

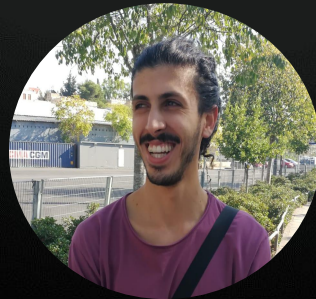
# End-to-End Quantum Software Development with CLASSIQ

Quantum Software Development Journey: From Theory to Application with Classiq - Part 1





# Personal Introduction



**Nadav Ben-Ami**

Technical Quantum Community  
Intern

B.Sc in Physics (Tel-Aviv Uni.)  
During gap year before QST Master's  
R&D experience in industry & academia

# Program Overview

## Quantum Software Development Journey: From Theory to Application with Classiq

- **Week 1: Introduction to the Classiq Platform & High-Level Functional Design**
- Week 2: Git & Software Development Skills
- Week 3: VQE and Introduction to Quantum Machine Learning
- Week 4: QNN and Advanced Applications

# Session Overview

## Introduction to the Classiq Platform & High-Level Functional Design

### Quantum Computing - 30 min

- Introductions
- Presentation of Classiq

### Classiq Hands-On - 60 min

- Hands-On Workshop - Basics, State Preparations, Arithmetics
- Q&A and Summary

**Classiq** is a quantum software tool that  
enables you to

**Design**

**Optimize**

**Analyze**

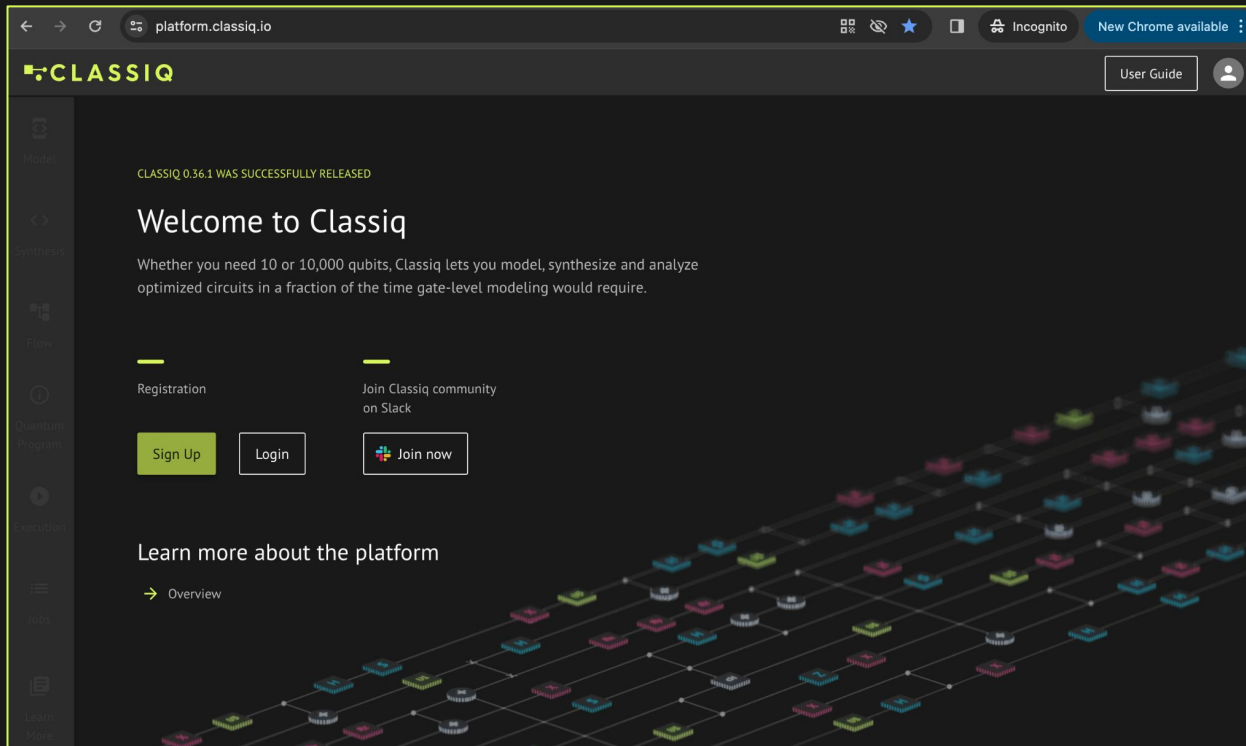
**Execute**

quantum algorithms

# Getting Started in 1 Minute !

Go to [platform.classiq.io](https://platform.classiq.io) and sign up

Install the Classiq Python SDK package



```
pip install -U classiq
```

←

→

↺

platform.classiq.io

CLASSIQ

User Guide

E

Model

Synthesis

Flow

Quantum Program

Execution

Jobs

Learn More

Slack

CLASSIQ 0.37.0-DEV.2 WAS SUCCESSFULLY RELEASED

Hello, Eden

Create quantum programs with Classiq

The Classiq Platform guided demo - learn the basics!

Join Classiq community on Slack

Build your first quantum model

Model examples

Walkthrough

Join now

New model

VQE

QSVM

HHL

Grover Search

Max K-Vertex Cover

from classiq import \*

See all examples

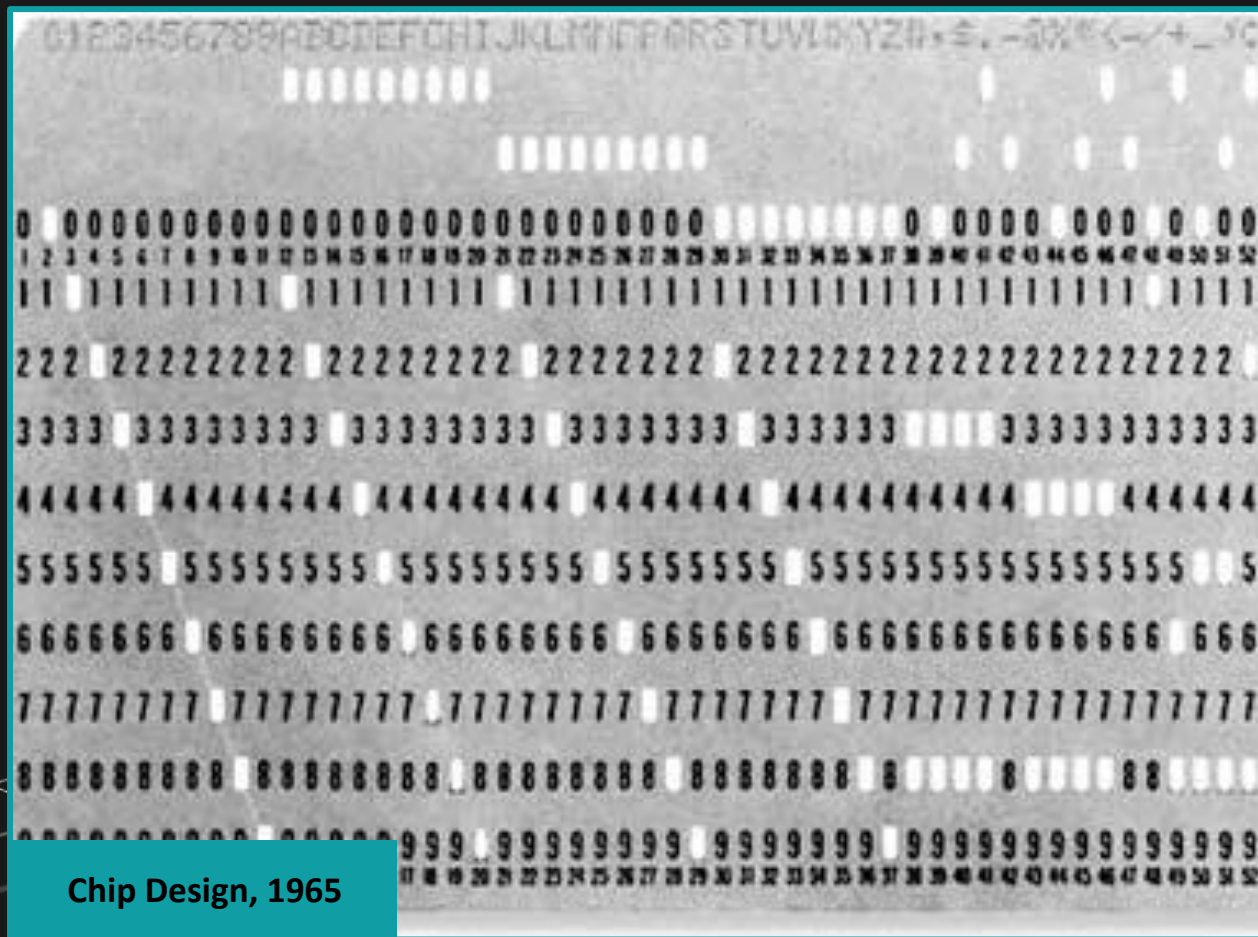




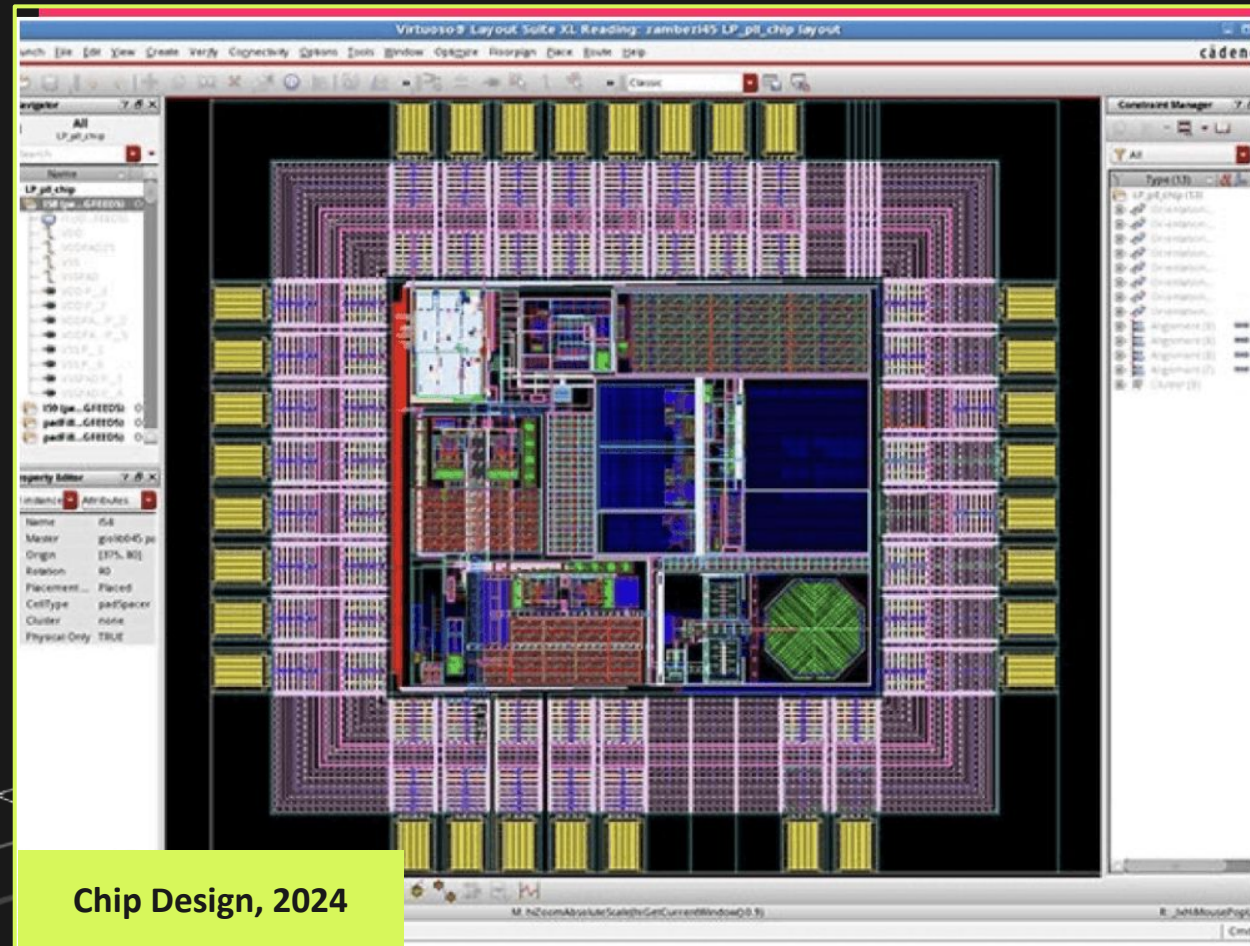
...Years ago **60**



# Electronic Design Challenge

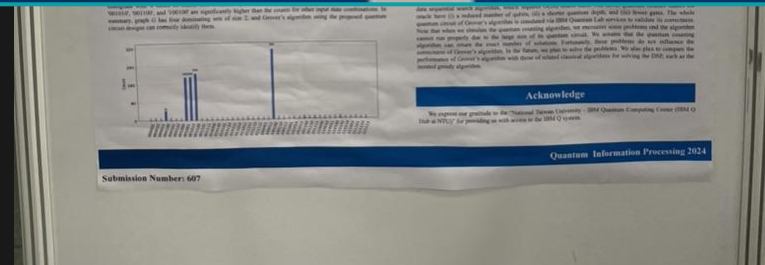
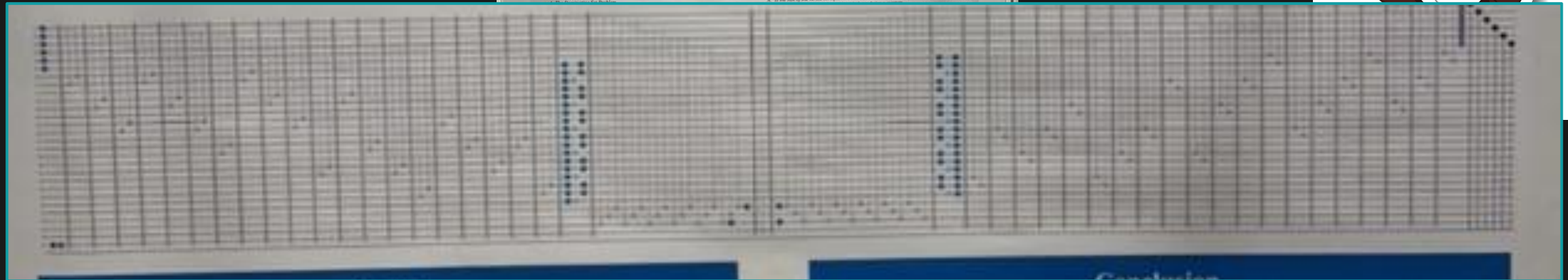
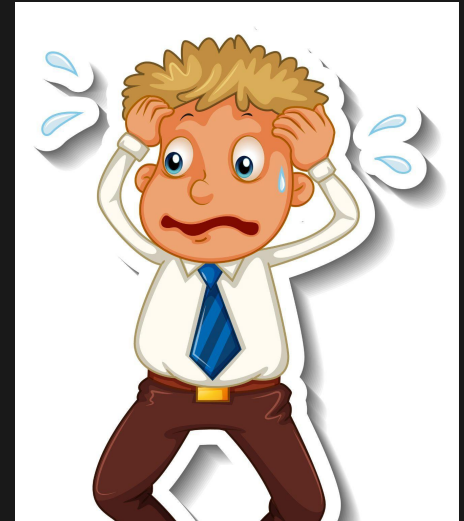
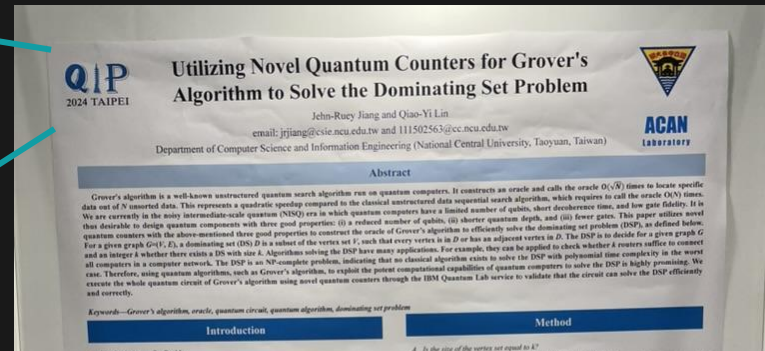


# Electronic Design Automation





# Design & Implementation of Novel Quantum Algorithms Before Classiq...



# Design of Novel Quantum Algorithms with Classiq

The screenshot displays the CLASSIQ IDE interface. A prominent red diagonal banner across the center reads "HIGH LEVEL FUNCTIONAL DESIGN". The interface is divided into several sections:

- Top Bar:** Includes the CLASSIQ logo, a "User Guide" button, and a user profile icon.
- Left Sidebar:** Contains navigation options: Model, Synthesis, Flow, Quantum Program, Execution, Jobs, Learn More, and Slack.
- Main Workspace:**
  - Models:** Displays a search bar and a list of models, including "Preparing QSVT for fixed-point amplitude amplification".
  - Synthesis Configuration:** Shows a "Synthesize" button and a "Constraints" section with a "Max Width" of 25.
  - Flow:** Displays a quantum circuit diagram with an "Adder" block.
  - Quantum Program:** Shows a code editor with the following code:
 

```
In [1]: from typing import List
          from classiq import QuantumProgram
          from classiq.qm import QIdentity, QZ, QH, QCallable, QParam, QX, allocate, apply_to_all, control,
```
  - Execution:** Shows a "Synthesize" button and a "Jobs" section.
  - Jobs:** Displays a list of jobs, including "Adder".
  - Learn More:** A button to access additional resources.
  - Slack:** A button to connect to the CLASSIQ Slack channel.
- Bottom Status Bar:** Indicates the current page.

# Design of Novel Quantum Algorithms with Classiq

```
52 qfunc main(output a: qbit[3], output b: qbit[3]){  
53     allocate<3>(a);  
54     allocate<3>(b);  
55     packed: qbit[];  
56     {a, b} -> packed;  
57     my_grover_search<7, 1, lambda<num_qubits>(oq) {  
58         my_oracle<lambda<num_qubits>(vars, result) {  
59             my_predicate(vars[0:2], vars[2:4], vars[4:7], result);  
60         }>(oq);  
61     }>(packed);  
62     packed -> {a, b};  
63 }
```

The screenshot displays the Classiq IDE interface, which is used for designing and synthesizing quantum algorithms. The interface is divided into several panels:

- Code Editor:** The central panel shows the quantum circuit code. The code defines a `my_grover_operator` function and a `main` function that uses `my_grover_search` and `my_oracle` to perform a Grover search. The code is written in a quantum-specific language that integrates with Qiskit.
- Models Panel:** Located on the left, it provides a search bar and a list of pre-defined quantum applications and models. The 'Grover Search' model is currently selected.
- Running Panel:** On the far left, it shows the status of the synthesis process, including a progress bar and a 'Running' indicator.
- Synthesis Configuration Panel:** On the right, it allows users to configure the synthesis process. It includes sections for 'CONSTRAINTS' (Max Width, Max Depth, Max Gate Count) and 'PREFERENCES' (Backend service provider, Backend name, Basis gates, Connectivity map, Symmetric connectivity, Output format, Transpilation option).

The interface is designed to be user-friendly, with a clear layout that separates the code editing from the configuration and execution details.

# Optimization of Novel Quantum Algorithms with Classiq

**PREFERENCES**

Synthesis Engine Version  
latest

Backend service provider

Backend name

Basis gates

Connectivity map  
Add

☒ Symmetric connectivity

Output format

Transpilation option

Timeout (sec)

Optimization Timeout (sec)

**Synthesis Configuration**

Reset Synthesize

**CONSTRAINTS**

Max width

Max depth

Max gate count  
Add

Optimization Parameter

Optimization parameter

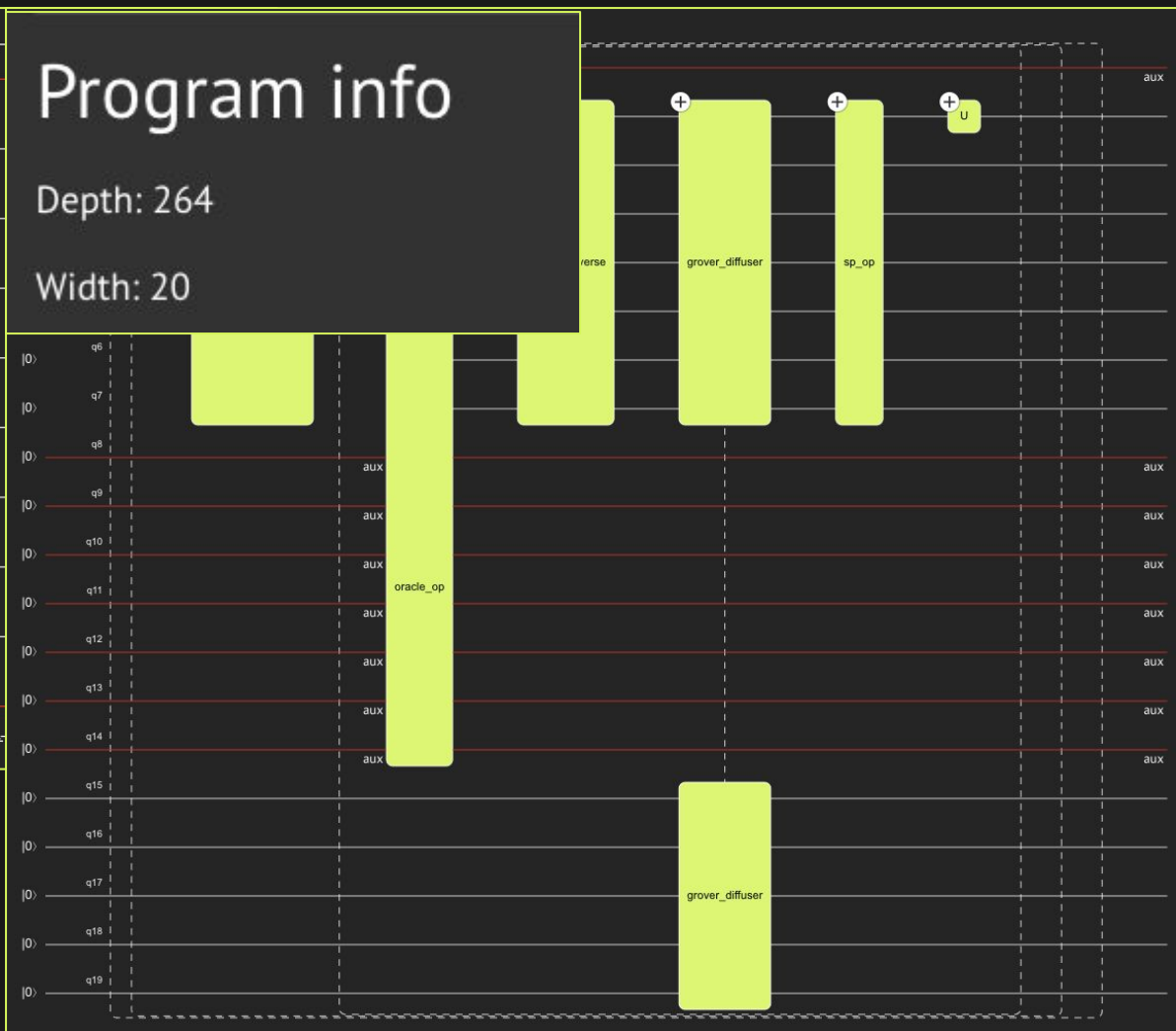
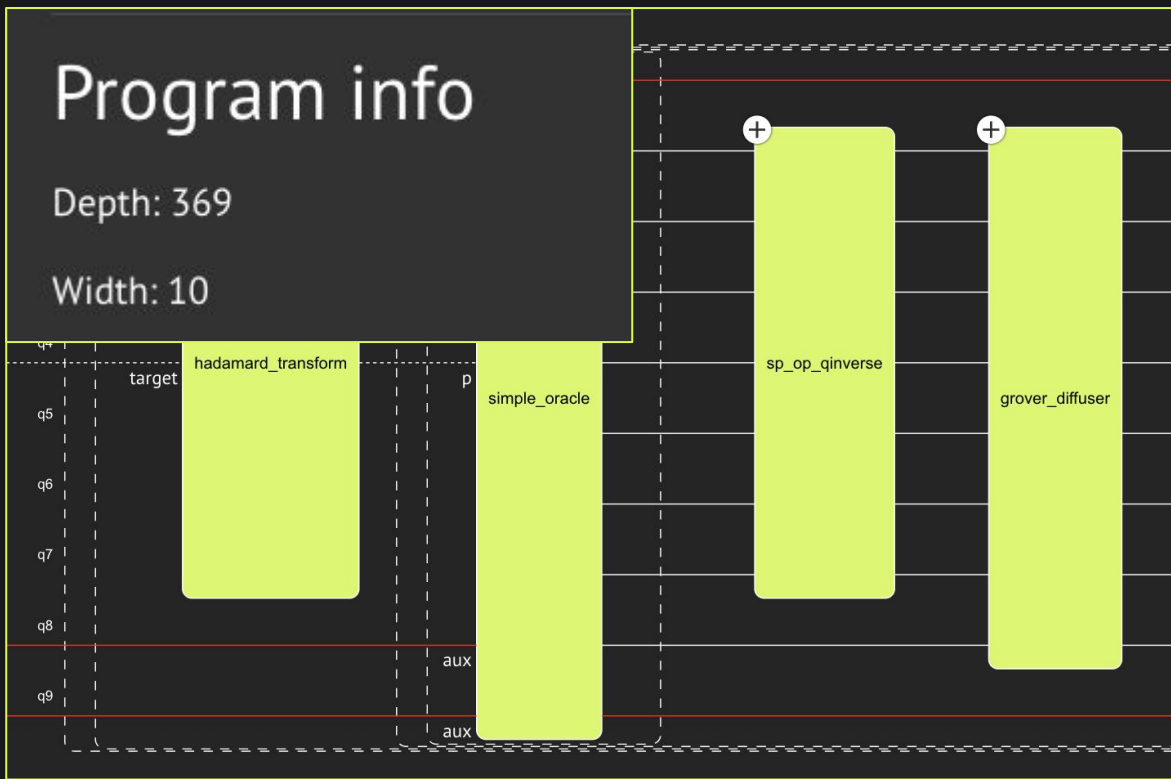
width

depth

no\_opt

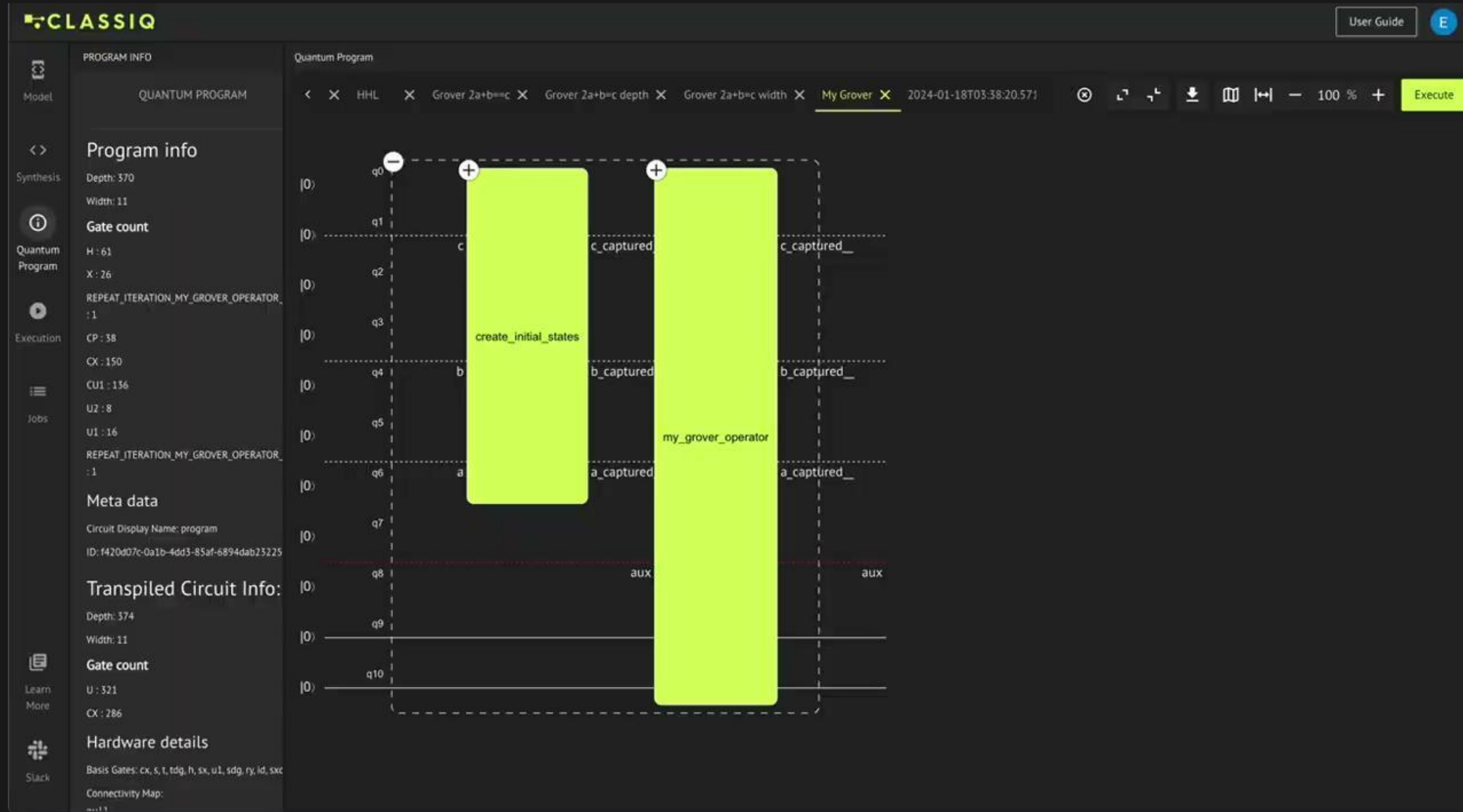
```
synthesize(create_model(main))
```

# Optimization of Novel Quantum Algorithms with Classiq

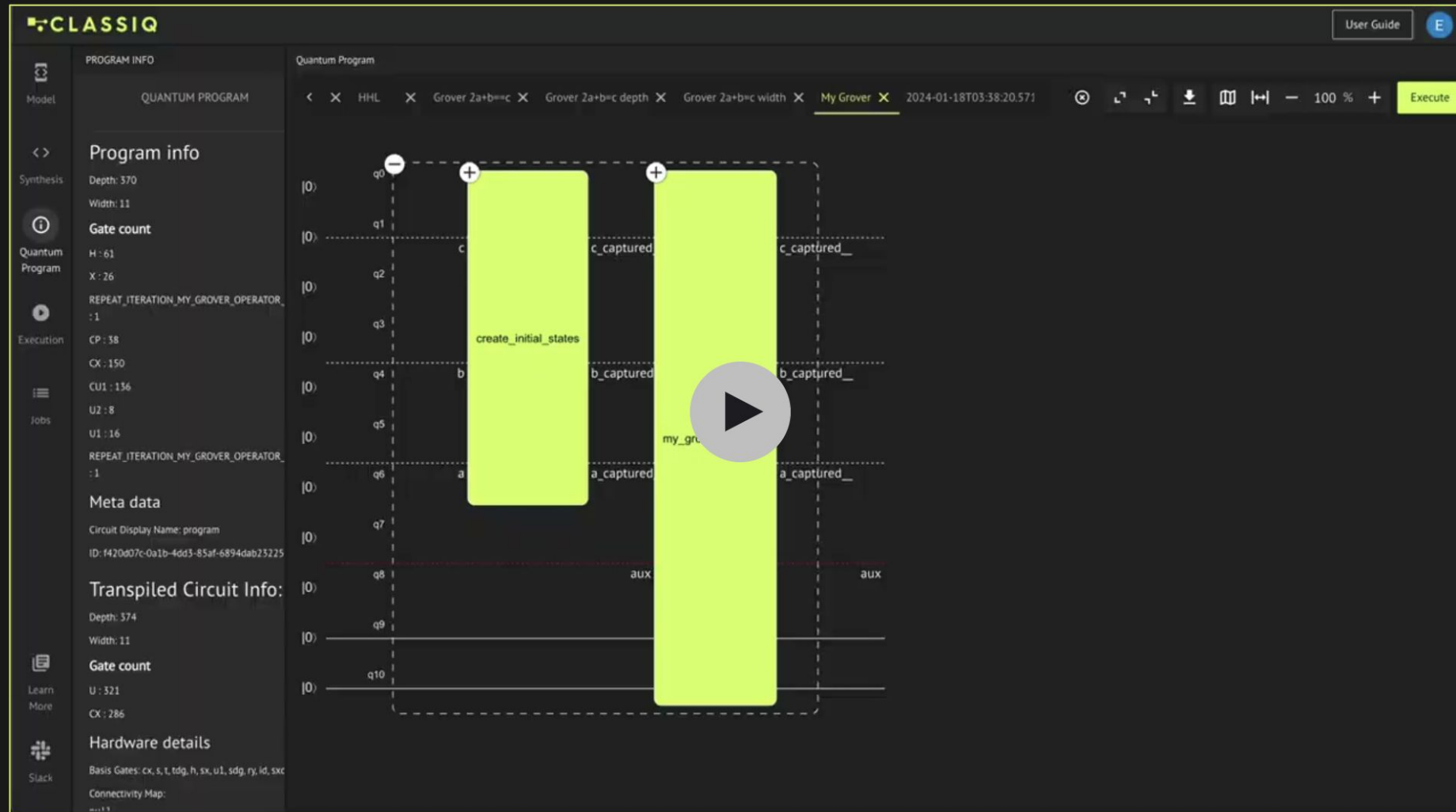




# Analysis of Novel Quantum Algorithms with Classiq



# Analysis of Novel Quantum Algorithms with Classiq



# Execution of Novel Quantum Algorithms with Classiq

CLASSIQ

Model

Synthesis

Quantum Program

Execution

Jobs

Learn More

Slack

EXECUTE QUANTUM PROGRAM

Execute Quantum

Quantum Program

My Grover

Vendor

All

Classiq

IBM Quantum

Azure Quantum

Amazon Braket

IonQ

Google

Classiq

☐ aer\_simulator\_ma

☐ nvidia\_state\_vecto

System provider: Classiq

System Status: Online

Max qubits: 30

☐ aer\_simulator\_der

☐ aer\_simulator\_sta

☒ aer\_simulator

IBM Quantum

IBM Quantum

☐ ibm\_nazca

☐ ibmq\_qasm\_simulator

☐ ibm\_cleveland

Amazon Braket

☐ Aria 2

☐ Forte 1

☐ Aspen-11

Azure Quantum

☐ ionq.qpu

☐ ionq.qpu.aria-1

☐ ionq.qpu.aria-2

☐ ionq.simulator

☐ quantinuum.qpu.h1-1

☐ quantinuum.sim.h1-1sc

IonQ

☐ simulator

☐ qpu.harmony

☐ qpu.aria-1

☐ qpu.aria-2

☐ qpu.forte-1

CLASSIQ

Model

Synthesis

Quantum Program

Execution

Jobs

Learn More

Slack

JOB HISTORY

MY GROVER - NOT EQUAL

1/18/2024

Quantum program executed: Fe225cba-Ee6d-4692-Ab69-D6b1e9ae7715

Job ID: 41957b62-D66a-4c75-9fba-9a499063f404

Vendor: Classiq

Backend name: Aer\_simulator

DateTime completed: Jan 18, 2024, 1:31 PM

Num shots: 2048

Status: Completed

96AF23FF-9955-42BB-82F4-7EF7A3319C77

1/18/2024

MY GROVER 2A+B=C

1/17/2024

MY GROVER

1/17/2024

2A+B=C WIDTH

1/17/2024

2A+B=C DEPTH

1/17/2024

2A+B=C BUILT IN GROVER

1/17/2024

GROVER ZERO

1/11/2024

GROVER WITH 0

1/11/2024

GROVER WITH APPLY NOT

1/11/2024

FEC9957D-1BCA-46AA-881D-A77E07D0AE3F

1/8/2024

E0190E07-57D1-417A-B580-50F48D568AA2

1/6/2024

F50D12F1-E4DE-4079-A183-C45E1F551CC7

1/5/2024

PERCISION 5 SEG 2

12/31/2023

SEGMENTS2 PERCISION 6

12/31/2023

OLD SDK

12/31/2023

Results

My Grover - not equal

Running on aer\_simulator

MEASUREMENT RESULTS (COUNT)

119

a=0, b=3, c=3

0000000

1111111

```
execute(quantum_program).result()
```

# Agenda

- What, Why and How Classiq ?
- Why functional building blocks?
- Hands-on Classiq's Python SDK workshop

# Agenda

- What, Why and How Classiq ?
- **Why functional building blocks?**
- Hands-on Classiq's Python SDK workshop

# Why Functional Building Blocks?

Robin Kothari

Staff Research Scientist  
Verified email at robinko.

Quantum algorithms quantum  
quantum computing query complex

Citations

3141



**HIGH LEVEL FUNCTIONAL DESIGN**

*“Don’t count the number of quantum algorithms you know, focus on the algorithms’ primitives you learn”*

R. Kothari @QIP24

## How to put this into practice

When reading a quantum algorithms paper:

- Identify the primitives. Are they classical or quantum primitives?
- Try to express the algorithm at high level in terms of primitives (E.g., HHL = Phase Estimation + Hamiltonian simulation + Amplitude Amplification)
- “upgrade” any classical primitives to speed up the algorithm?
- Primitives appear in many other algorithms?

Quantum AI



2024 TAIPEI

# Why Functional Building Blocks?





# Session Overview

- What, Why and How Classiq ?
- Why functional building blocks?
- **Hands-on Classiq's Python SDK workshop**
  - Join Classiq slack channel [here](#)
  - Go to the channel [#qcourse-self-study-module-2024](#))
  - Download the files and engage during the hands-on part!

 CLASSIQ

# THANK YOU

 CLASSIQ.IO

