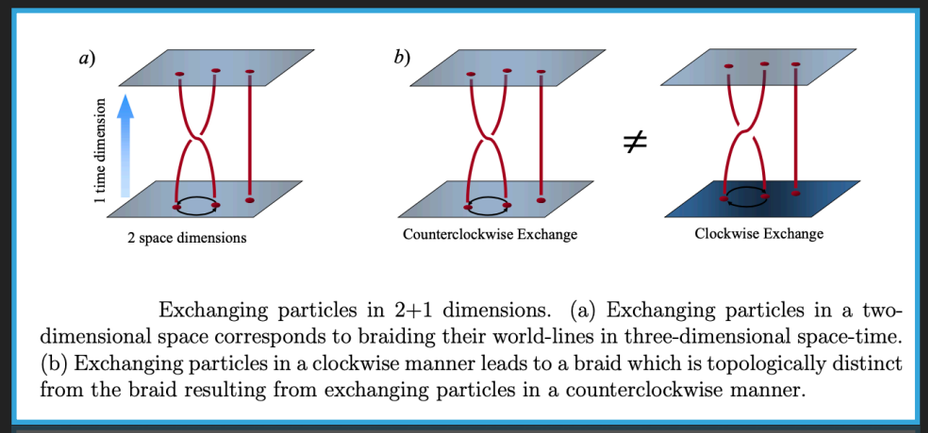
ARAHANT ASHOK KUMAR

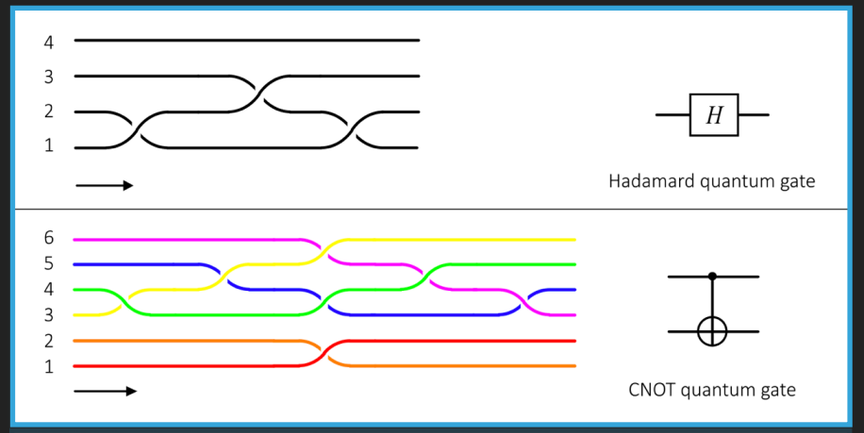
TOPOLOGICAL QUANTUM COMPILER

TOPOLOGICAL QUANTUM COMPUTING (TQC)

- In TQC, Quasiparticles such as Majorana fermions are braided in specific patterns and sequences to obtain Quantum gates
- A braid involves 2 particles being wrapped around each other…



Exchanging particles in 2+1 dimensions. (a) Exchanging particles in a two-dimensional space corresponds to braiding their world-lines in three-dimensional space-time. (b) Exchanging particles in a clockwise manner leads to a braid which is topologically distinct from the braid resulting from exchanging particles in a counterclockwise manner.



- In TQC, all the quantum gates depend only on the braid pattern. This would heavily suppress errors already at the level of the hardware, with little need for resource intensive quantum error correction.
- Physically, each of these braids are realised through movements of these particles on a Nanowire (Quantum Hardware)

CLASSICAL COMPILER

Code execution in current classical computers

- Compiler - Source code to Assembly language
  - Lexer - text into tokens
  - Parser - tokens into a Parse Tree
  - Parse tree into Intermediate Code
  - Intermediate code to Assembly language
- Assembler - assembly language to Object files
- Linker - links multiple object files and strings them together into ONE executable file
- Loader - Loads the executable onto the main memory for execution

(TOPOLOGICAL) QUANTUM COMPILER

TQC QUANTUM COMPILER ARCHITECTURE

<table>
<tr><td rowspan="3">Quantum Compiler</td><td colspan="2">TQC Library</td></tr>
<tr><td>Restricted APIs</td><td>Modules</td></tr>
<tr><td colspan="2">System calls</td></tr>
<tr><td></td><td colspan="2">Assembly instructions</td></tr>
<tr><td rowspan="2">Programmable Quantum Material</td><td colspan="2">Programmable QM</td></tr>
<tr><td>H/W config</td><td>APIs</td></tr>
<tr><td></td><td colspan="2">Quantum Material (QM)</td></tr>
</table>

TOPOLOGICAL QUANTUM COMPILER STAGES

- Preprocessing - It converts the given inputs into machine readable formats
  - Nanowire - adjacency matrix
  - Preprocessing Positions - valid branch states for gates
- Topological Quantum Compiler Algorithm
  - Nanowire Rules - Physical constraints, Nanowire state, etc.
  - Phases - Validation and Braiding
- Output - Particle movement sequences, Nanowire state matrix
- Measurement
- Animation
    - Nanowire
    - Braiding

TOPOLOGICAL QUANTUM COMPILER

- Concepts used
  - Algorithms theory
  - Graph theory (Dijkstra's algorithm)
  - Staged Compiler architecture
  - Layered OS architecture
  - Object oriented programming
    - Inheritance
    - Polymorphism
  - Shell scripts
- Features
  - Algorithm correctness
  - Reusability
  - Staged architecture
  - Robustness
  - Object class design
  - Validation and graceful error handling

WORK DONE

- Repo: https://github.com/ShabaniLab/TQC-Braiding-Nanowire

- Stage 1 - Summer 2020
  - Compiler architecture
  - Preprocessing - Nanowire
  - Braiding algorithm
- Stage 2 - Fall (Sep) 2020
  - Rule 3 - Voltage regulation
  - Rule 7 - Braiding direction
- Stage 3 - Fall (Oct) 2020
  - Animation - Nanowire
  - Animation - Braiding
- Stage 4 - Fall (Oct) 2020
  - Measurement - Fusion
- Stage 5 - Fall (Oct) 2020
  - Pauli X gate (1 Qubit)
  - Phase S gate (1 Qubit)
  - Hadamard gate (1 Qubit)
- Stage 6 - Fall (Nov) 2020
  - Redesign Compiler architecture
- Stage 7 - Fall (Nov-Dec) 2020
  - Preprocessing - Positions
  - Quantum Circuit

STAGE 1 – PRE-PROCESSING – NANOWIRE

Given a Nanowire structure, an Adjacency matrix is constructed, which is used to determine the paths that the particles take in order to form a braid.

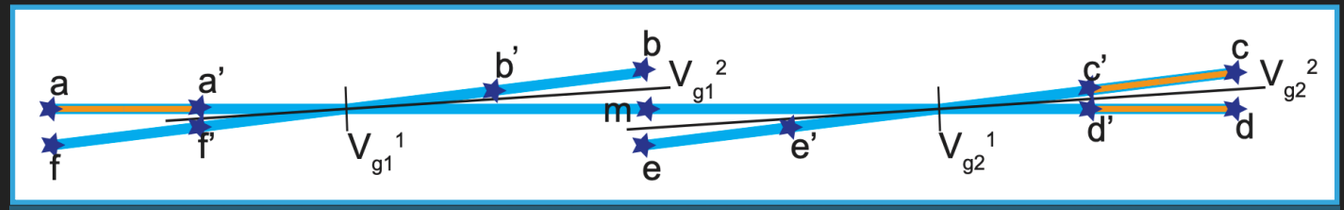
Representation of the Nanowire must follow certain rules:

- Sequence of the branches is Anti-clockwise, from Topmost and Leftmost
- Positions are sequenced Outward to inward
- Each intersection is followed by the Voltage gates

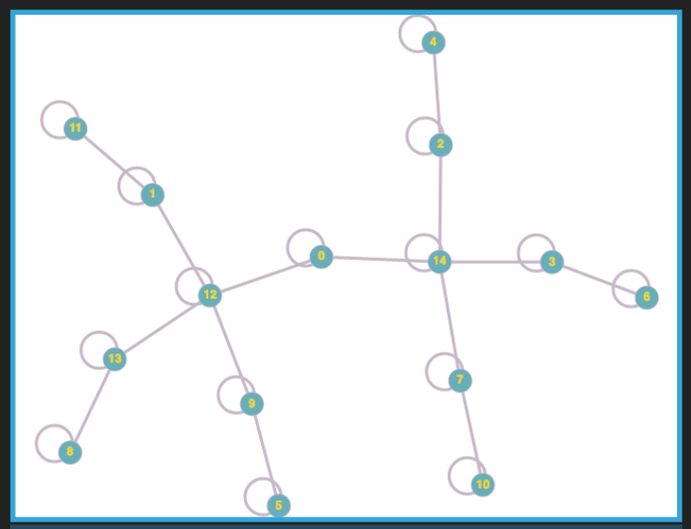
Nanowire Adjacency Matrix

1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	1	1	0	0	0	1	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	0	1	0	0	0	0
1	0	0	0	1	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	1	1	0
0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	1	1	0	0
0	1	1	0	0	0	0	0	0	1	0	1	1	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	1

Double-X junction Nanowire

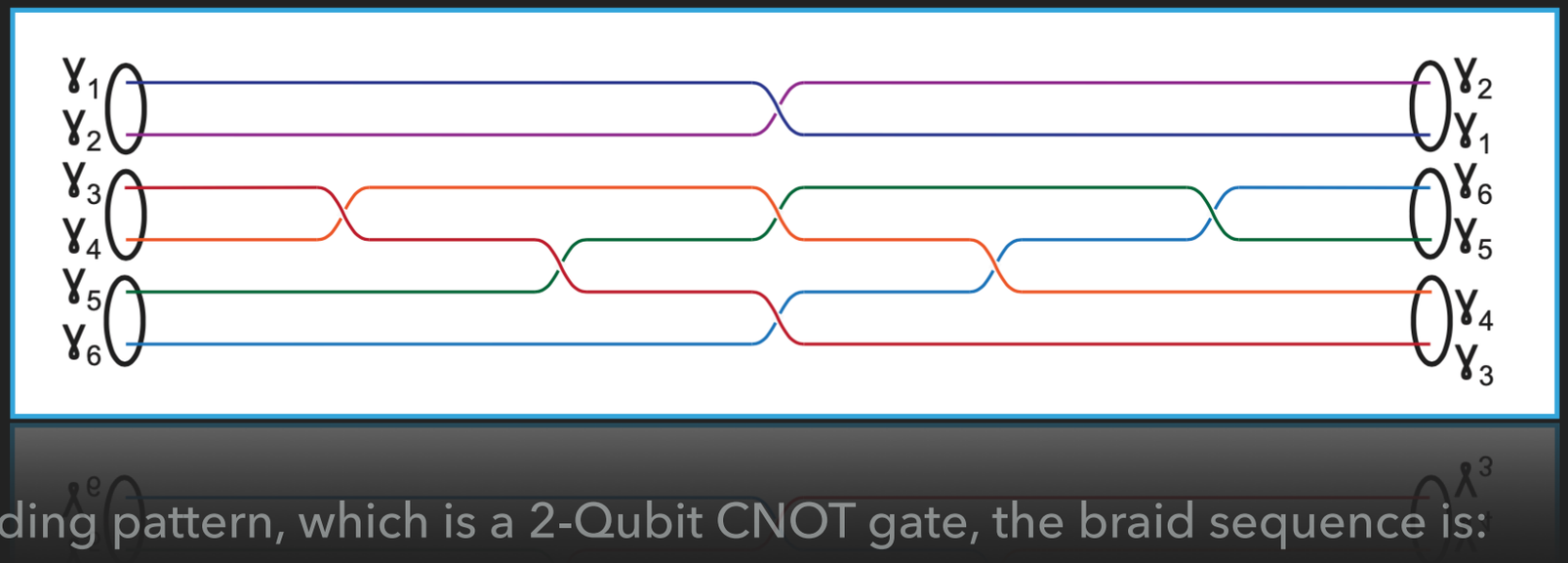


Reconstructed Nanowire Graph



PRE-PROCESSING - BRAID SEQUENCE

- Given a Braid pattern for a Quantum gate, it's processed into a sequence of braids of 2 particles.
- Each slice of this sequence is an atomic operation.



- For this Braiding pattern, which is a 2-Qubit CNOT gate, the braid sequence is:
  - (3, 4), (3, 5), (1, 2), (4, 5), (3, 6), (4, 6), (5, 6).

Initial Particle positions

The initial positions of the particles on the Nanowire, in the above example, are a,a',c,c',d,d'.

ALGORITHM – RULES

- Nanowire State validity - Each braiding must result in a valid nanowire state
- Particle-Zero mode isolation - No two particles from different zero modes can occupy adjacent positions on the Nanowire during the braiding operation.
- Non-intersecting zero-modes - The non-participating paired zero-mode particles cannot be on the same crossing.
  - This would involve voltage regulation.
- Braiding Rotation - Clockwise and anti-clockwise braiding.
- Braiding Sequence - The entire braiding procedure follows the given sequence
- Atomic Braiding operation - Each braiding slice involves 2 particles

- Braiding Concurrency limits - limit on the # concurrent braiding operations
- Intermediate positions - Every braiding operation puts the particles back in their respective final positions.

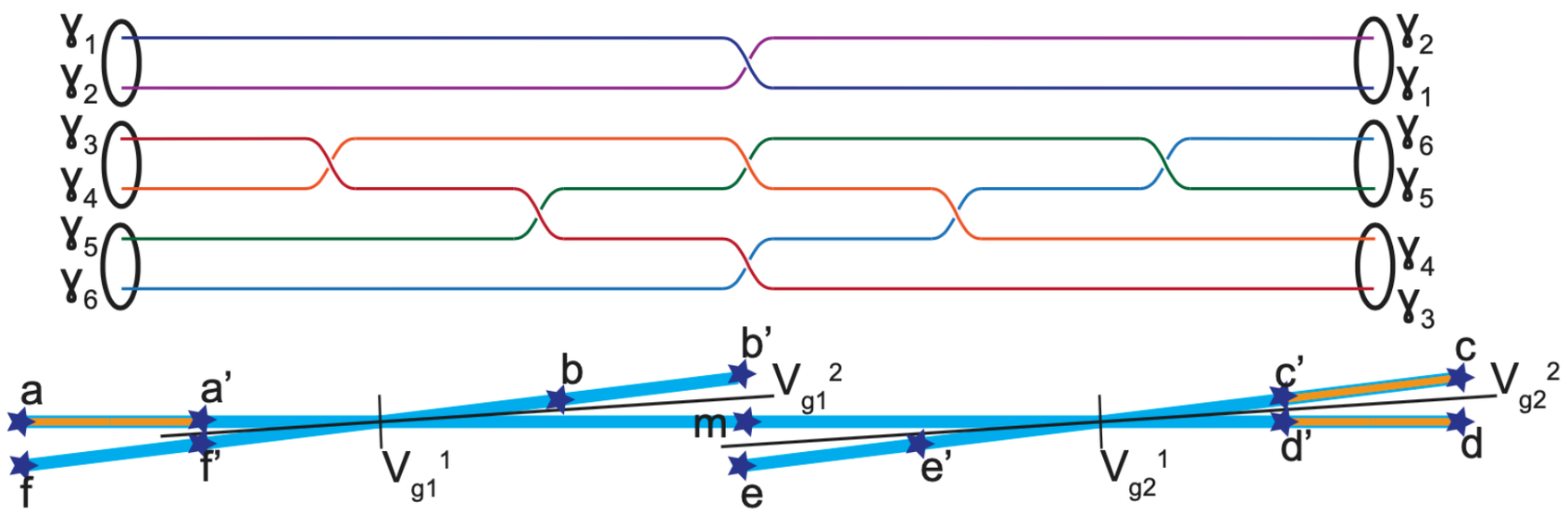
ALGORITHM – VALIDATION PHASE

- Validate Final Positions - Retrieve and Validate the expected final positions for the particles, after the Braiding operation is completed.
- Validate Intermediate Positions - Retrieve, validate and Rank the potential Intermediate positions for the participating particles.
  - 1st particle that moves gets the 1st empty branch. Later, specify rules/ restrictions.
- Ranking multiple positions - If there are more than one free branch, there can be multiple final positions, which are ranked based on their...
- Validate Resulting State - Nanowire validation algorithm which returns a score for every expected final/ intermediate position.

ALGORITHM – BRAIDING PHASE

- Retrieve involved branches - Retrieve the list of branches and its positions involved in the movement of the particle.
- Get Shortest Path - the shortest path for a particle from its current position to the given (valid) final/ intermediate position using Dijkstra's algorithm.
- *Get Voltage Changes - If braiding involves particles from different zero modes, perform necessary voltage gate changes.

OUTPUT – NANOWIRE STATE MATRIX

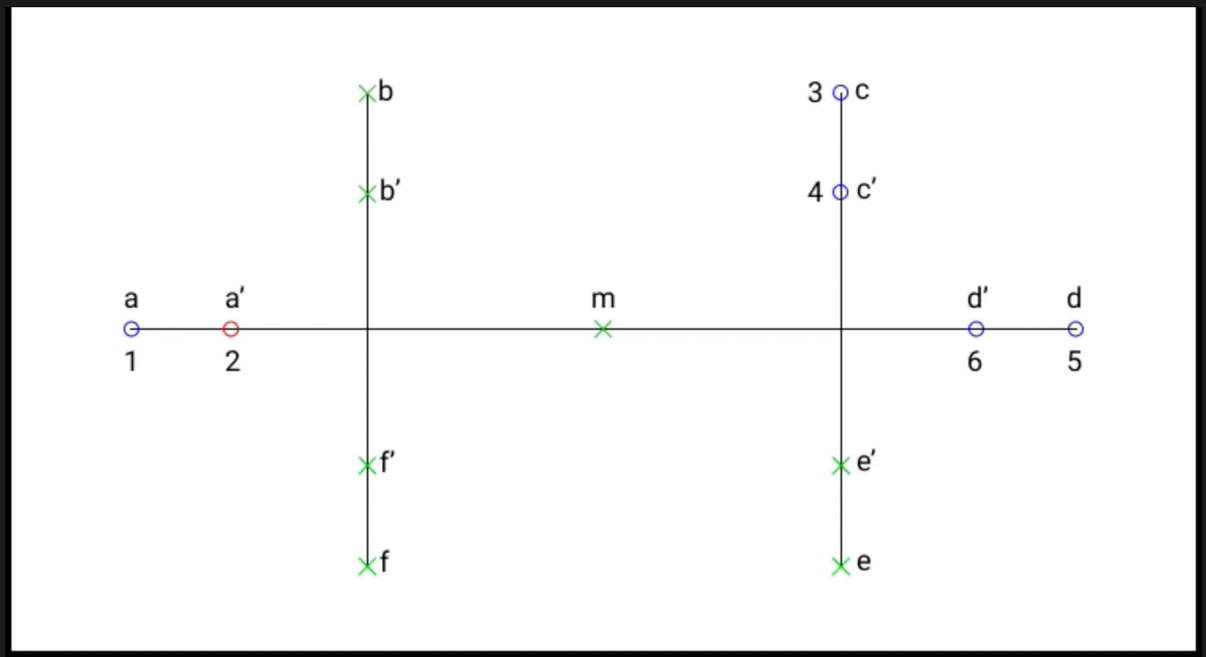


γ_1	a	a	a	a	a	a	a	a	**b**	b	**a'**	a'	a'	a'	a'	a'	a'	a'	a'	a'	a'	a'	a'	a'
γ_2	a'	a'	a'	a'	a'	a'	a'	**f**	f	**a**	a	a	a	a	a	a	a	a	a	a	a	a	a	a
γ_3	c	c	**m**	m	m	m	m	m	m	**d**	d	**e**	e	e	e	e	e	e	e	e	e	e	e	e
γ_4	c'	**e**	e	**c**	c	c	c	c	c	c	c	c	c	c	c	**d**	d	d	d	d	**e'**	e'	e'	e'	e'
γ_5	d	d	d	d	d	d	d	**c'**	c'	c'	c'	c'	c'	**m**	m	**c**	c	c	c	c	c	c	**m**	**d'**
γ_6	d'	d'	d'	d'	d'	**e**	e	e	e	e	**m**	m	**e'**	e'	e'	e'	e'	**m**	m	m	m	**d**	d	d
V_{g2}^1	o	o	o	o	o	o	**s**	s	**o**	o	o	o	o	o	o	o	**s**	s	**o**	o	o	o	o	o
V_{g2}^2	o	o	o	o	**s**	s	**o**	o	o	o	o	o	o	o	o	o	o	o	**s**	s	**o**	o	o	o

OUTPUT – NANOWIRE STATE MATRIX

Nanowire state matrix for Braiding between particles (1, 2)

a	a	a	a'	f'	f'	f'	f'	f'	a'
a'	b'	b'	b'	b'	b'	a'	a	a	a
d	d	d	d	d	d	d	d	d	d
c	c	c	c	c	c	c	c	c	c
c'	c'	c'	c'	c'	c'	c'	c'	c'	c'
d'	d'	d'	d'	d'	d'	d'	d'	d'	d'

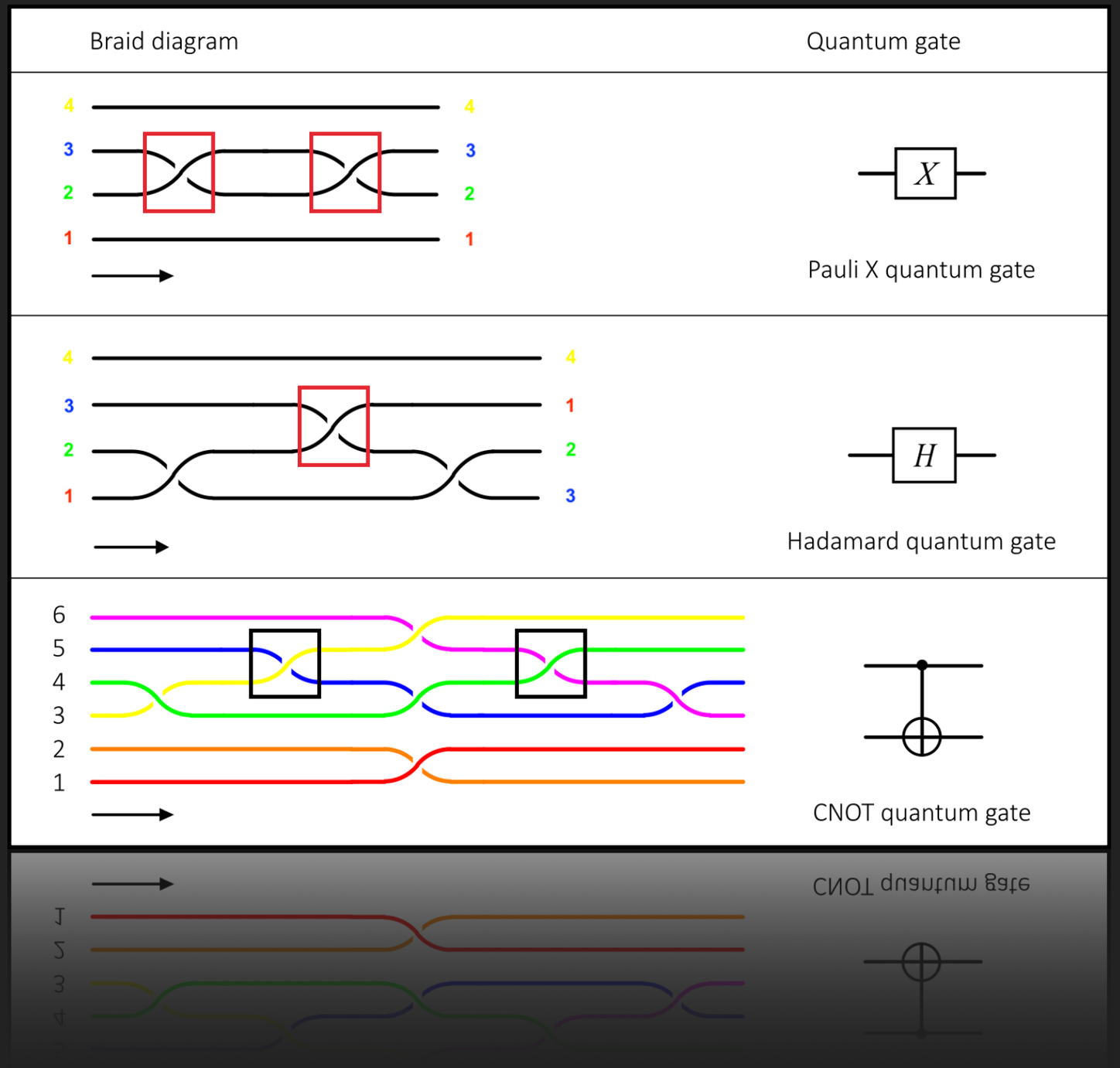


STAGE 2

- Rule 3 with Voltage Regulations
- For (category 2) braiding, which involves particles from different zero modes
- Gates
  - X - (2,3), (2,3)
  - Hadamard - (1,3)
  - CNOT - (3,5), (4,6)

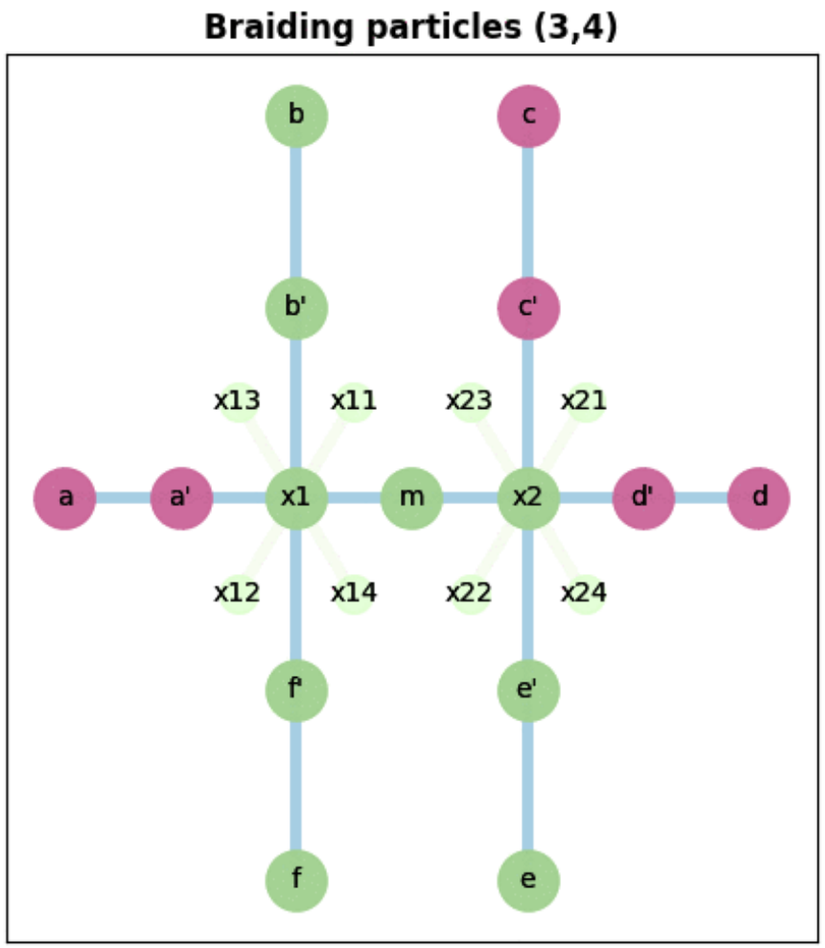
- Rule 7 - Braiding Direction.
- To account for clockwise and counter-clockwise braiding
- Default is counter-clockwise braiding

Gates which have category 2 braiding



STAGE 3

- Animation - Nanowire movement

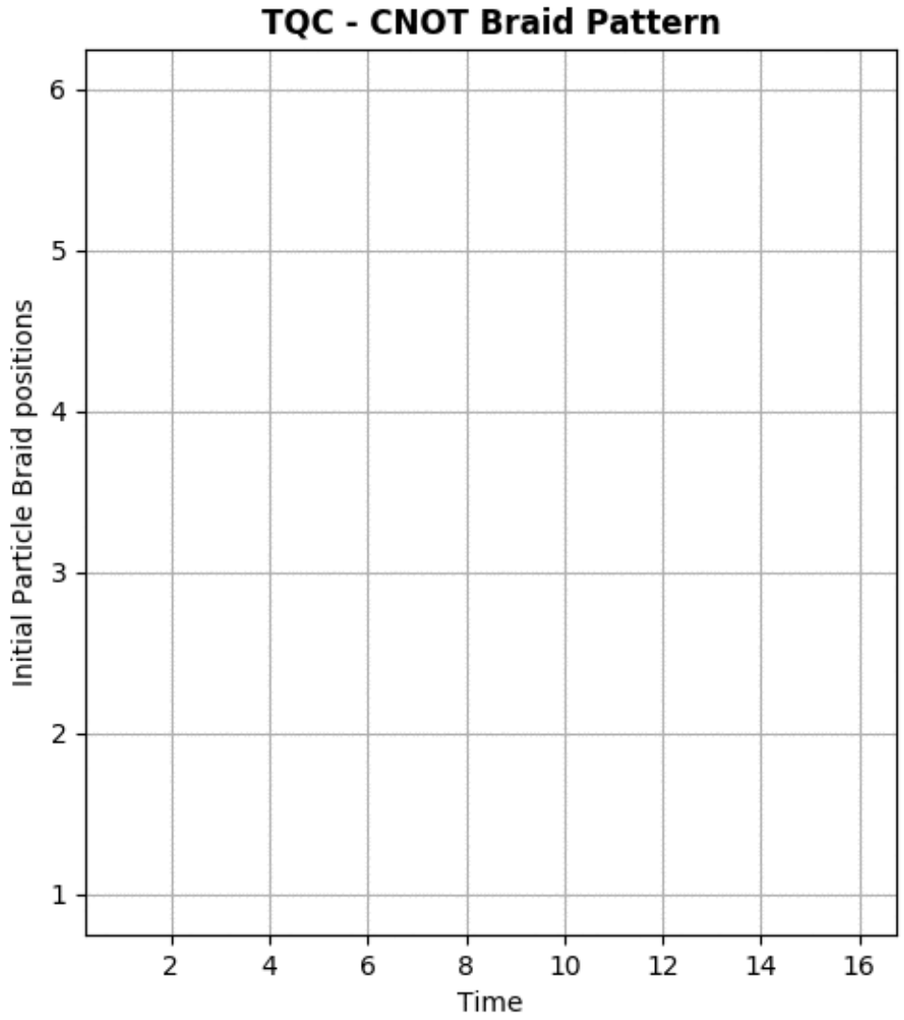


Braid table

Particle 1	Particle 2
3	4
3	5
1	2
4	5
3	6
4	6
5	6

STAGE 3

- Animation - CNOT Braiding



Braid table

Particle 1	Particle 2
3	4
3	5
1	2
4	5
3	6
4	6
5	6

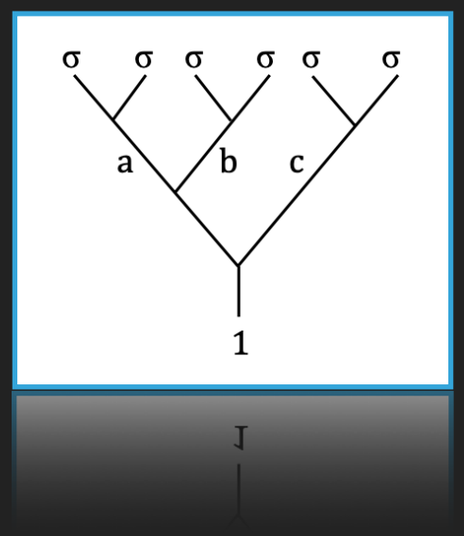
STAGE 4

Fusion Rules

P1	P2	Res
σ	σ	1
σ	σ	Ψ
1	σ	σ
σ	1	σ
1	1	1
1	Ψ	Ψ
Ψ	1	Ψ
Ψ	Ψ	1
Ψ	σ	σ

Fusion Channel

Q1	Q2	a	b	c
0	0	1	1	1
1	0	Ψ	Ψ	1
0	1	1	Ψ	Ψ
1	1	Ψ	1	Ψ

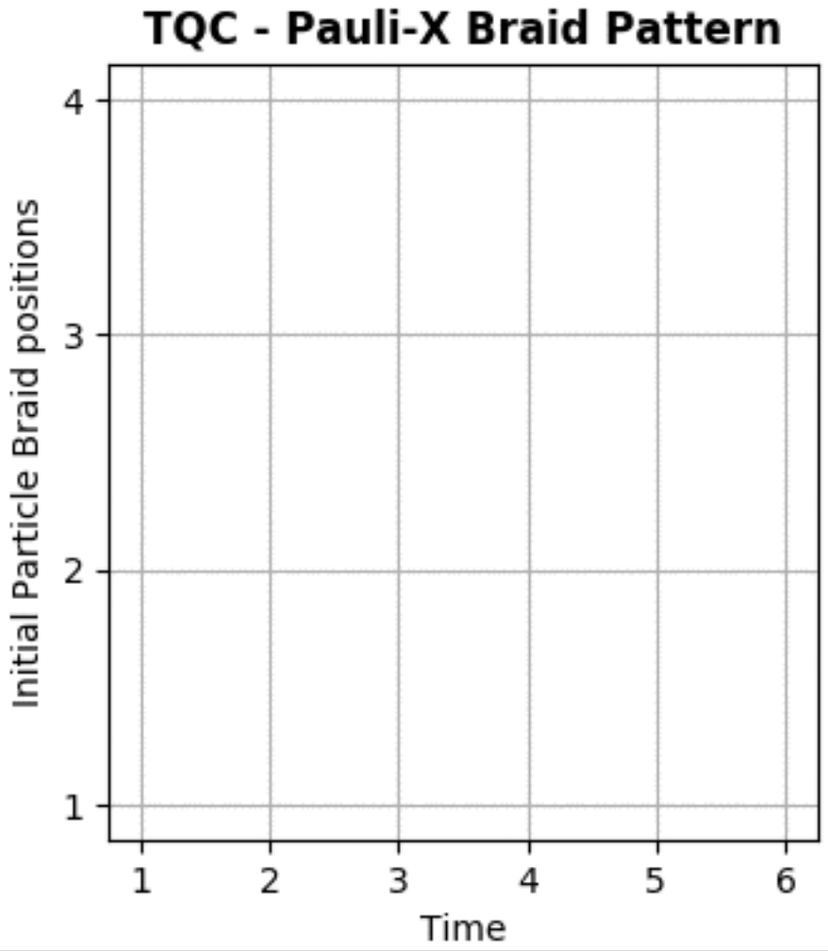


CNOT Fusion measurements

Pair-1	Pair-2	Pair-3	Qubit-1	Qubit-2
Ψ	1	Ψ	1	1
Ψ	Ψ	1	1	0
1	Ψ	Ψ	0	1
1	Ψ	Ψ	0	1
Ψ	Ψ	1	1	0
Ψ	Ψ	1	1	0
Ψ	1	Ψ	1	1
1	Ψ	Ψ	0	1
Ψ	Ψ	1	1	0
Ψ	1	Ψ	1	1

STAGE 5

- Pauli-X 1 Qubit gate

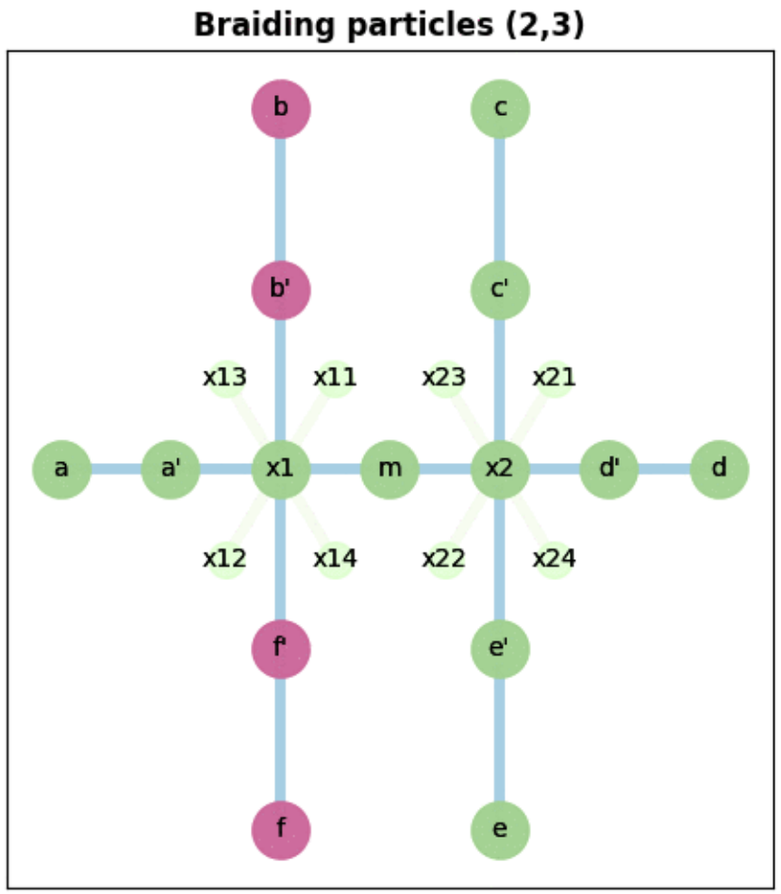


Braid table

Particle 1	Particle 2
2	3
3	2

STAGE 5

- Pauli-X 1 Qubit gate

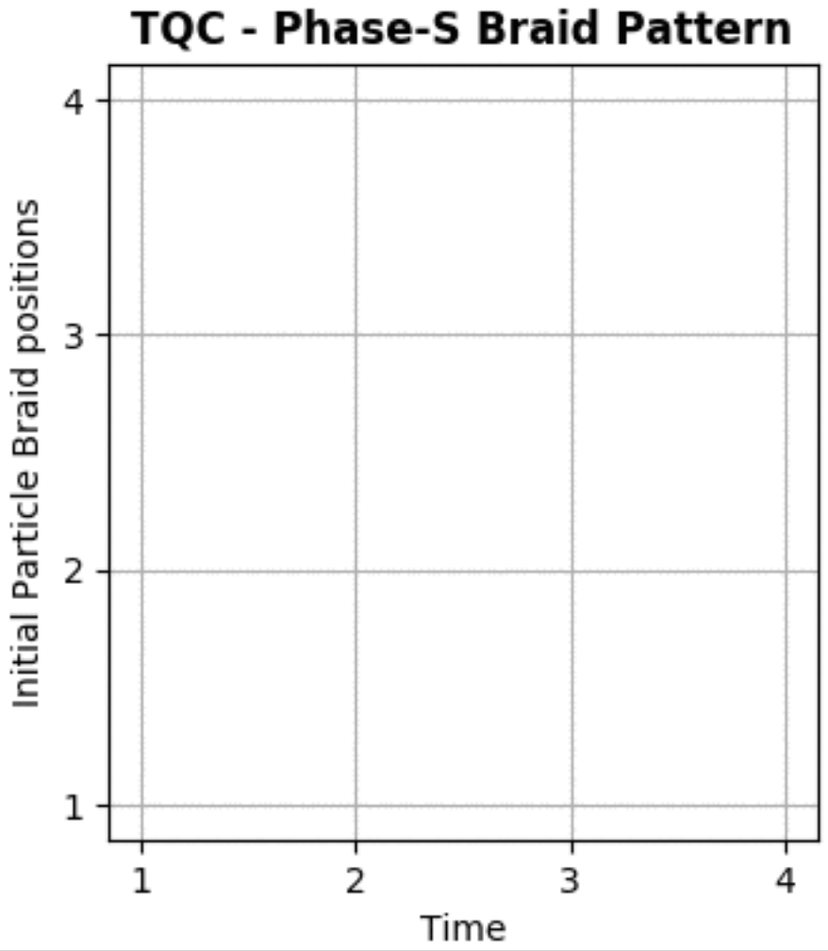


Braid table

Particle 1	Particle 2
2	3
3	2

STAGE 5

- Phase S 1 Qubit gate

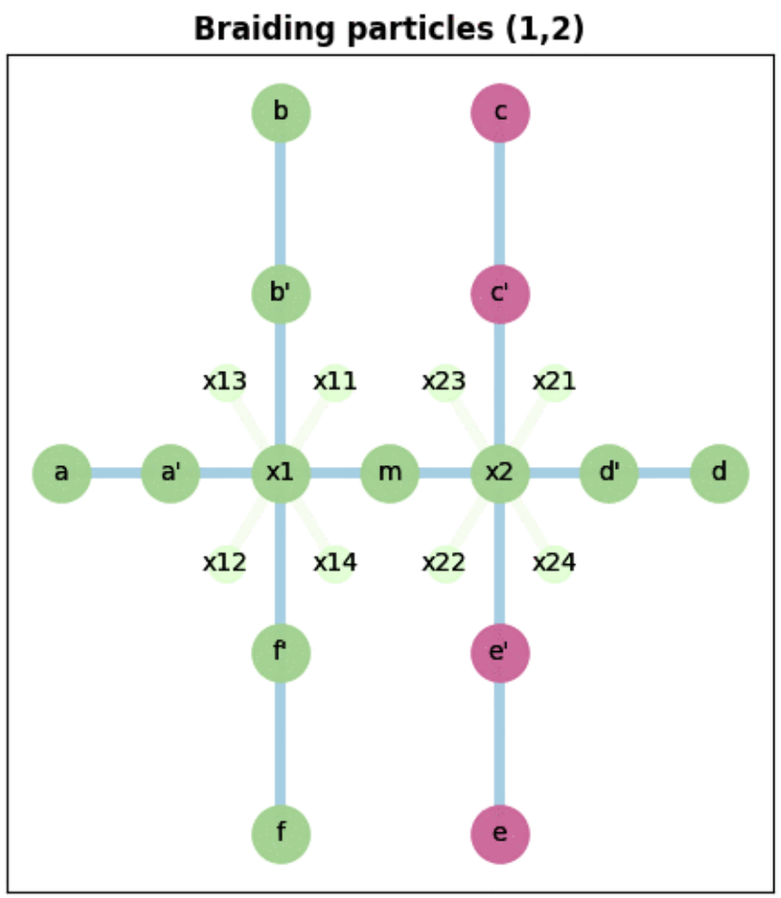


Braid table

Particle 1	Particle 2
1	2

STAGE 5

- Phase S 1 Qubit gate

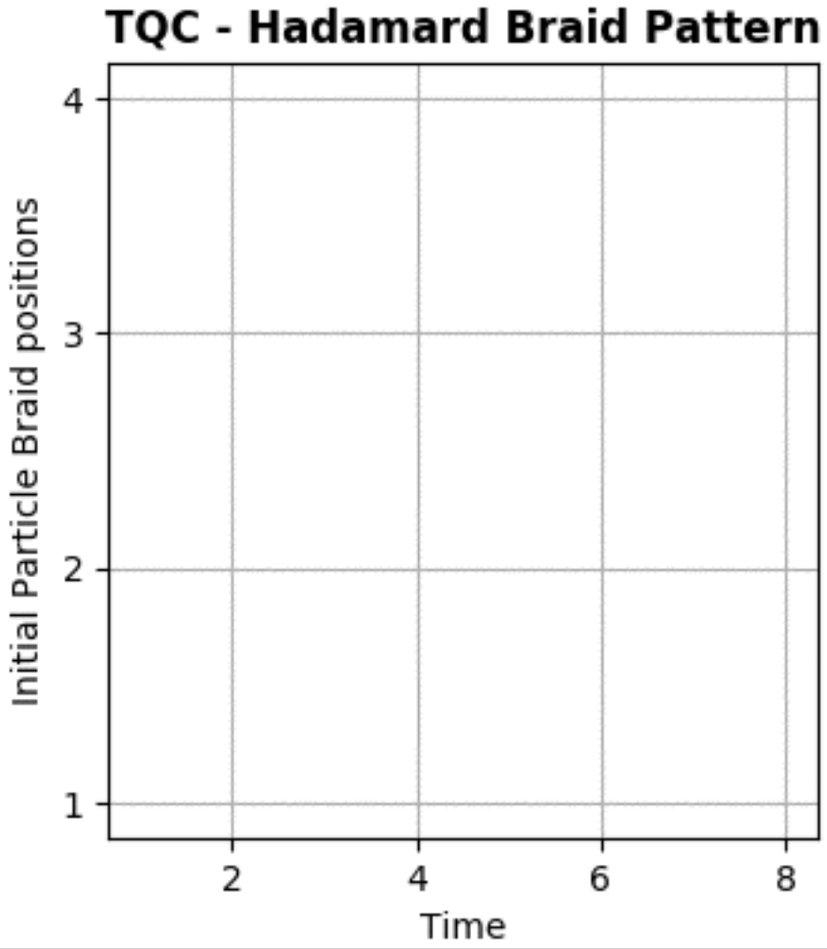


Braid table

Particle 1	Particle 2
1	2

STAGE 5

- Hadamard 1 Qubit gate

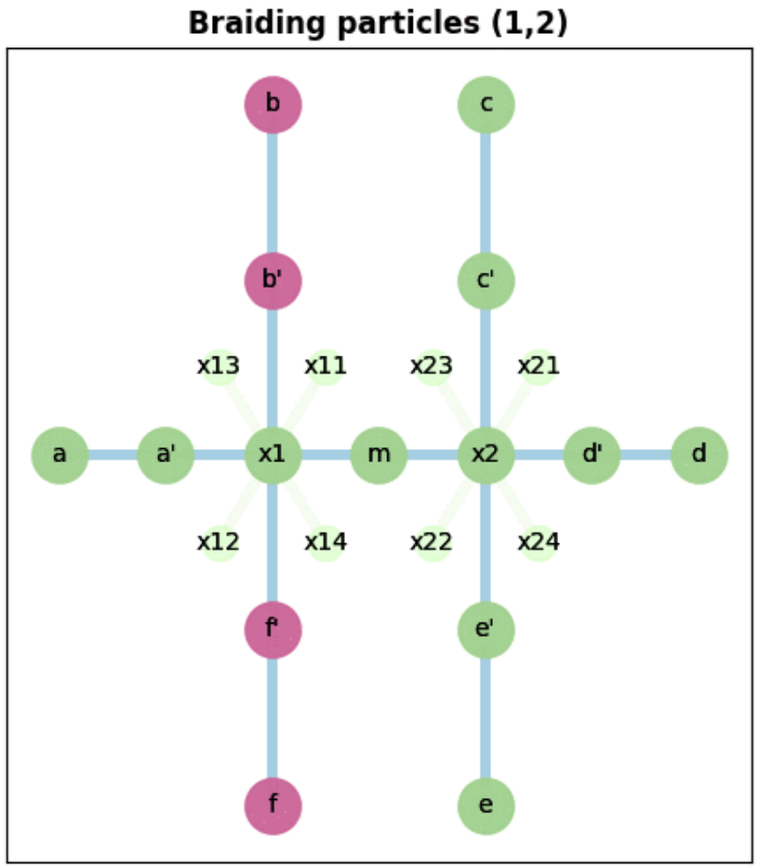


Braid table

Particle 1	Particle 2
1	2
1	3
2	3

STAGE 5

- Hadamard 1 Qubit gate



Braid table

Particle 1	Particle 2
1	2
1	3
2	3

STAGE 6 – TQC COMPILER ARCHITECTURE

- Package
  - Nanowire
  - Graph
  - Exception
  - Compiler
  - Braiding
  - Utility
  - Validation
  - Metrics
  - Measurement
  - Animation
- Implementation
  - Shell script
  - Preprocessing - Nanowire
  - Algorithm - Compile
  - Algorithm- Measure
  - Animation
- Inputs
  - Circuit configuration
  - Nanowire structure
  - Nanowire positions
  - Initial particle positions
  - Braid sequence
  - Fusion Channel
  - Fusion rules
- Outputs
  - Nanowire matrix
  - Nanowire vertices
  - Nanowire state matrix
  - Particle movements - Nanowire
  - Particle positions - Nanowire
  - Particle positions - Braid
  - Measurements
  - Braid animation
  - Nanowire animation

STAGE 7 – PREPROCESSING POSITIONS

- Valid branch config for gates
  - CNOT - Double, Clockwise, Adjacent
  - Hadamard - Double, Opposite
  - Pauli-X - Double, Opposite
  - Phase-S - Single, N/A
- Moving the particles
  - Zero mode orientation is preserved
    - standalone - hadamard and cnot
  - Zero mode orientation is reversed
    - Adds an extra braid
    - cnot gate in entanglement circuit.

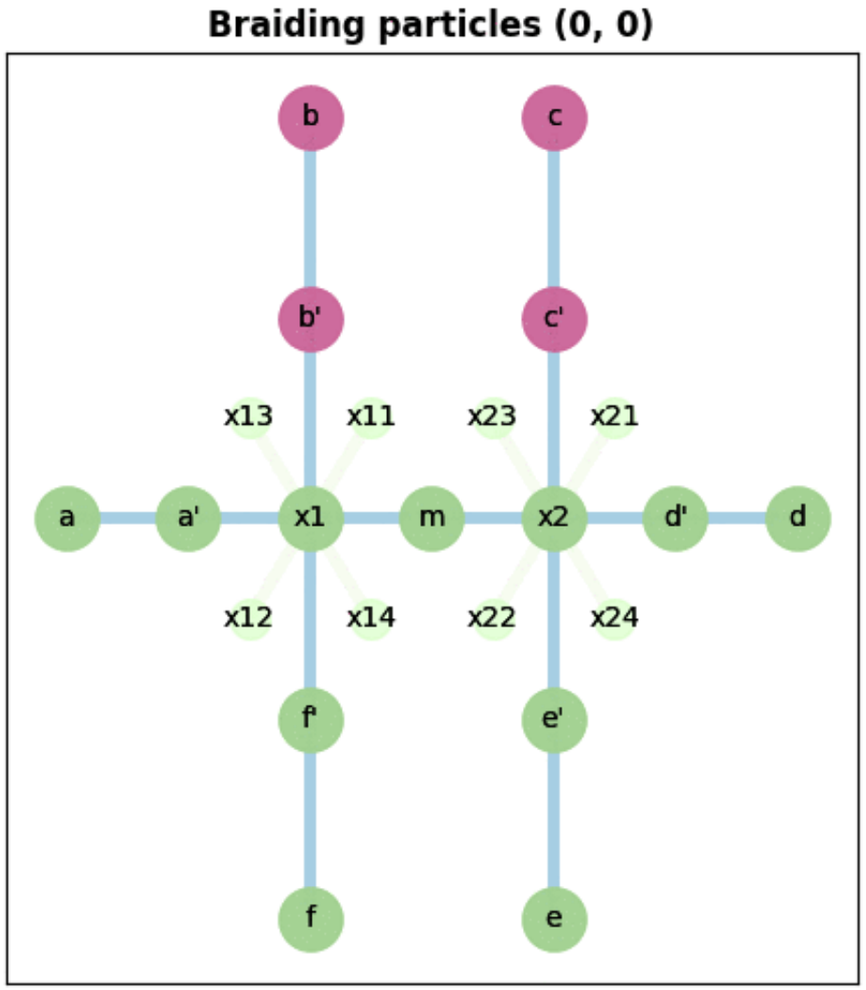
CNOT

```
##############################################################################
#                  P R E P R O C E S S I N G   P O S I T I O N S             #
##############################################################################
Positions preprocessing started...
----- Moving particles (4, 3) -----
0,0,4,b'-x1-f',0,0,0,0
0,0,3,b-b'-x1-m-x2-d'-d,0,0,0,0
0,0,4,f'-x1-m-x2-d',0,0,0,0
Positions preprocessing completed...
```

Hadamard

```
##############################################################################
#                  P R E P R O C E S S I N G   P O S I T I O N S             #
##############################################################################
Positions preprocessing started...
----- Moving particles (3, 4) -----
0,0,3,c'-x2-e',0,0,0,0
0,0,4,c-c'-x2-m-x1-f'-f,0,0,0,0
0,0,3,e'-x2-m-x1-f',0,0,0,0
Positions preprocessing completed...
```

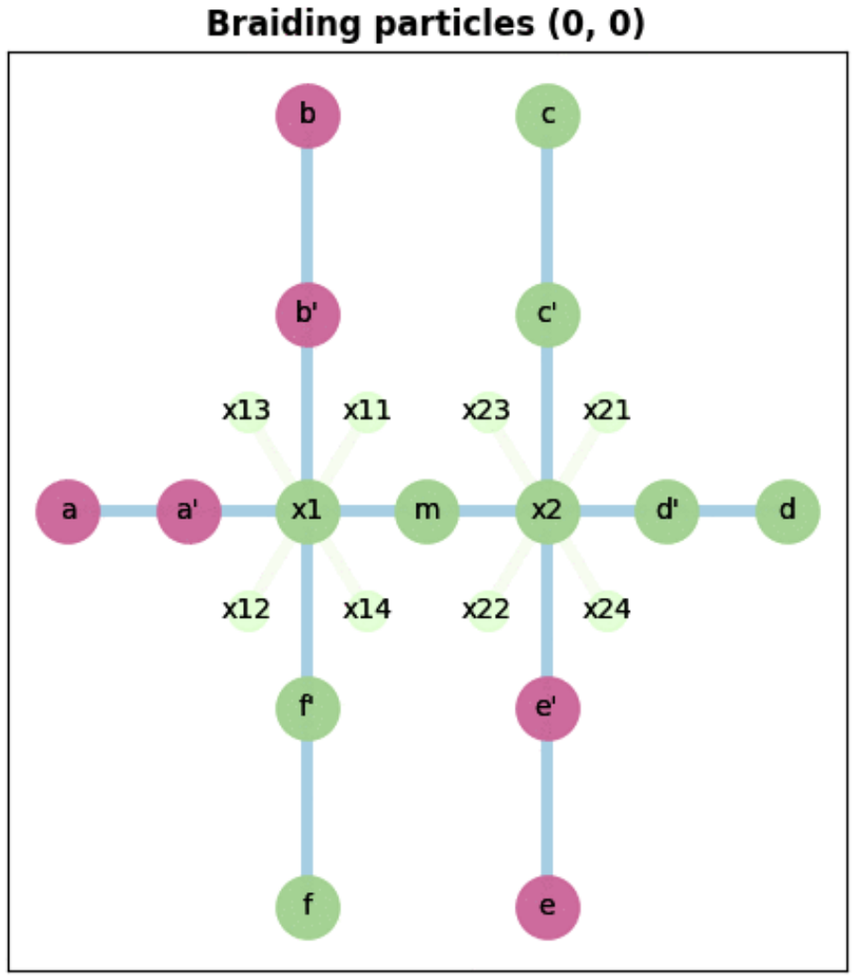
STAGE 7 – PREPROCESSING POSITIONS – HADAMARD



Braid table

Particle 1	Particle 2
0	0
3	4
2	4
2	3

STAGE 7 – PREPROCESSING POSITIONS – CNOT

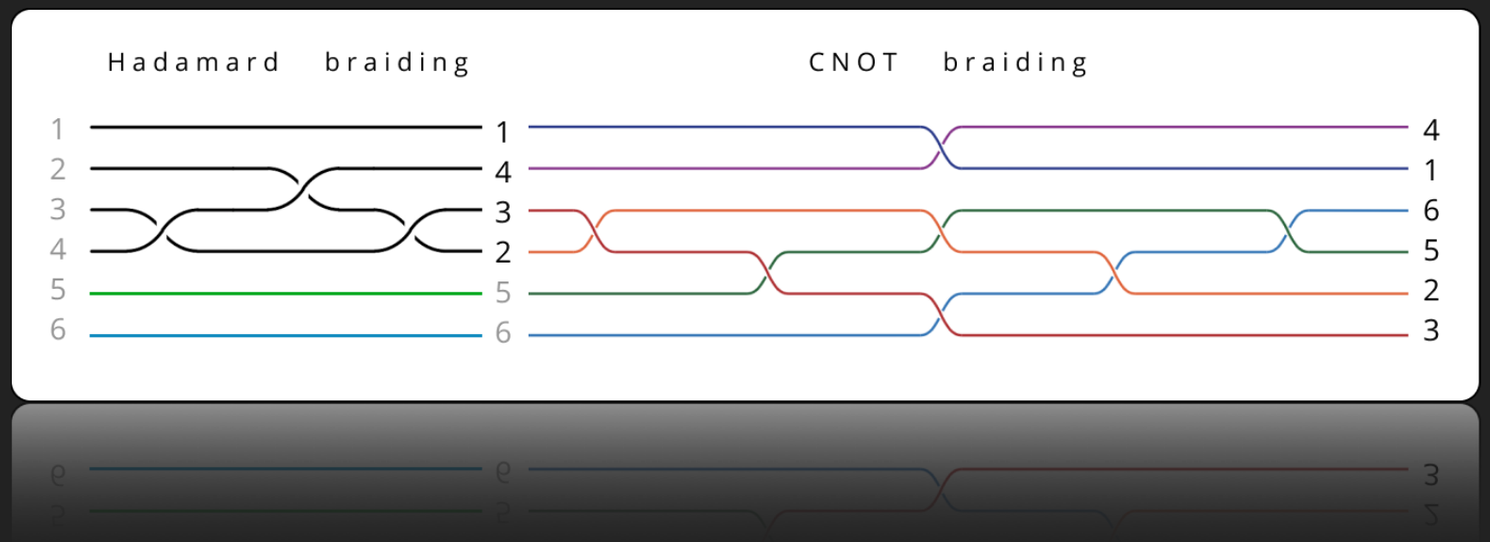


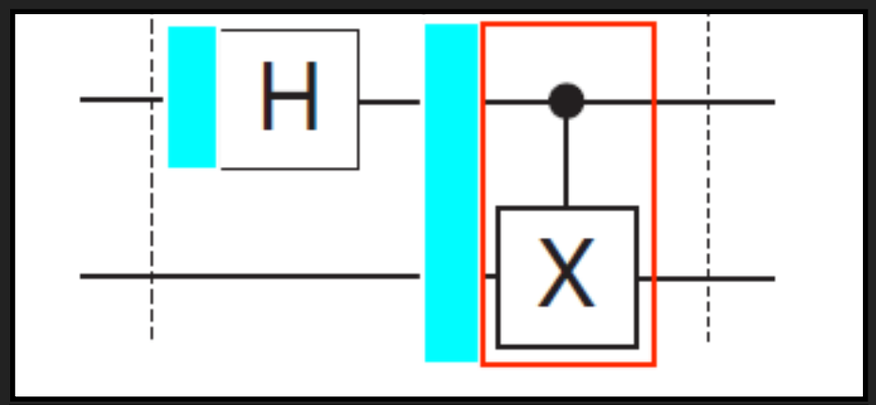
Braid table

Particle 1	Particle 2
0	0
3	4
3	5
1	2
4	5
3	6
4	6
5	6

STAGE 7 – CIRCUIT

- In a circuit, the output of one gate feeds into another.
- As a result, some of the particles (may) need to be moved so as to satisfy the valid branch config of the next gate before its execution.
- Braid sequence and positions
  - Mapping
- Entanglement circuit
  - Hadamard
    - Preprocessing
    - Braiding
  - CNOT
    - Preprocessing
    - Braiding



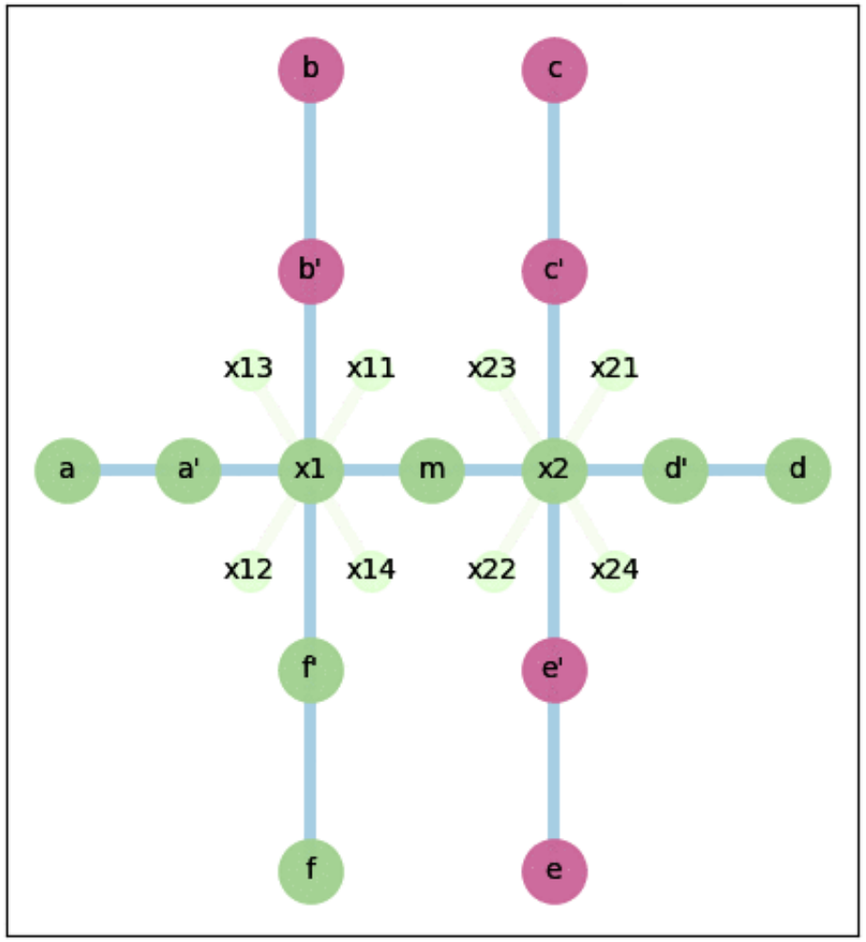


STAGE 7 - CIRCUIT

```
################################################################################################
   T O P O L O G I C A L  Q U A N T U M  C O M P U T I N G:  entanglement  C I R C U I T
################################################################################################
Topological Quantum Computing started...

Started hadamard preprocessing...
----- Moving particles (3, 4) -----
0,0,3,c'-x2-d',0,0,0,0
0,0,4,c-c'-x2-m-x1-f'-f,0,0,0,0
0,0,3,d'-x2-m-x1-f',0,0,0,0

Started hadamard braiding...
----- Braiding particles (3, 4) -----
3,4,3,f'-x1-a',0,0,0,0
3,4,4,f-f'-x1-m,0,0,0,0
3,4,3,a'-x1-f'-f,0,0,0,0
3,4,4,m-x1-f',0,0,0,0
----- Braiding particles (2, 4) -----
2,4,2,b'-x1-a',0,0,0,0
2,4,4,f'-x1-m,S,0,0,0
2,4,2,a'-x1-f',0,S,0,0
2,4,4,m-x1-b',0,S,0,0
----- Braiding particles (2, 3) -----
2,3,2,f'-x1-a',0,0,0,0
2,3,3,f-f'-x1-m,0,0,0,0
2,3,2,a'-x1-f'-f,0,0,0,0
2,3,3,m-x1-f',0,0,0,0
----- hadamard braiding completed -----

Started cnot preprocessing...
----- Moving particles (3, 4) -----
0,0,3,f'-x1-m-x2-d'-d,0,0,0,0
0,0,4,f-f'-x1-m-x2-d',0,0,0,0

Started cnot braiding...
----- Braiding particles (3, 2) -----
3,4,4,d'-x2-c',0,0,0,0
3,4,3,d-d'-x2-m,0,0,0,0
3,4,4,c'-x2-d'-d,0,0,0,0
3,4,3,m-x2-d',0,0,0,0
----- Braiding particles (3, 5) -----
3,5,3,d'-x2-c',0,0,0,0
3,5,6,e'-x2-m,0,0,0,S
3,5,5,e-e'-x2-d',0,0,S,0
3,5,3,c'-x2-e'-e,0,0,0,0
3,5,6,m-x2-e',0,0,0,0
----- Braiding particles (1, 4) -----
1,2,2,b'-x1-a',0,0,0,0
1,2,1,b-b'-x1-f',0,0,0,0
1,2,2,a'-x1-b'-b,0,0,0,0
1,2,1,f'-x1-b',0,0,0,0
----- Braiding particles (2, 5) -----
4,5,5,d'-x2-c',0,0,0,0
4,5,4,d-d'-x2-m,0,0,0,0
4,5,5,c'-x2-d'-d,0,0,0,0
4,5,4,m-x2-d',0,0,0,0
```

STAGE 7 – CIRCUIT ANIMATION NANOWIRE

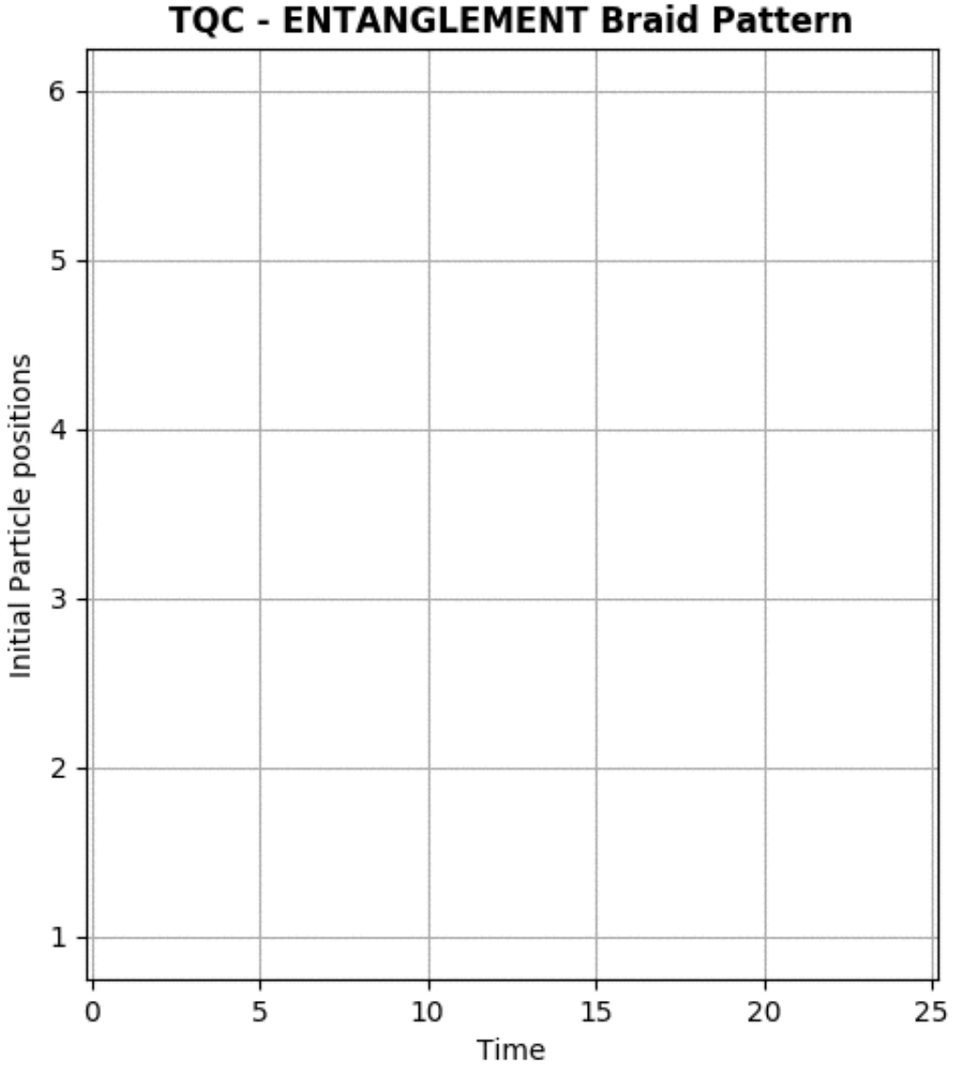
Braiding particles (0, 0)



Braid table

Particle 1	Particle 2
0	0
3	4
2	4
2	3
0	0
3	4
3	5
1	2
4	5
3	6
4	6
5	6

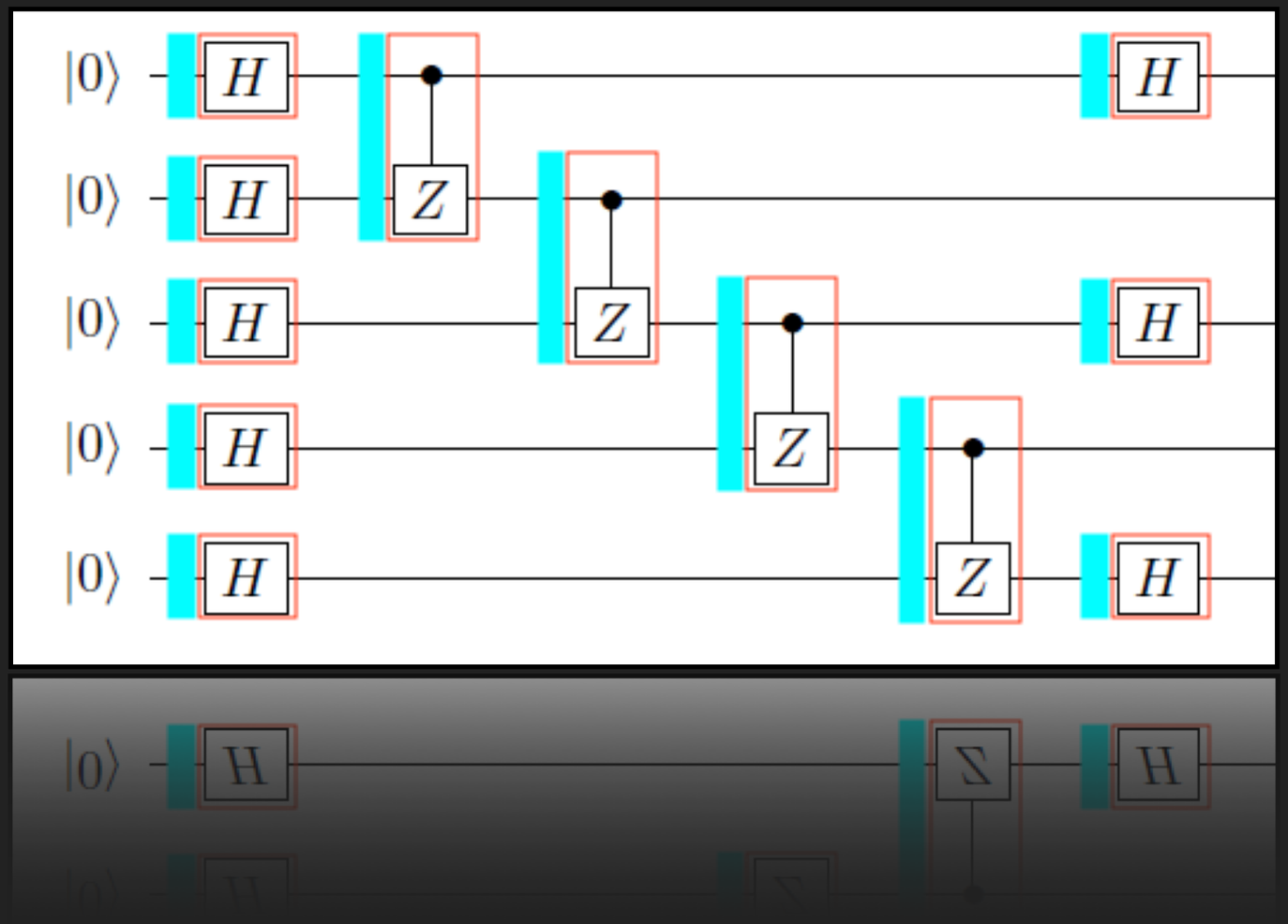
STAGE 7 – CIRCUIT ANIMATION BRAID



entanglement Braid table

Particle 1	Particle 2
3	4
2	4
2	3
0	0
3	4
3	5
1	2
4	5
3	6
4	6
5	6

STAGE 7 – CIRCUIT N-QUBITS



FUTURE WORK

- Universal gate set
  - Gate config
  - Nanowire braiding models
- N-qubit circuit
  - Gate-Qubit specification
  - Larger Nanowire structure
- Optimisation
  - Circuit (deriving gates)
  - Braid
  - Movement
- TQC Software stack
  - Nanowire-Braiding Simulator
  - TQC compiler
  - TQC Python library

FUTURE APPLICATIONS

- Quantum Computing
  - Potentially error-free Quantum gates and circuits
  - Realisation of Topological Quantum Compiler for Finance
- Cryptography
  - Braiding-based cryptography
- Knot theory
  - Jones polynomial
  - DNA recombination