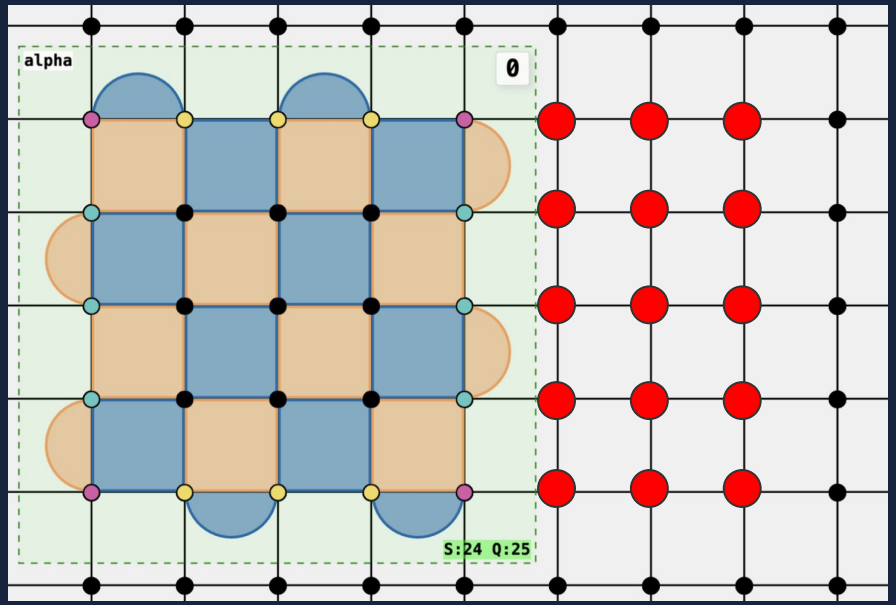
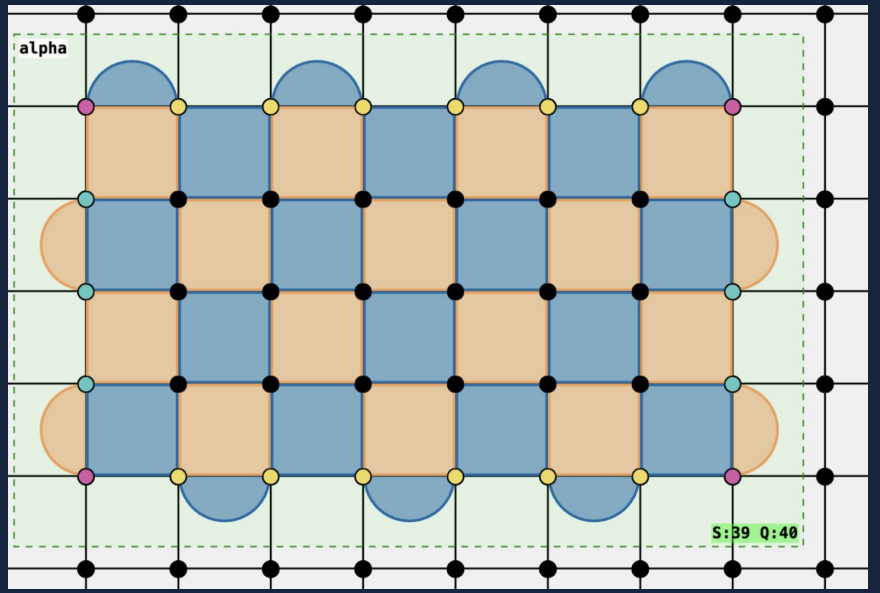
# Table of Contents

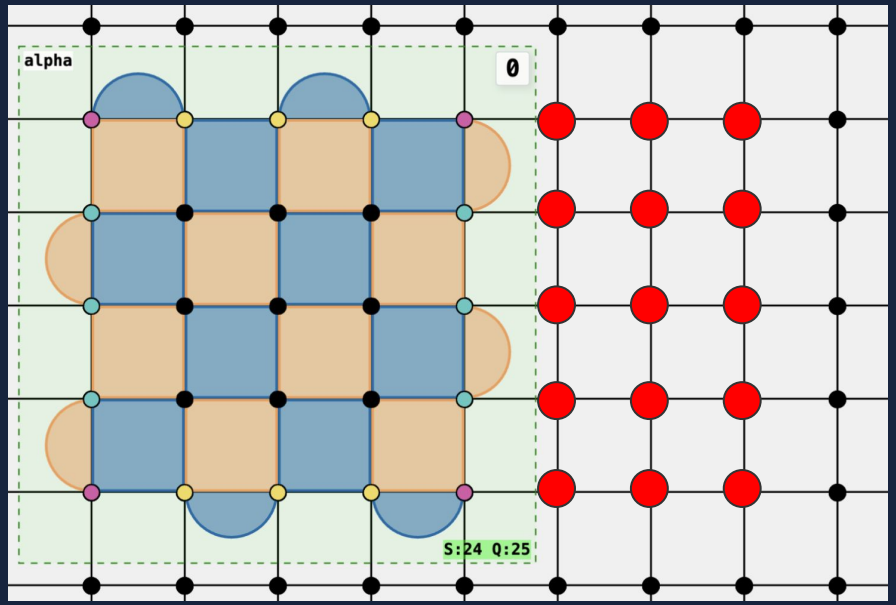
# Lattice Surgery basics

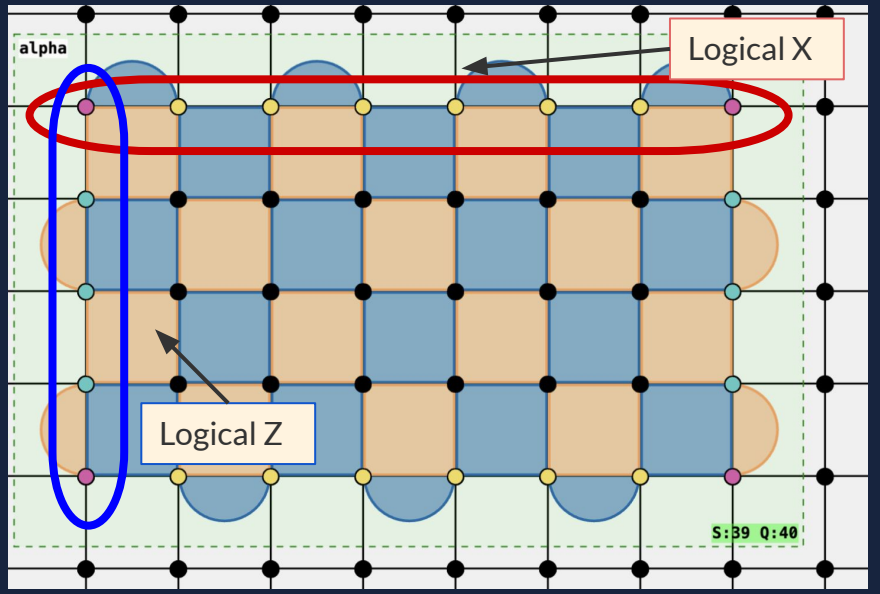



Lattice surgery operations revolve around logical qubits and “free” physical qubits surrounding the logical qubits.

# Lattice Surgery basics



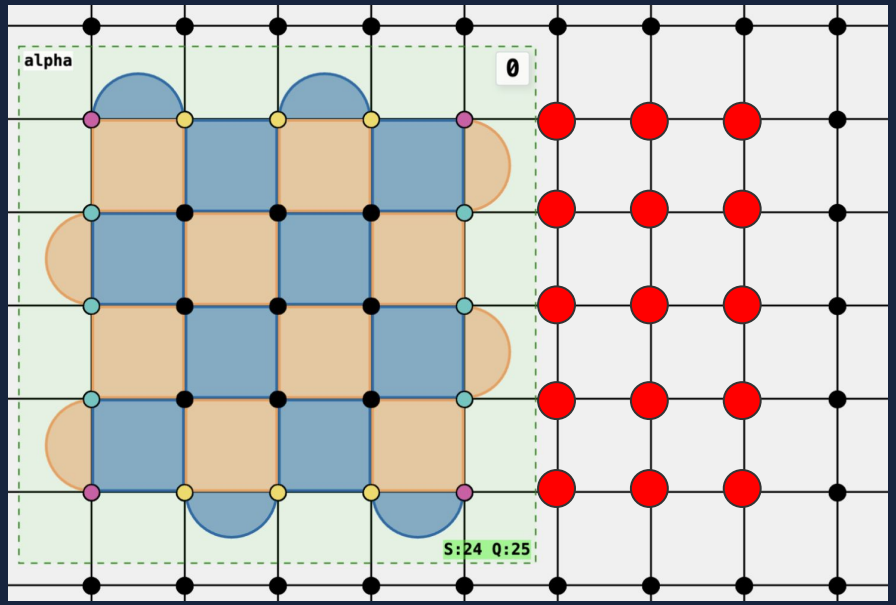




A logical qubit (of the surface code) refers a collection of physical qubits where syndrome-measuring circuits “enforce” a set of stabilizers from the surface code,

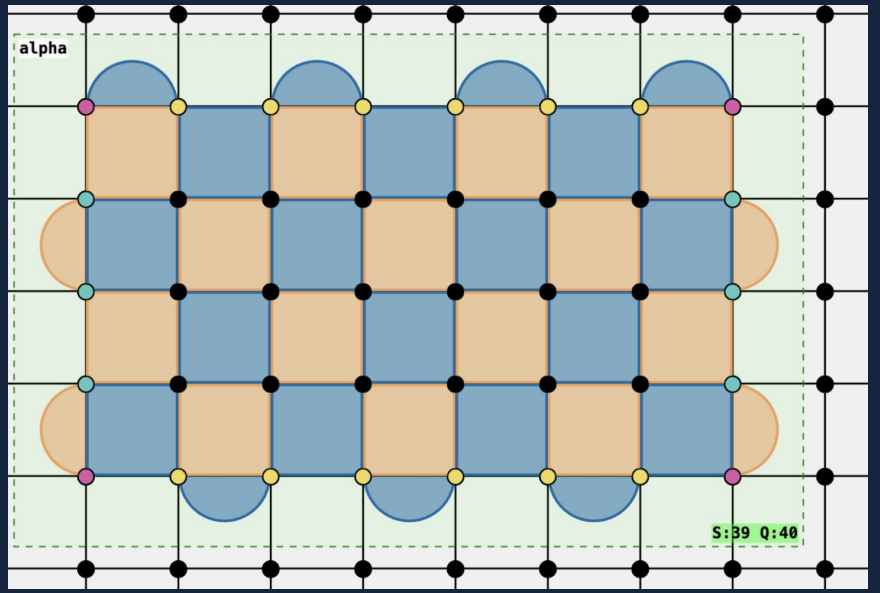
Such that there is one degree of freedom to define Pauli operators of the logical qubit.

# Lattice Surgery basics

T = t

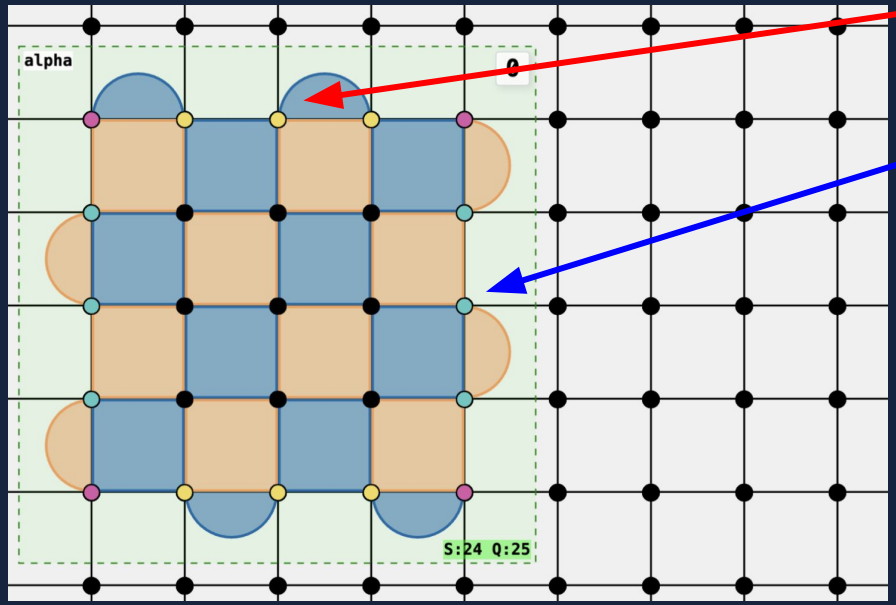



T = t+1




At any particular time step, some physical qubits are part of a logical qubits and some are not. By altering these sets based on a set of well-defined rules, we can build a sequence of such time-steps that **alter the shapes, orientation, and size of these logical qubits**.

# Lattice Surgery basics



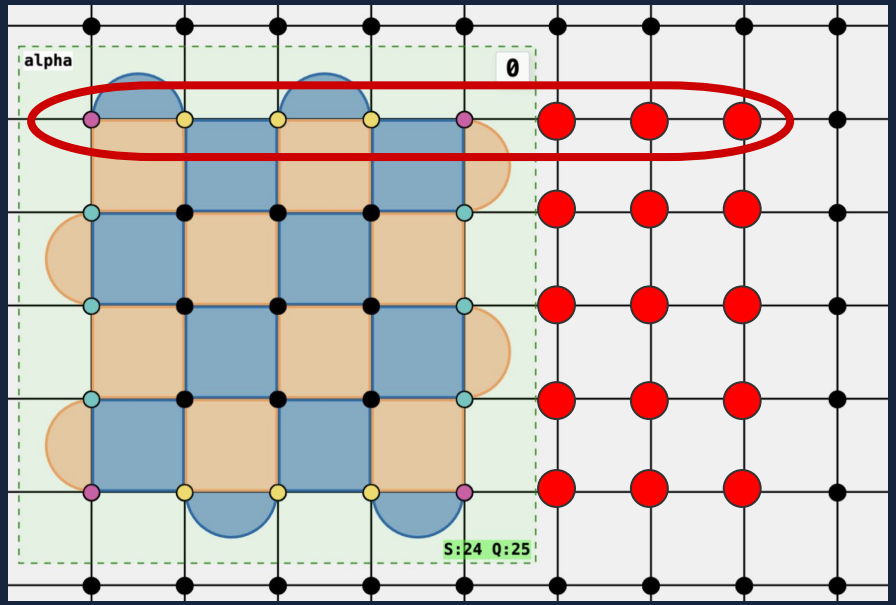

X-boundary

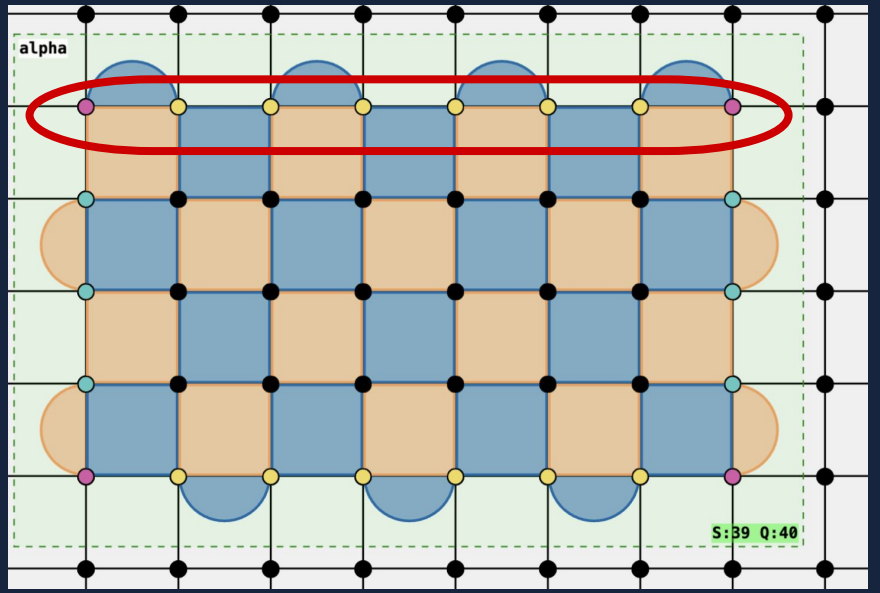
Z-boundary

Within Entwine, the boundary type is helpfully colored on the data qubits to help the user identify at a glance.

The literature is often confusing as to how the boundaries types are defined. Regardless, this image indicates our definition of boundary types.

# Grow

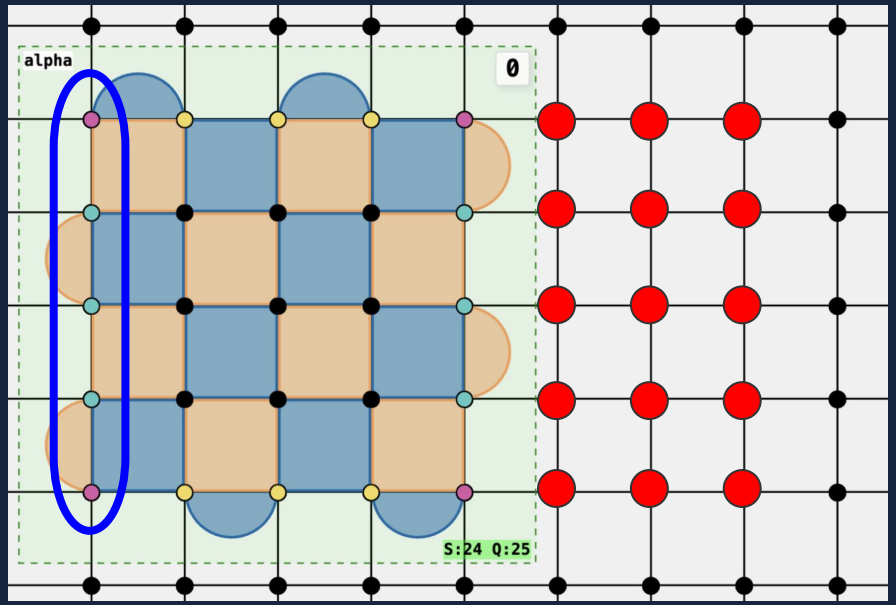


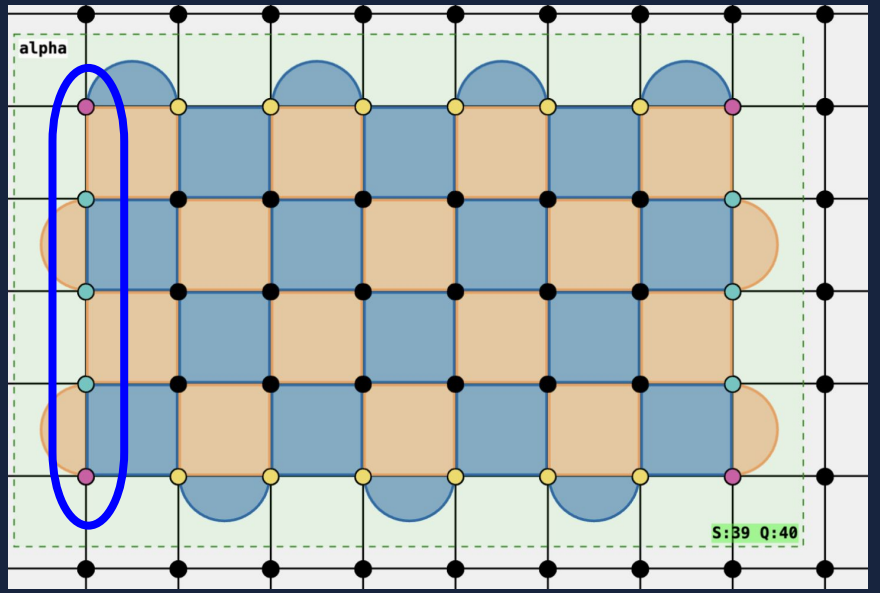




Prior to Grow (extending in the X-logical direction),
data qubits to be included into the new patch are set to +X state.

The new logical X operator (circled) is defined in terms of the old logical X operator.

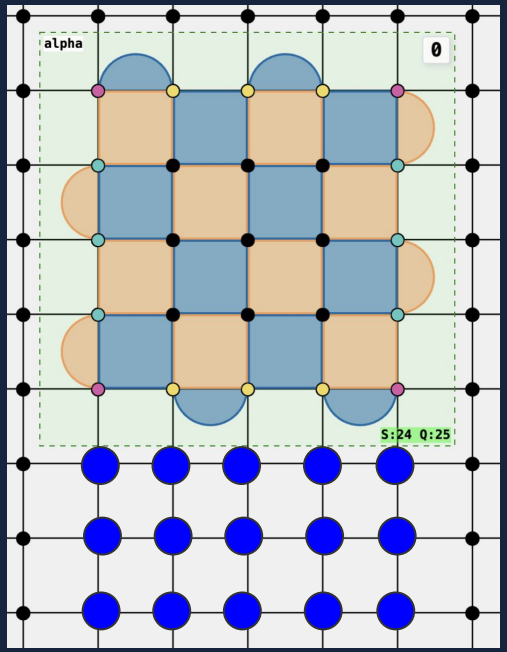
# Grow

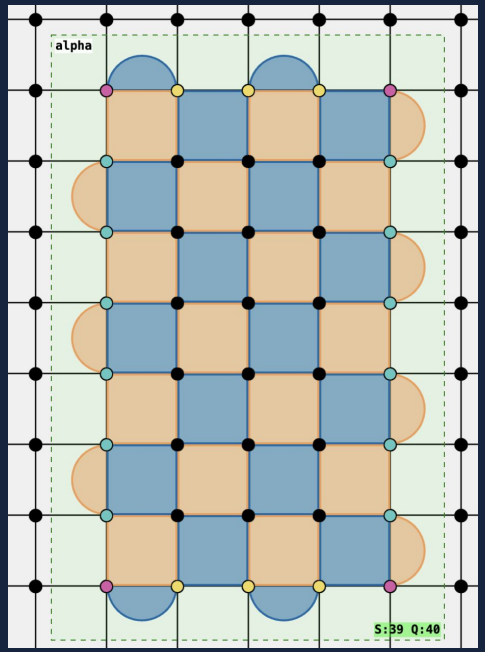


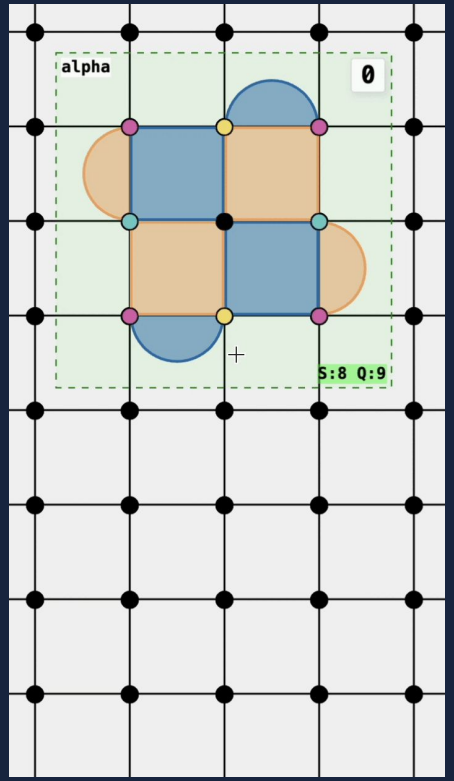




Definition of Logical Z operator is unchanged.

# Grow

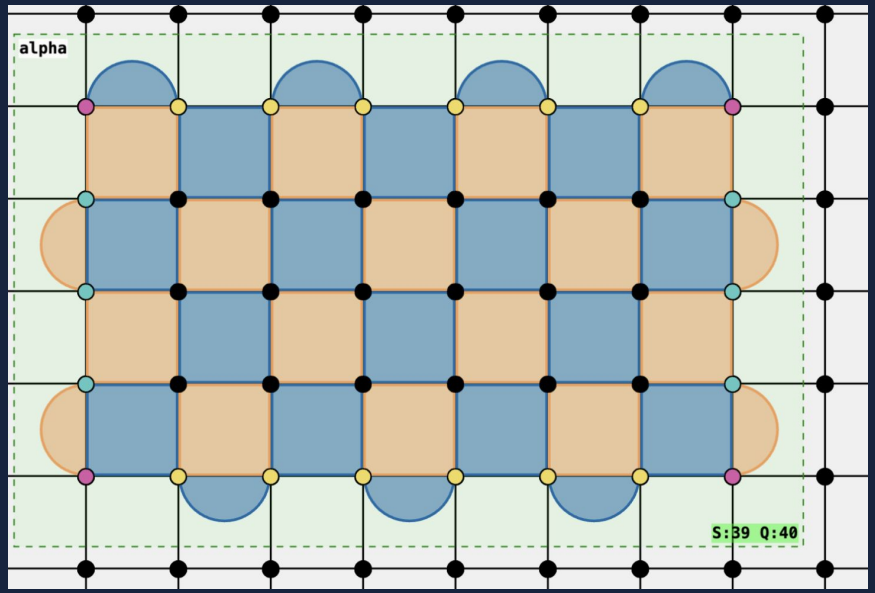


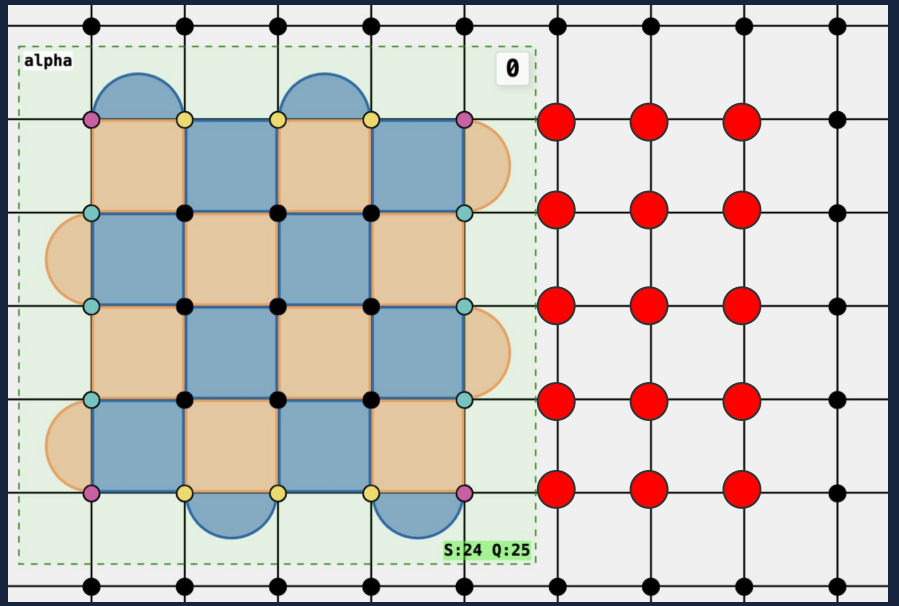




Likewise, prior to Grow (extending in the Z-logical direction),
data qubits to be included into the new patch are set to +Z state.
Logical X is unchanged, while logical Z is similarly changed as in the previous example.

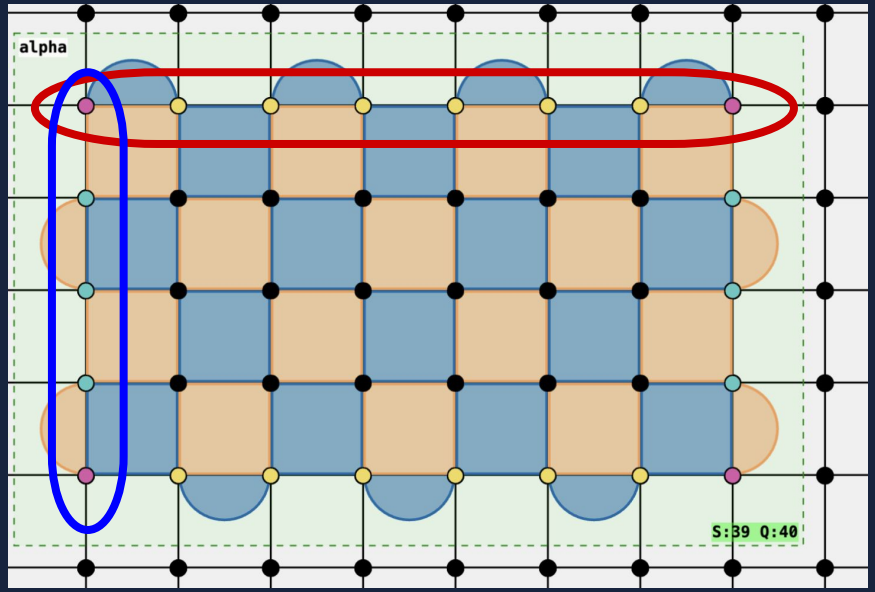
# Shrink

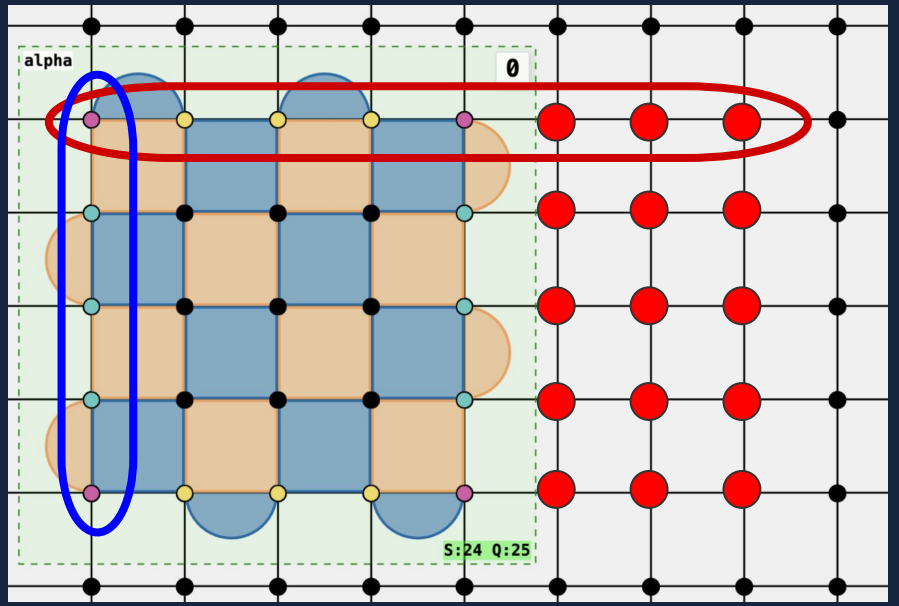



Shrink is basically the reverse of Grow.
When reducing the size of the logical qubit **along the direction of logical X**, data qubits that are no longer part of the new logical qubit are **measured in the X-basis**.

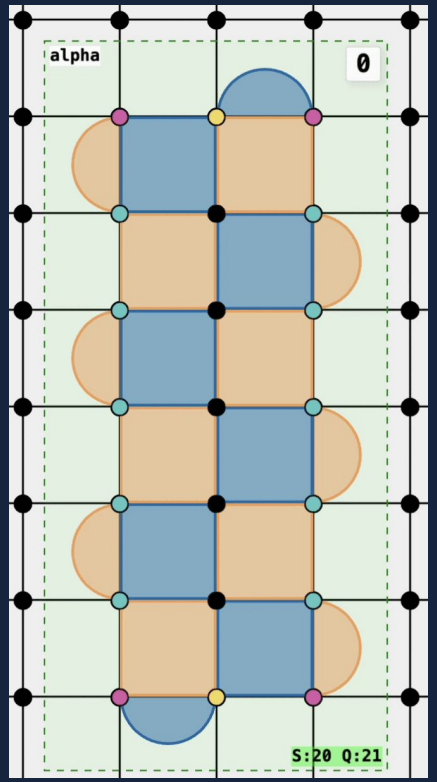
# Shrink



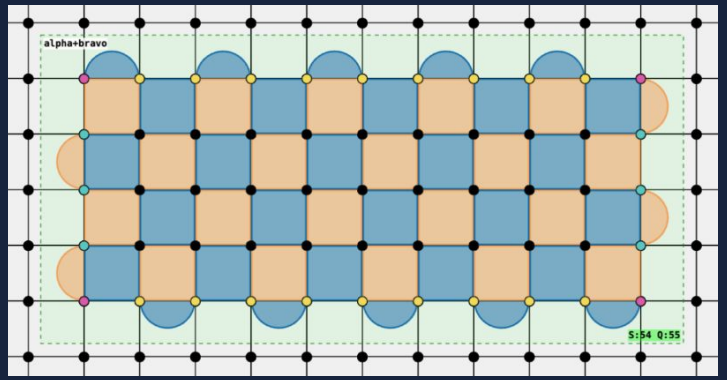

The new logical X operator is related to the old logical X operator by the product of the measurement values of the data qubits that were part of the old logical X but no longer part of the new one.

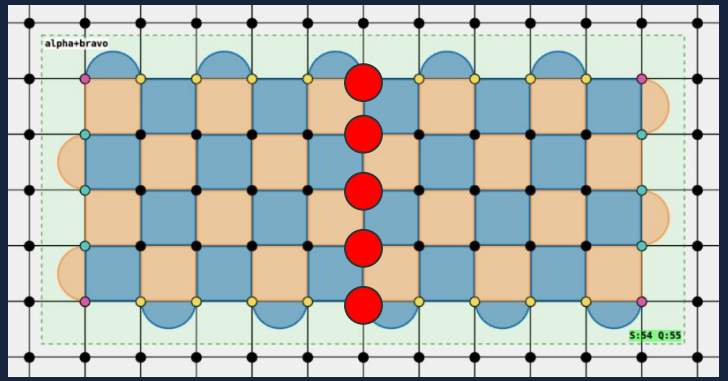
# Shrink

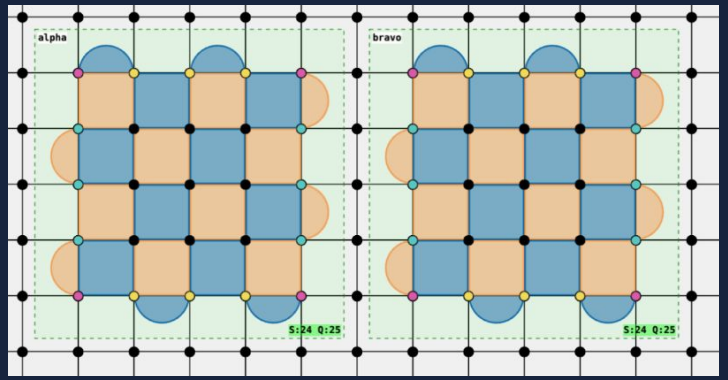



In Entwine, the process of defining the Shrink operation is similar to the Grow operation.

# Split (on the X boundary)

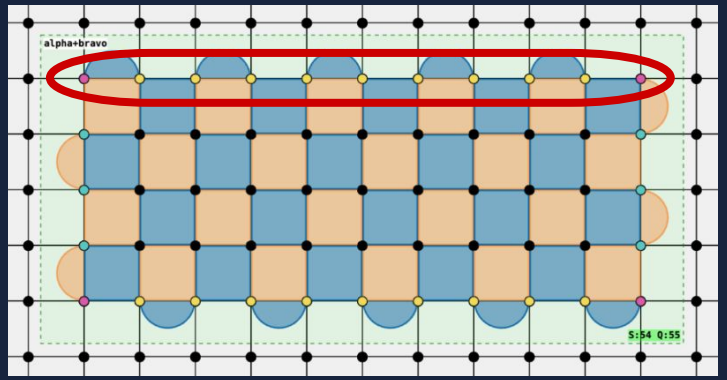


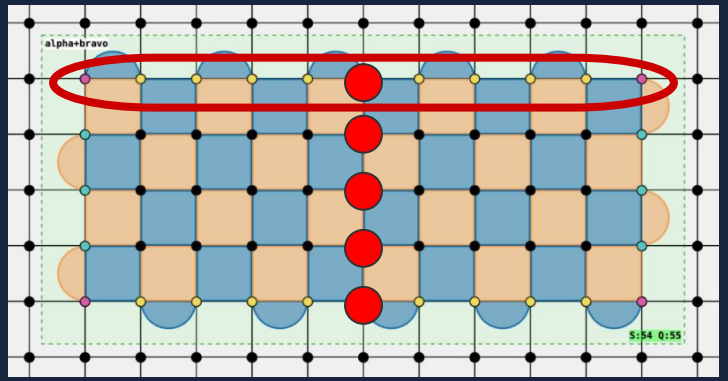



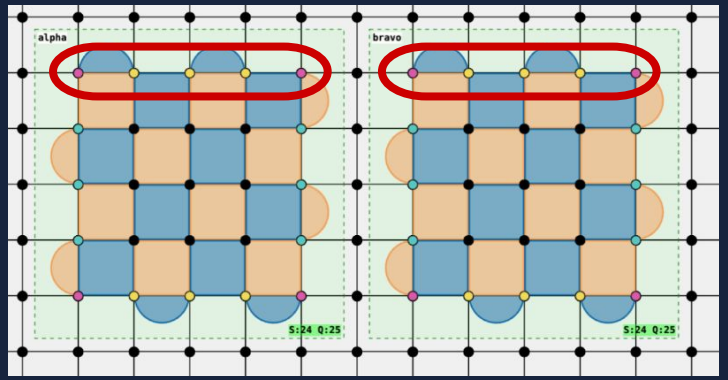




A subset of data qubits that were part of the old logical qubit is measured out in the X basis, in such a way as to result in 2 logical qubits after.

# Split (on the X boundary)

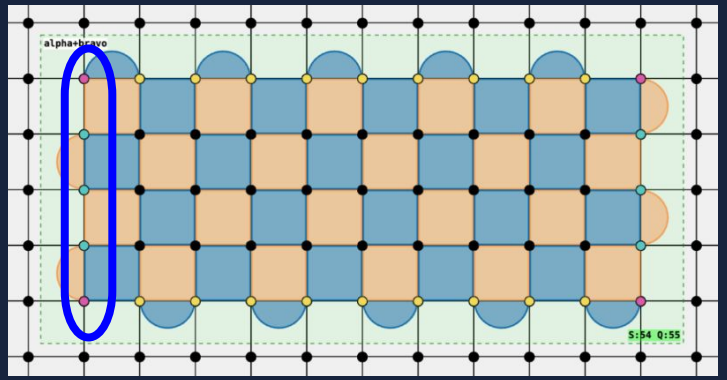


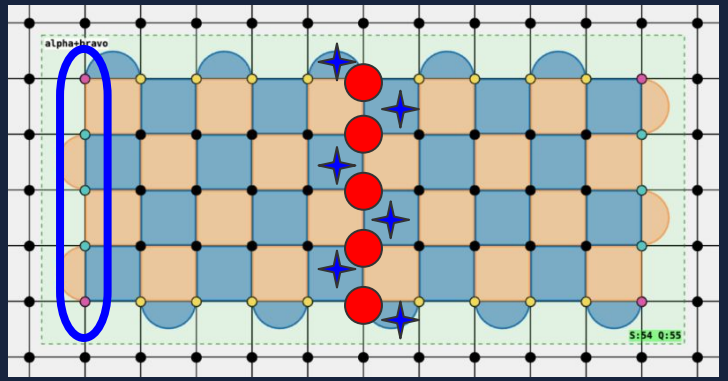



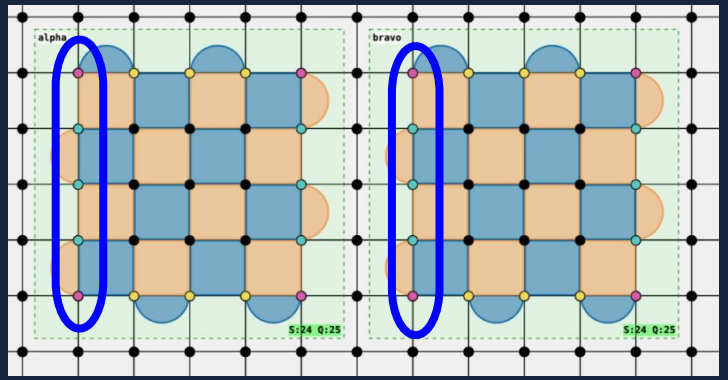




The old logical X operator is related to the new logical XX operator, with the product of one the data qubits that was measured. The eigenvalue of the the new logical XX operator is influenced by the measured value of the data qubit.

# Split (on the X boundary)

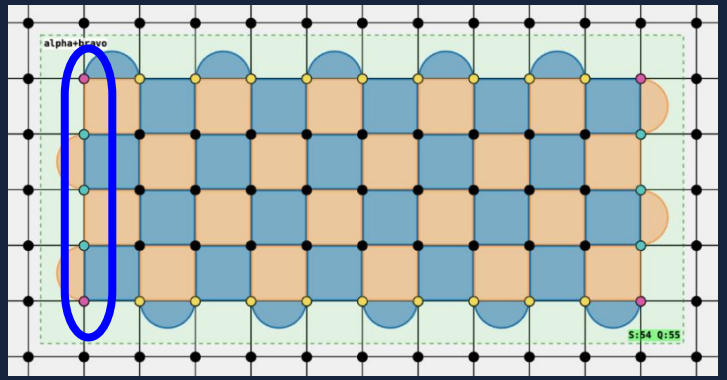


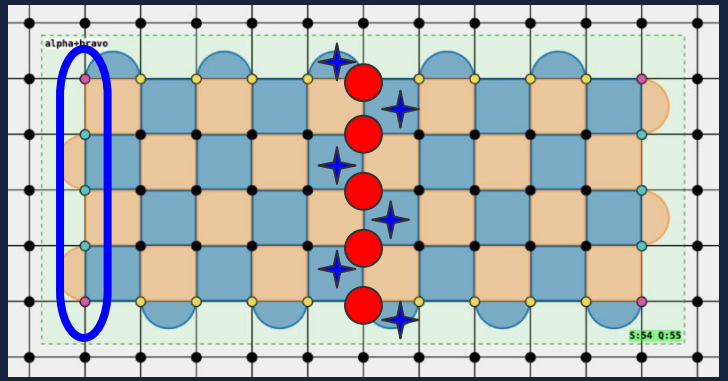



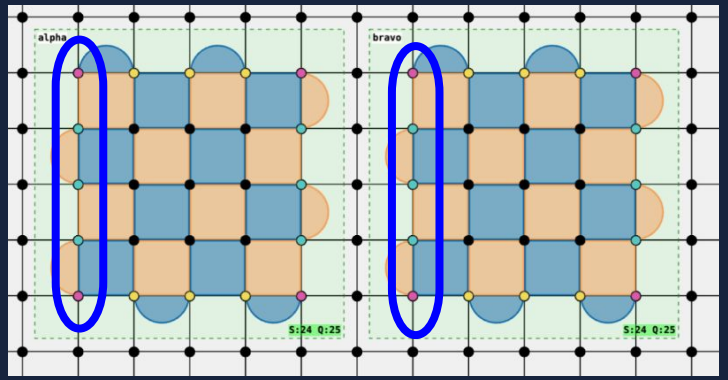




The old logical Z operator is redefined as logical Z operator of one the logical qubits. The logical Z operator of the other logical qubit is defined as the product of the logical Z operator of the first logical qubit and the measured values of the Z stabilizers (marked in figure).

# Split (on the X boundary)

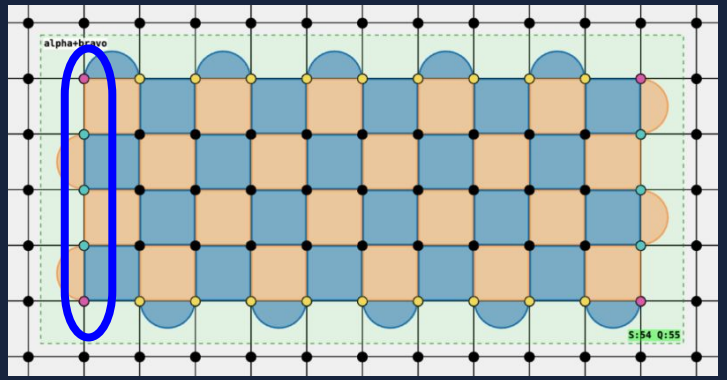


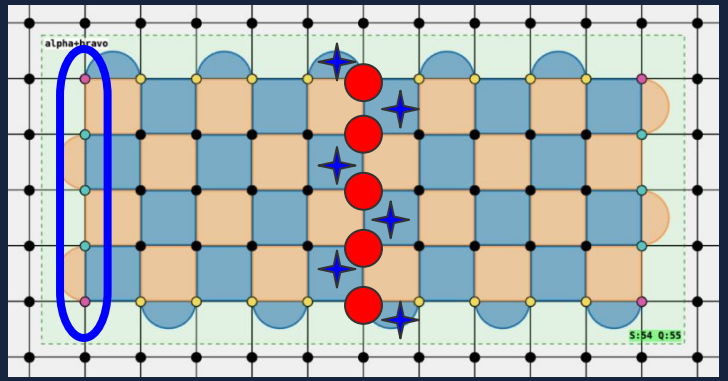



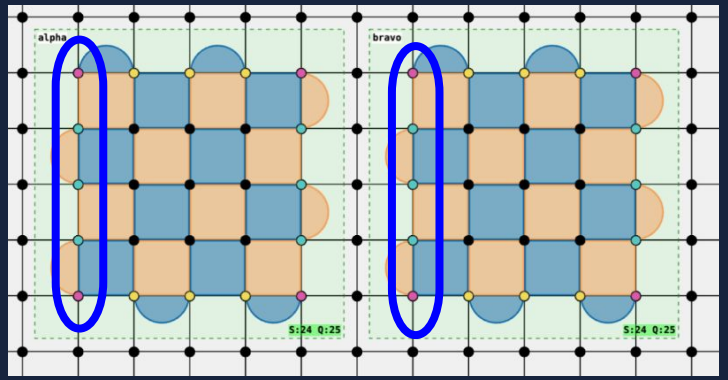




Take note that regardless of the logical state encoded by the old logical qubit, the logical state of the new logical qubits post-Split will be stabilized by logical ZZ, given by the product of the marked Z-stabilizers.

# Split (on the X boundary)

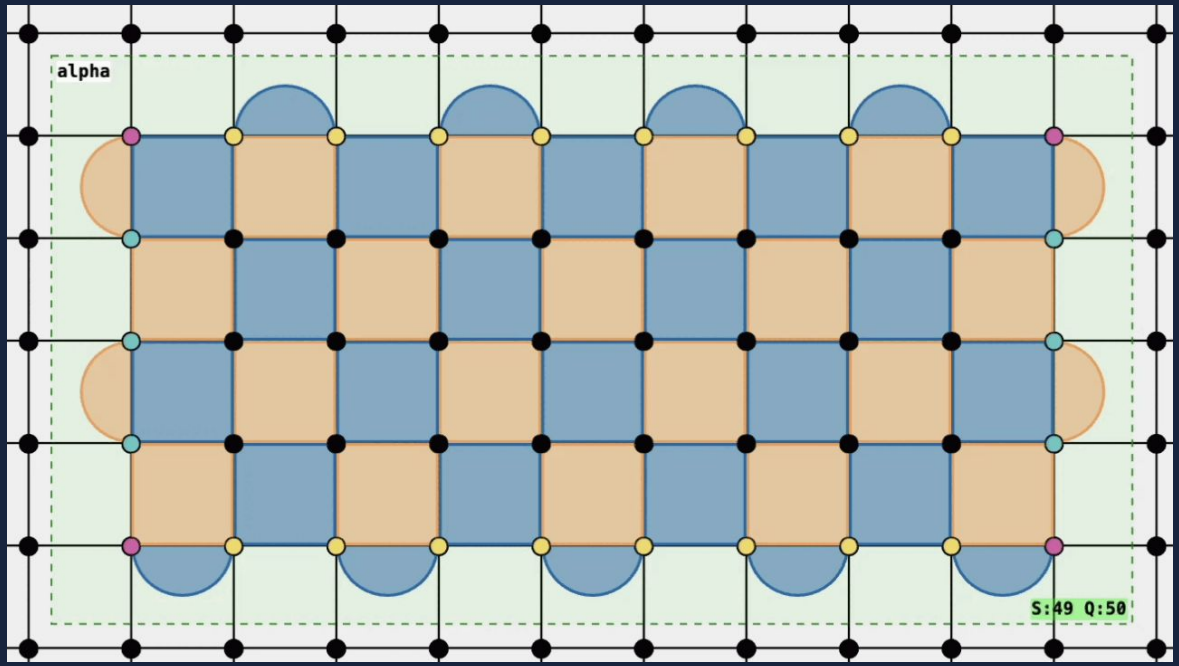


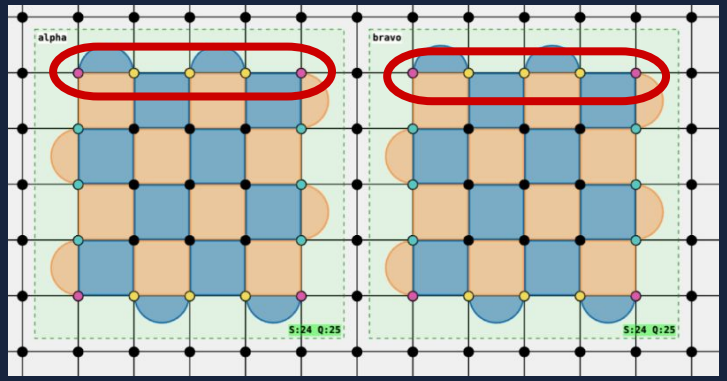

Because the eigenvalue of the logical ZZ operator is known (determined), the Split operation is also a part of the logical Measurement operation of Pauli ZZ.

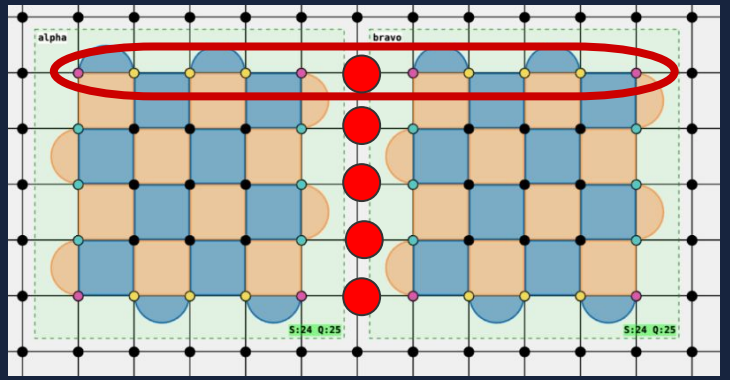
# Split (on the X boundary)

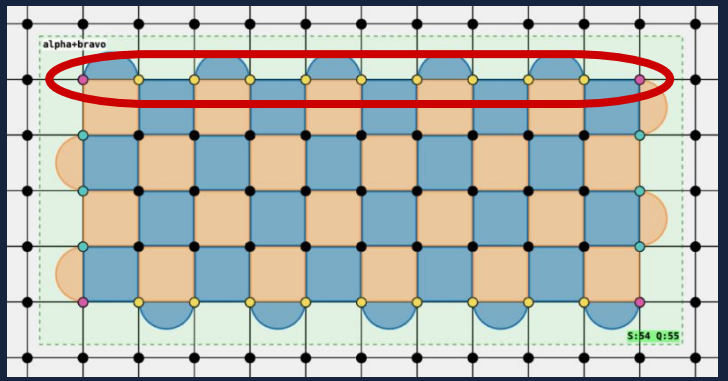



Simply select a set of data qubits in Entwine, to create the split one logical qubit into two.

# Merge (on the X boundary)

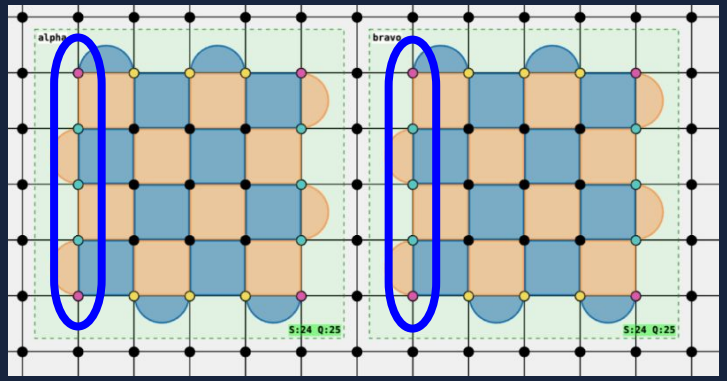


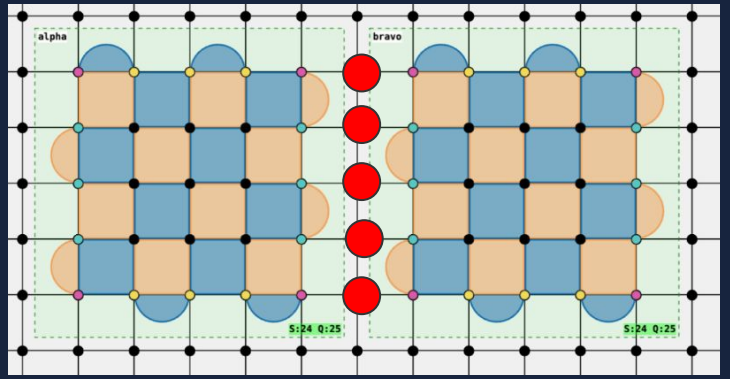



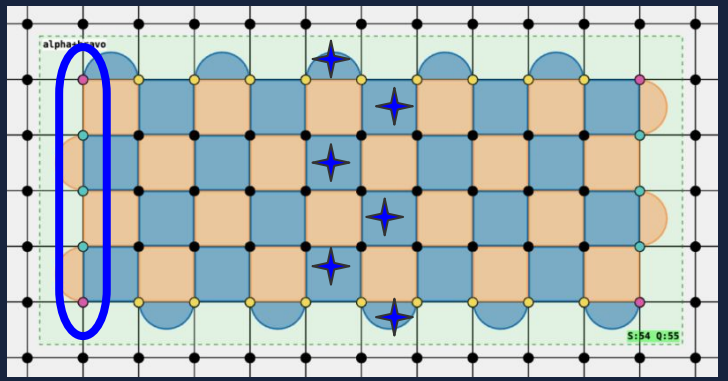




The Merge on the X boundary lines up the 2 separate logical X operators of their respective logical qubits, to create a single new logical X operator.

# Merge (on the X boundary)

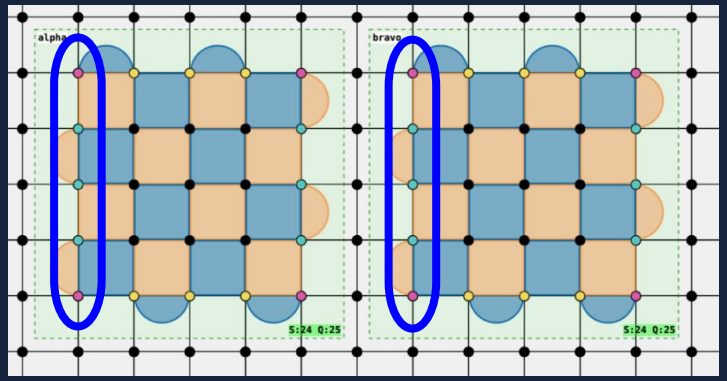


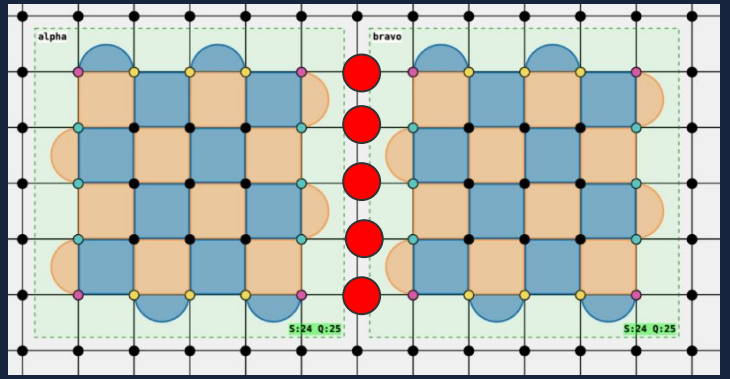



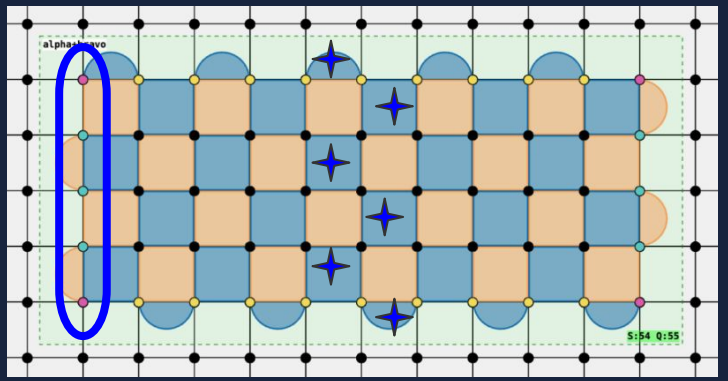




Just like the previous Split operation, the new logical qubit post-Merge is stabilized by the product of the marked Z-stabilizers, equivalent to the logical ZZ operator of the old logical qubits.

# Merge (on the X boundary)

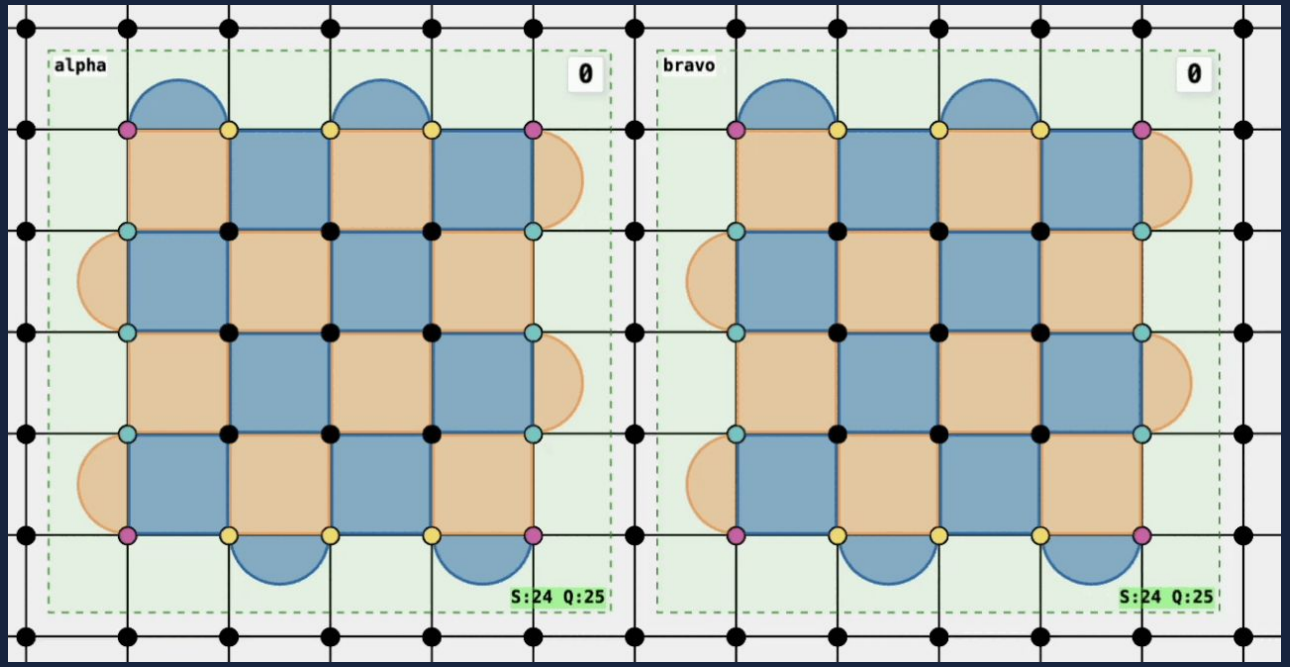



Just like the previous Split operation, the new logical qubit post-Merge is stabilized by the product of the marked Z-stabilizers, equivalent to the logical ZZ operator of the old logical qubits.

# Merge (on the X boundary)




Selecting a set of data qubits in Entwine that includes boundaries of 2 adjacent patches creates the Merge operation.

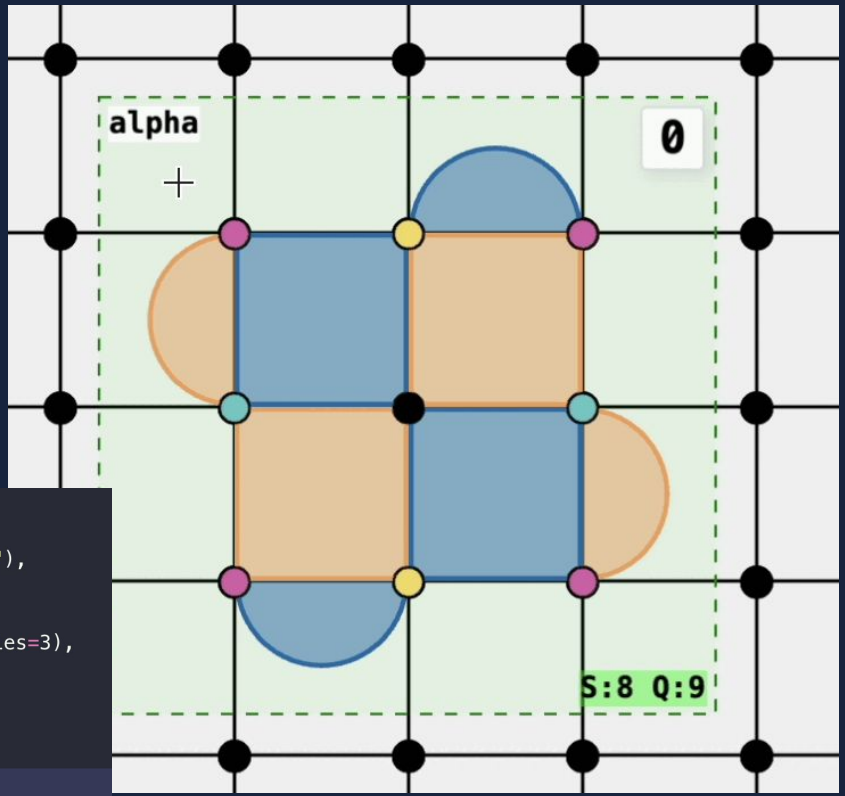
# Mid-Circuit Reset

In Entwine, it is possible to re-initialize a particular logical qubit after a destructive logical measurement is carried out.

Simply drag-and-select the data qubits that were part of the original logical qubit to bring it back.

In terms of lattice surgery sequence, we have the following:

```
operations = (
    (
        ResetAllDataQubits("alpha", state="0"),
    ),
    (
        MeasurePatchSyndromes("alpha", n_cycles=3),
    ),
    ( # Frame 1
        MeasureLogicalZ("alpha"),
    ),
    ( # Frame 2
        ResetAllDataQubits("alpha", state="0"),
    ),
    (
        MeasurePatchSyndromes("alpha", n_cycles=3),
    ),
)
```



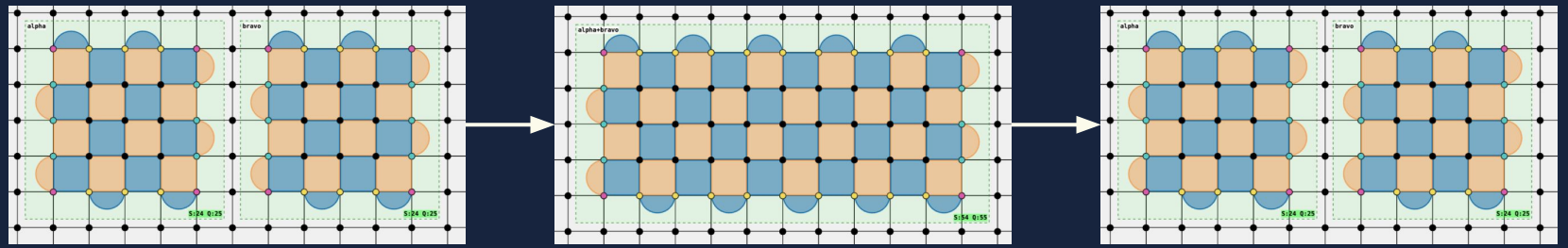
# Simple Lattice surgery Combinations

# Merge + Split → multi-qubit pauli measurement

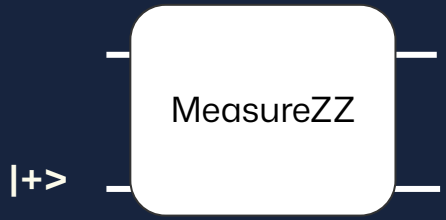

A two-qubit Pauli (non-destructive) measurement takes two qubits as input and outputs two qubits as well as a classical bit for the eigenvalue of the Pauli ZZ operator.

This can be achieved by the Merge + Split combination, on the X boundary.

It is not hard to see that a similar Merge + Split combination, on the Z boundary will yield the MeasureXX logical operation.

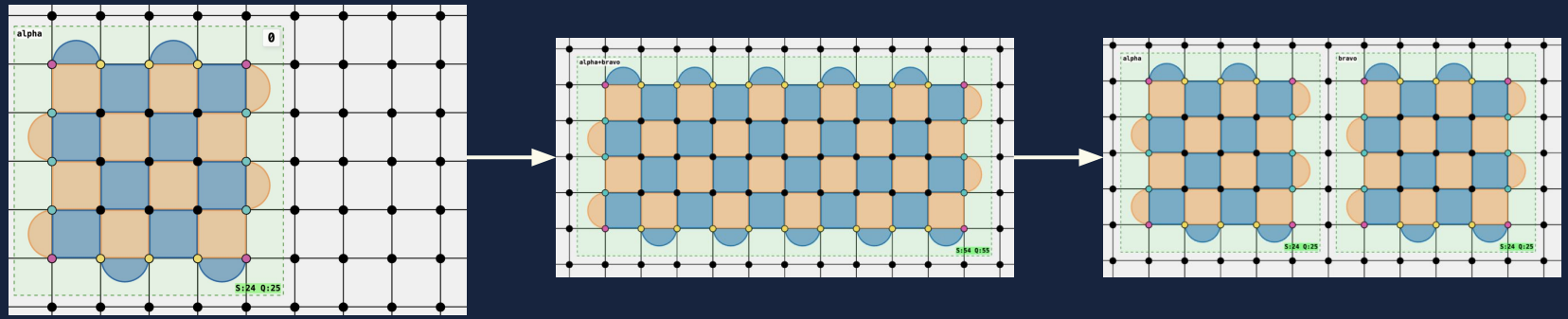


# Grow + Split

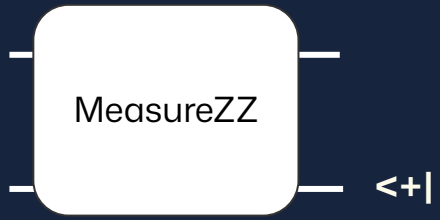



Suppose, somewhere in your logical circuit, you need to bring in an ancilla qubit in the |+> state, and then do the MeasureZZ operation afterwards.

You might be tempted to just initialize a second logical qubit, and do a Merge + Split combination. However, you could also simply just Grow the first logical qubit, and then do a Split.

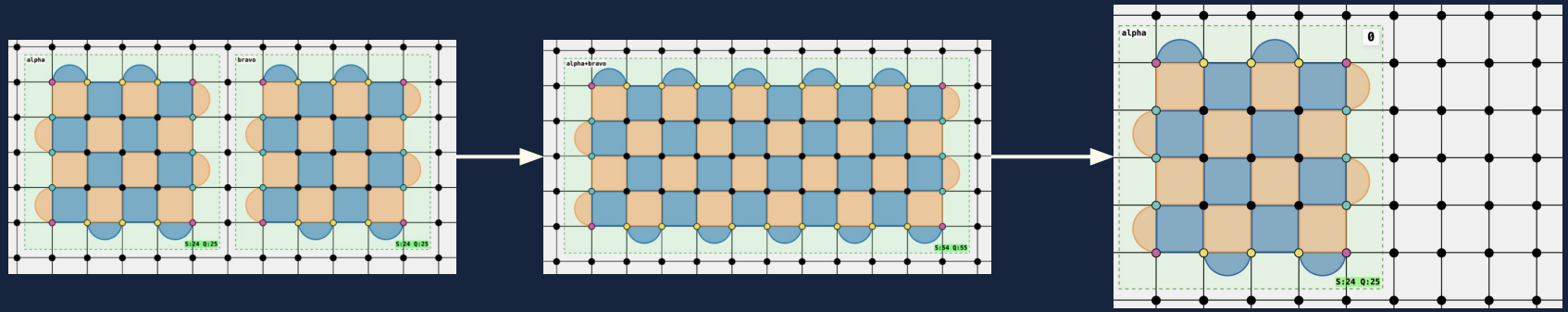


# Merge + Shrink



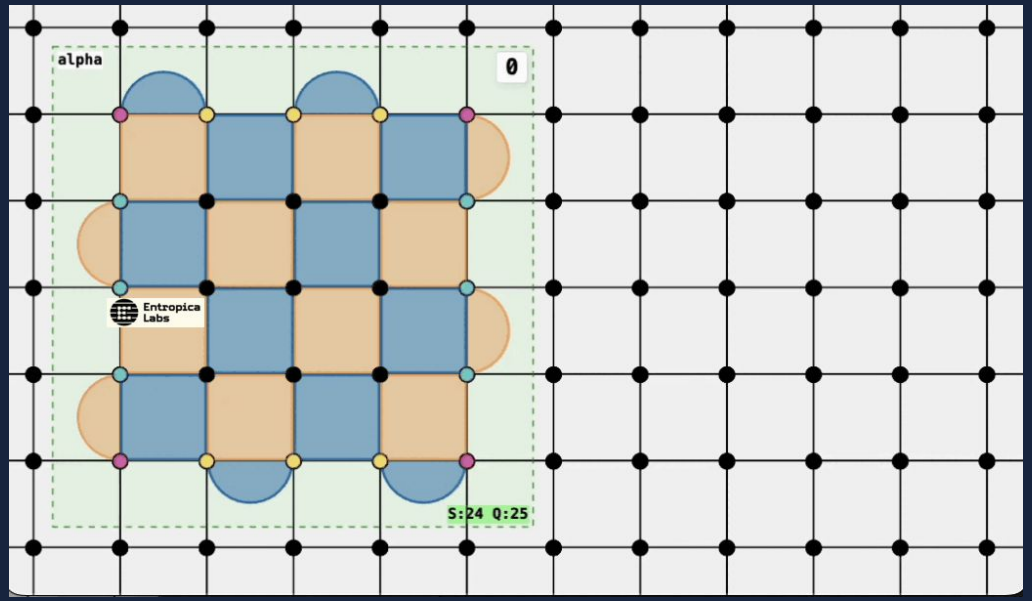

Likewise, sometimes you may need to measure out one of the logical qubit straight-away after a MeasureZZ operation.

You can easily do so with a Merge + Shrink combination.



# Grow + Shrink

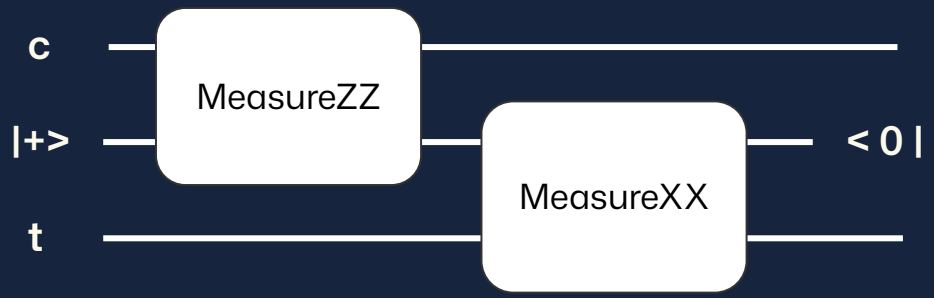
Logical qubit encoded with surface code has a fixed spatial location. Sometimes you may need to shift its location.




This can be done with a Grow and Shrink combination.

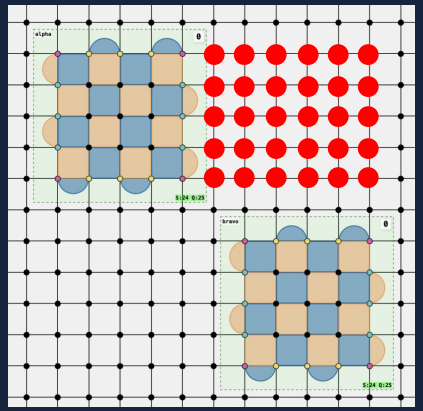
Try to pay attention to how the logical Pauli operators change during this combination operation.

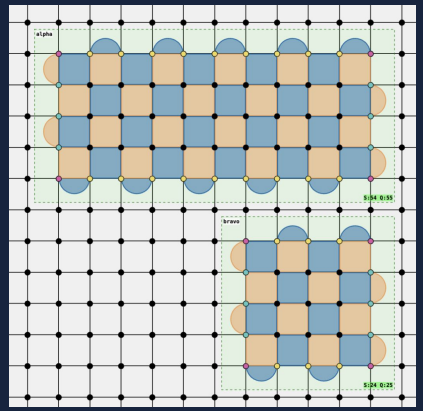
# Logical CNOT

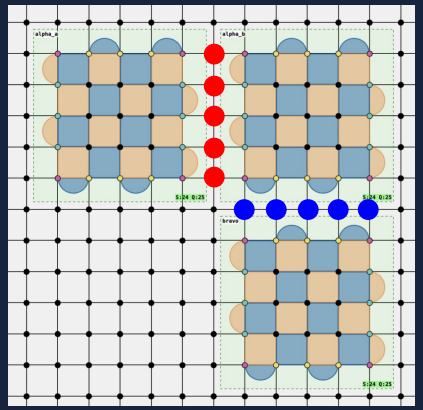


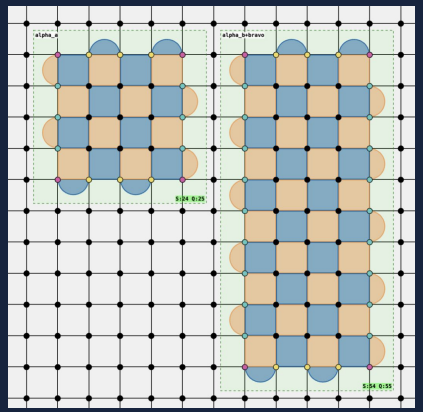

The logical CNOT can be implemented through measurement of Pauli operators, as depicted in the above figure, up to single-qubit Pauli corrections on both control and target qubits.

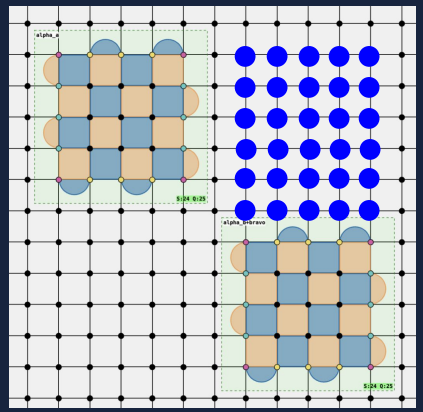
# Logical CNOT











The logical CNOT can thus be implemented with a sequence of lattice surgery combinations, Grow+Split → Merge+Shrink

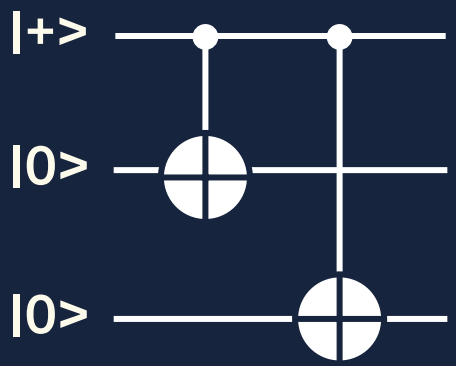
# Tutorial example

State preparation for logical GHZ state

# Logical State Preparation: GHZ

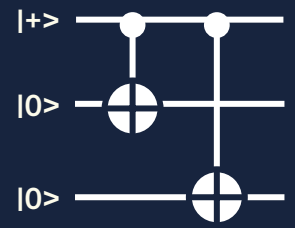
This is the GHZ state: $\frac{|000>+|111>}{\sqrt{2}}$

And we might prepare the state using the following circuit.

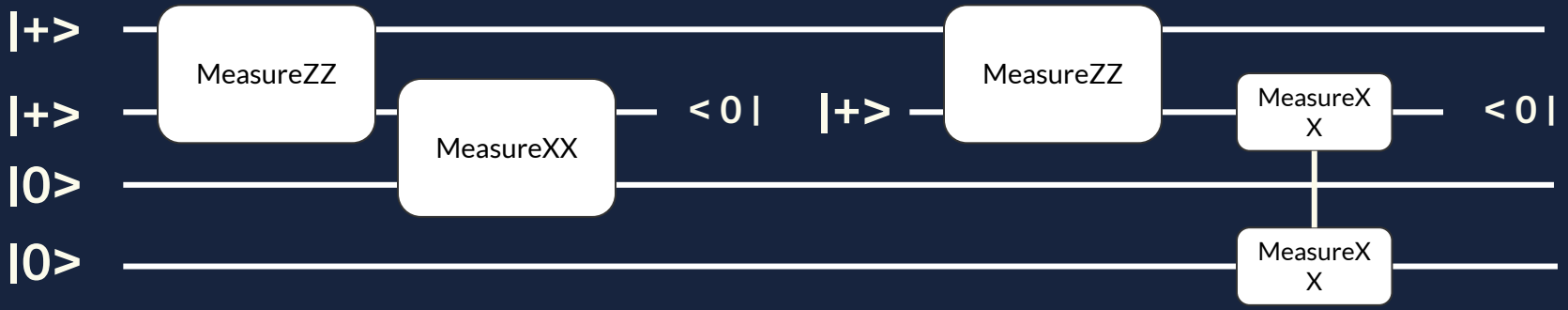



To implement this circuit with surface code logical qubits, we need to consider how to implement the logical operations such as CNOT with lattice surgery.

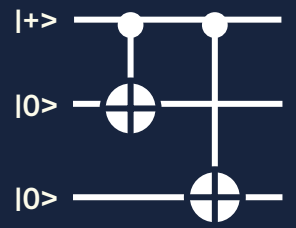
# Logical State Preparation: GHZ



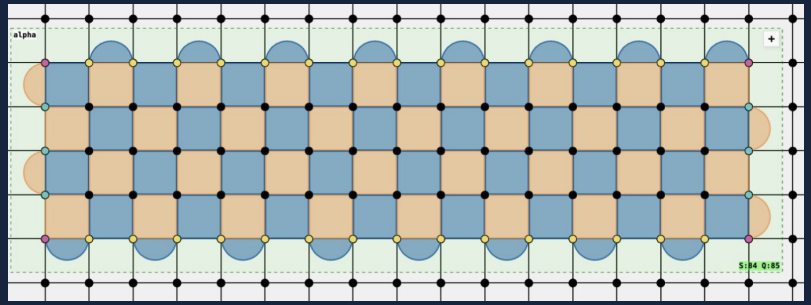

One way to do this is to convert the CNOT gates into a sequence of lattice surgery operations. Notice how we have to include one auxiliary logical qubit.

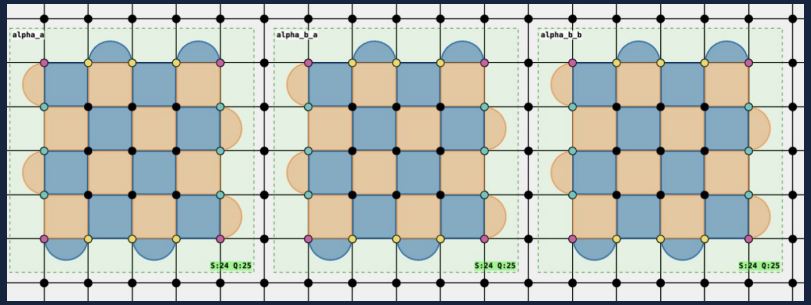


# Logical State Preparation: GHZ




The state preparation circuit can be optimized, removing the need for the auxiliary logical qubit, by cleverly dealing directly with what lattice surgery can do, instead of thinking of the higher abstraction of CNOTs.




Start by preparing the single “long” patch, initialized in the +X logical state. Then perform the Split operation, breaking up the existing logical X operator into 3 individual ones.

One can verify that the resulting output state is indeed stabilized by the logical Pauli operators: {XXX , ZZI, IZZ }

# Bonus: logical CZ gate