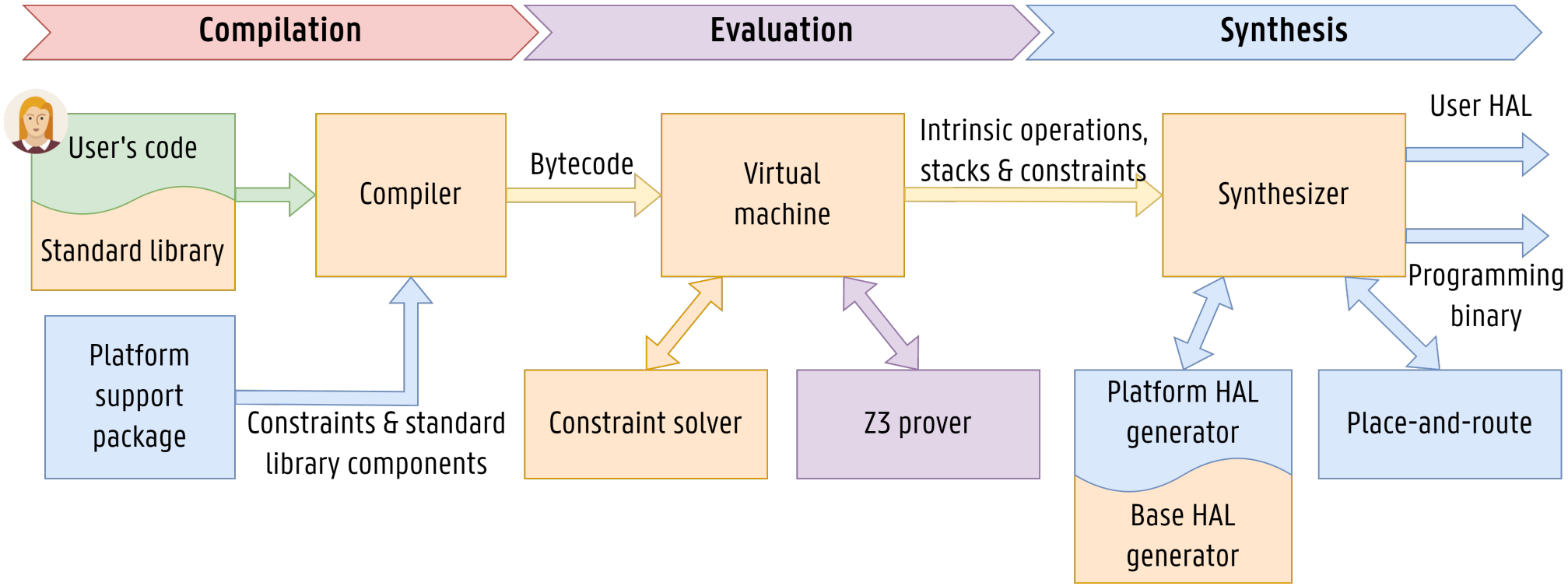
What is a circuit made of?

- **Filters**
- **Gain** and **loss** elements
- **Modulators** and **detectors**
- **Splitters** and **combiners**
- **Couplers**
- **Switches**
- **Phase shifters** and **delay lines**
- **Sources**
- **Sinks**, and **empty** signals
- Together, these form the **intrinsic operations**



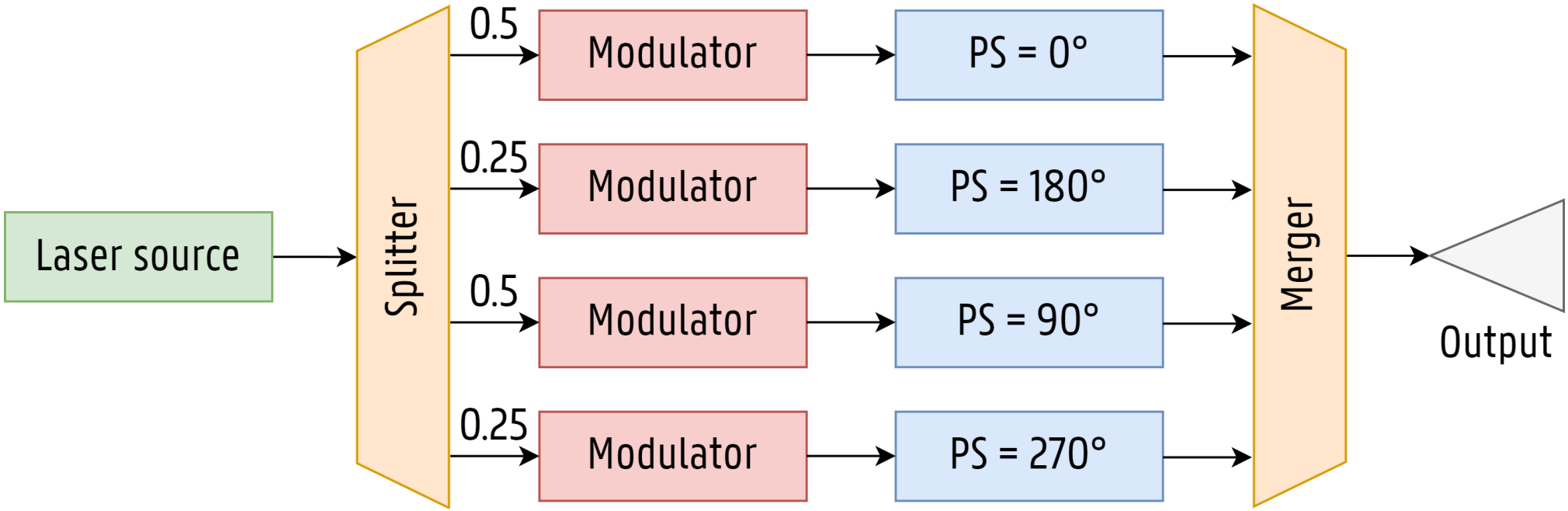
GHENT UNIVERSITY

Overview


Compilation
Evaluation
Synthesis
User's code
Standard library
Compiler
Bytecode
Virtual machine
Intrinsic operations, stacks & constraints
Synthesizer
User HAL
Programming binary
Platform support package
Constraints & standard library components
Constraint solver
Z3 prover
Platform HAL generator
Base HAL generator
Place-and-route

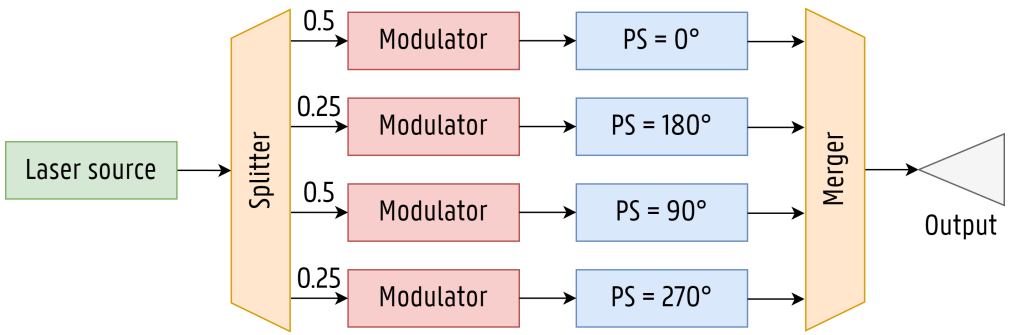
Examples

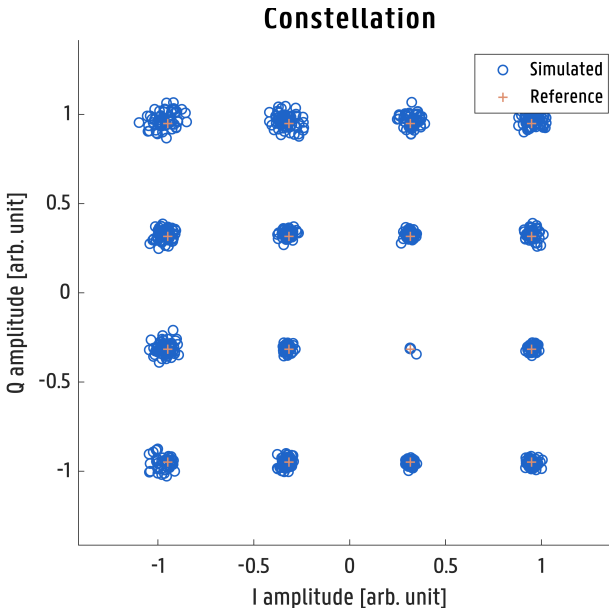
16-QAM 400 Gb/s modulator



16-QAM 400 Gb/s modulator (cont.)

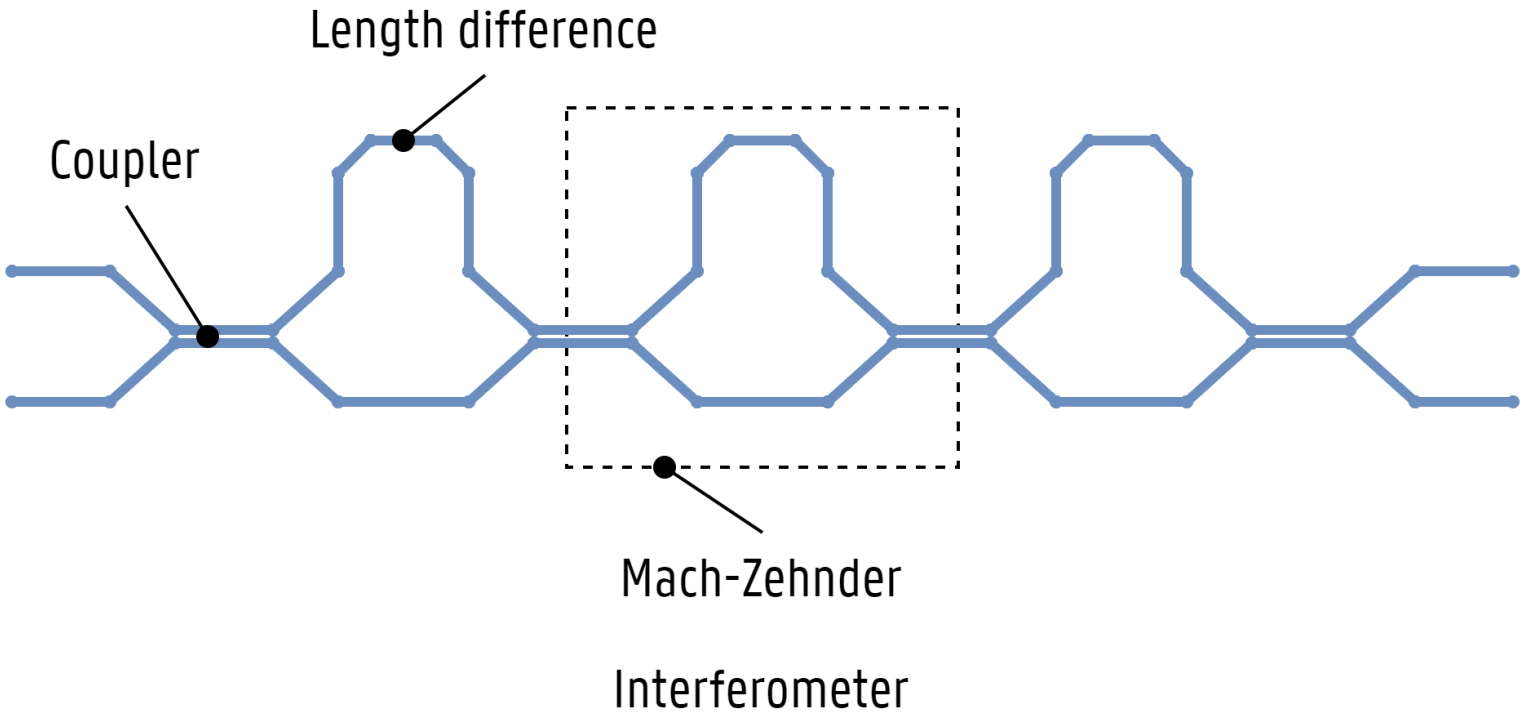
```
1   syn coherent_transmitter(                    PHÔS
2       input: optical,
3       [a, b, c, d]: [electrical; 4],
4   ) -> optical {
5       input
6           |> split((1.0, 1.0, 0.5, 0.5))
7           |> zip((a, c, b, d))
8           |> modulate(Modulation::Amplitude)
9           |> constrain(d_phase = 90°)
10          |> merge()
11  }
```





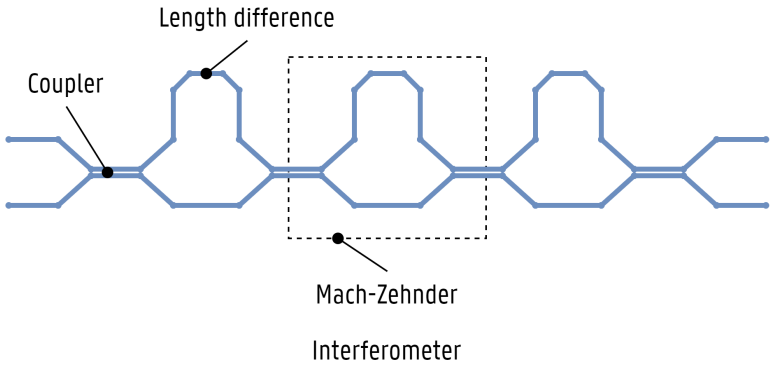
GHENT UNIVERSITY

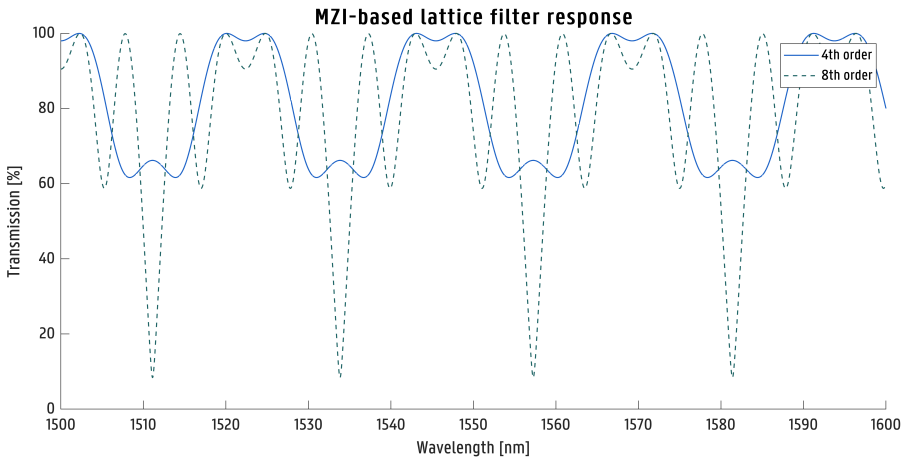
Lattice filter



Lattice filter (cont.)

```
1   syn lattice_filter(                                    PHÔS
2     a: optical,
3     b: optical,
4     filter_kind: FilterKind
5   ) -> (optical, optical) {
6     filter_kind_coefficients(filter_kind)
7       |> fold((a, b), |acc, (coeff, phase)| {
8         acc |> coupler(coeff)
9             |> constrain(d_phase = phase)
10      })
11  }
```





GHENT UNIVERSITY

Conclusion

GHENT
UNIVERSITY

Future works

- Implementing PHÔS fully
- Co-simulation with digital and analog circuits
- Place-and-route
- Language improvements
- Advanced constraint inference
- **TEST ALL THE THINGS!**

Key takeaways

- Novel programmatic way of **describing photonics**:
  - **Expressive**, **flexible**, **reusable**, and **programmable**
  - Opens the way to **VLSI for photonics**

Key takeaways

- Novel programmatic way of **describing photonics**:
  - **Expressive**, **flexible**, **reusable**, and **programmable**
  - Opens the way to **VLSI for photonics**
- Novel **constraint system** for photonics:
  - **Optimization**, **verification**, **simulation**

Key takeaways

- Novel programmatic way of **describing photonics**:
  - **Expressive**, **flexible**, **reusable**, and **programmable**
  - Opens the way to **VLSI for photonics**
- Novel **constraint system** for photonics:
  - **Optimization**, **verification**, **simulation**
- Now we need **you** to **improve** it!

Thank you for listening

GHENT
UNIVERSITY

FACULTY OF ENGINEERING
AND ARCHITECTURE

Sébastien d'Herbais de Thun

Student

Photonics Research Group

E sebastien.dherbaisdethun@ugent.be
M +32 (0) 473 12 86 57

www.ugent.be

Universiteit Gent
@ugent
@ugent
Ghent University
Sébastien d'Herbais de Thun

GHENT
UNIVERSITY

Sources

[1] J. Bao, Z. Fu, et al., “Very-large-scale integrated quantum graph photonics,” *Nature Photon.*, pp. 1–9, Apr. 2023, doi: 10.1038/s41566-023-01187-z.

[2] B. Christopher, “Calculating the Spectral Properties of an Optical Ring Resonator.” Accessed: Jun. 28, 2023. [Online]. Available: https://www.comsol.fr/blogs/calculating-the-spectral-properties-of-an-optical-ring-resonator/

Backup

GHENT
UNIVERSITY

Why a compiled language?

- Why not an intepreted language like Python?
  - **Stack collection** for tunability, reconfigurability, and programmability
  - **Static analysis** with a prover
  - Good **separation of concerns**
  - Dynamic languages are **error prone**