CLASSIQ

Intro to Classiq

Nadav Ben-Ami

Quantum Solutions Architect

CLASSIQ

Classiq - The Quantum Software Leader

A team of 120 world class experts

$200+M in funding

Presence in 12 countries with offices in US, Europe, Korea and Japan

Fortune-500 customers and partners

Roadmap to become industry standard

90 patents

Leaders In Quantum Software Development

What we do:

- Hardware-Agnostic, Quantum Software Platform
- Quantum Applications Factory
- Gov / National / Enterprise Quantum CoE providers
- Quantum AI at scale

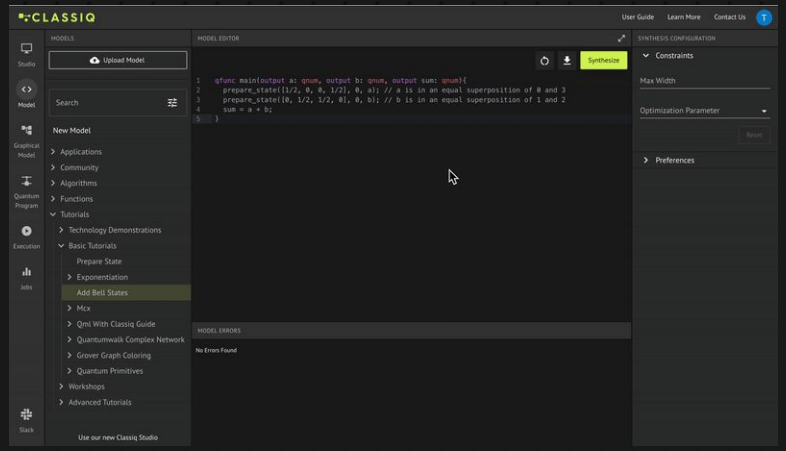
Trusted By Leading Investors

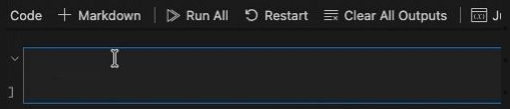
SoftBank, TEAM8, Entrée Capital, NORWEST VENTURE PARTNERS, Hewlett Packard Enterprise, HSBC, Hamilton Lane, NIGHTDRAGON, NTT, INTESA SANPAOLO, wing, cdp Venture Capital Sgr

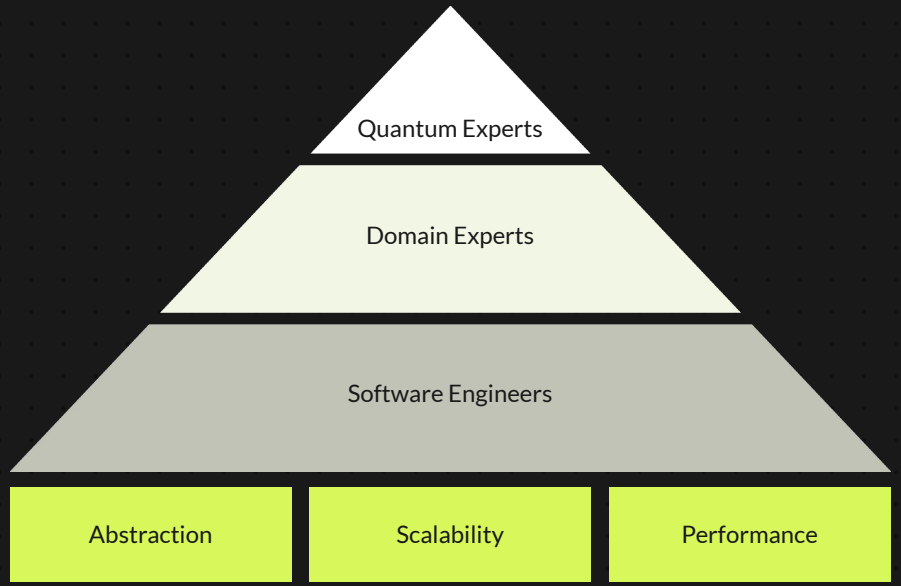
Lip-Bu Tan, CEO, Intel

Harvey Jones Former CEO, Synopsys & Board Member, Nvidia

Classiq Fits All Type Of Quantum Users







CLASSIQ

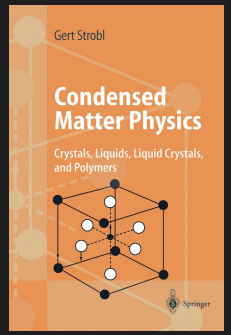
Enterprise Customers

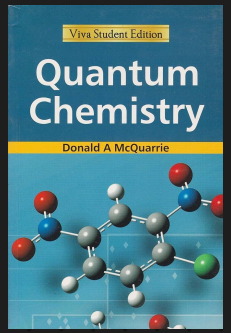
NTT	BMW	HSBC	TOSHIBA	Rolls-Royce	INTESA SANPAOLO
Confidential	Manufacturing Optimization	Fraud Detection	Quantum Machine Learning	Jet Engine Optimization	Options Pricing
IONQ	vodafone	WELLS FARGO	AT&T	DBS	MITSUBISHI CHEMICAL GROUP
Quantum Hardware	Network Optimization	Confidential	Confidential	Confidential	Quantum Chemistry
epb	Deloitte.	LG	SoftBank	COMCAST	Point72
Confidential	Multiple Internal and Customer use-cases	Quantum Machine Learning	Quantum Chemistry	Quantum Machine Learning	Portfolio Optimization
QuEra	Sumitomo	CTC	CHUGAI	Microsoft	TEL-AVIV SOURASKY MEDICAL CENTER ICHILOV MEDICAL EXCELLENCE AND COMPASSIONATE CARE
Quantum Hardware	Quantum AI	Quantum Machine Learning	Drug Discovery	Quantum Education	Brain Signal Processing
MIZUHO	DDR&D	IAI	RAFAEL	HTX	SHIONOGI
Options Pricing	Confidential	Confidential	Confidential	Confidential	Quantum Machine Learning

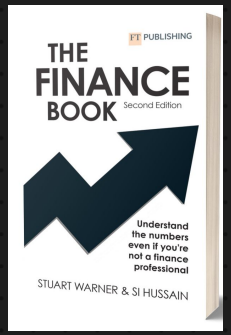
Quantum Algorithms: The Key Challenge

Development of efficient and useful quantum algorithms

Domain-Specific Problems







Quantum Algorithms

- QSVT
- HHL
- QPE

Classical Algorithms Development

Low-Level Languages
E.g. Assembly

```
section .data
    result times 100 dd 0        ; Allocate space for 100 integers initialized to 0

section .bss

section .text
    global _start

_start:
    ; Initialize loop counter
    mov ecx, 0                   ; ecx will serve as our loop counter (i)

.loop:
    ; Check if ecx (i) is 100
    cmp ecx, 100
    jge .end_loop                ; If ecx >= 100, end the loop

    ; Calculate 2*(i**2) - 5*i + 3
    mov eax, ecx                 ; eax = i
    imul eax, eax                ; eax = i * i (i**2)
    shl eax, 1                   ; eax = 2 * (i**2)

    mov ebx, ecx                 ; ebx = i
    imul ebx, 5                  ; ebx = 5 * i

    sub eax, ebx                 ; eax = 2*(i**2) - 5*i
    add eax, 3                   ; eax = 2*(i**2) - 5*i + 3

    ; Store result in the array
    mov [result + ecx * 4], eax  ; result[i] = eax

    ; Increment loop counter
    inc ecx
    jmp .loop

.end_loop:
    ; Exit the program
    mov eax, 1                   ; syscall: sys_exit
    xor ebx, ebx                 ; status: 0
    int 0x80                     ; invoke syscall

```

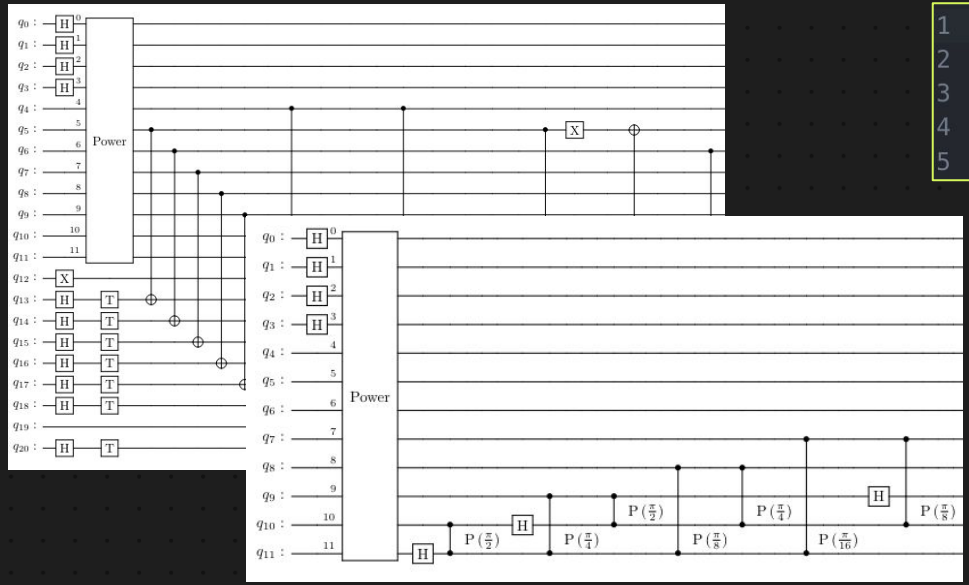
High-Level Languages
E.g. Python

```python
[2*(i**2)-5*i+3 for i in range (100)]
```

Quantum Algorithms Development

Low-Level Thinking

Building a quantum circuit



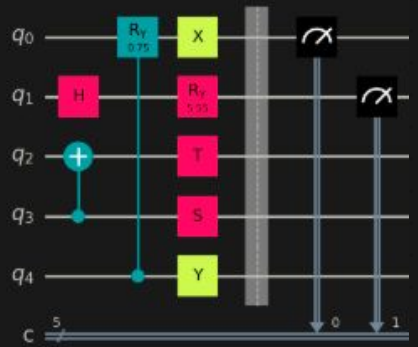
High-level Thinking

Designing a quantum algorithm

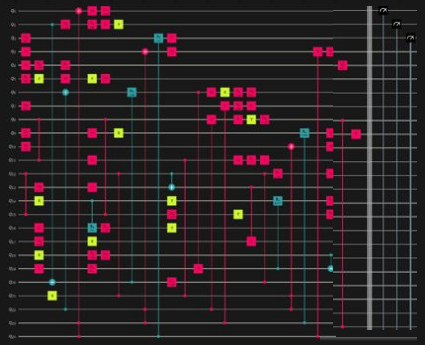
```
1   qfunc main(output x: qnum, output y: qnum) {
2     allocate<4>(x);
3     hadamard_transform(x);
4     y = (x ** 2) + 1;
5   }
```

The quantum software challenge

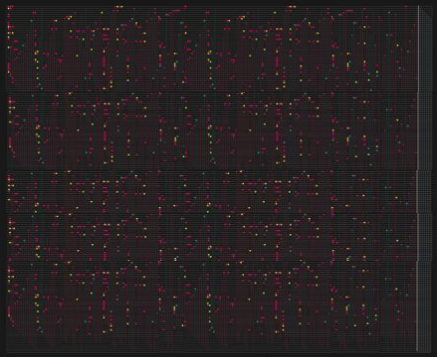
It is impossible to design large quantum circuits using today's development methods



Doable



Very complex



Impossible

The Challenges in Practice

From Model To Execution

Model

- From low-level to high-level
- Bridging design details to abstractions

Optimization

- Hardware-agnostic algorithms design
- Resource efficiency
- Flexible compilation

Analyze

- Visualizing complex large-scale circuits
- Tracking data flow
- Identifying logical errors

Execute

- Coordinating multiple jobs in one environment
- Accessing diverse backends
  - Quantum HW
  - Simulators
  - Cloud providers

Classiq: World's First
Quantum Operating System

The Classiq Platform

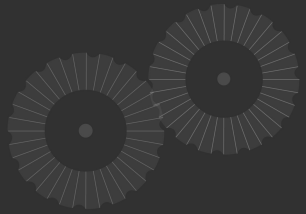
Design, Debug, Optimize			Execute
Model	Synthesis Engine	Visualizer	Executer
High level design of quantum algorithms	Synthesis the optimal quantum circuit	Interactive application for circuit visualization	Easy execution of quantum programs on all major quantum hardware

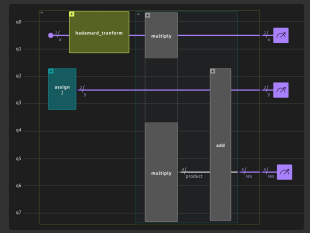
```
from classiq import *

@qfunc
def main(x: Output[QNum],
         y: Output[QNum],
         res: Output[QNum]):
    allocate(2, x)
    hadamard_transform(x)
    y |= 2
    res |= 3 * x + y

quantum_program = synthesize(main)
show(quantum_program)
```

✓ 3.3s Python





IBM Google

CLASSIQ

Practice!
Hands-on is the key!

- **Go to:** docs.classiq.io
- **Getting Started**
  - Register with your academic email
  - Review the Onboarding Tutorial
- Open the **Classiq Platform**
  - Start coding with our new **Classiq Studio**
  - tutorials/basic_tutorials/the_classiq_tutorial/**classiq_overview_tutorial**.ipynb
- **Check also:** Qmod Tutorial - Part 1 + 2, **Synthesis** and **Execution** Tutorials

Live Demo – What Is Classiq?

Thank you!

CLASSIQ