IQI 04, Seminar 7

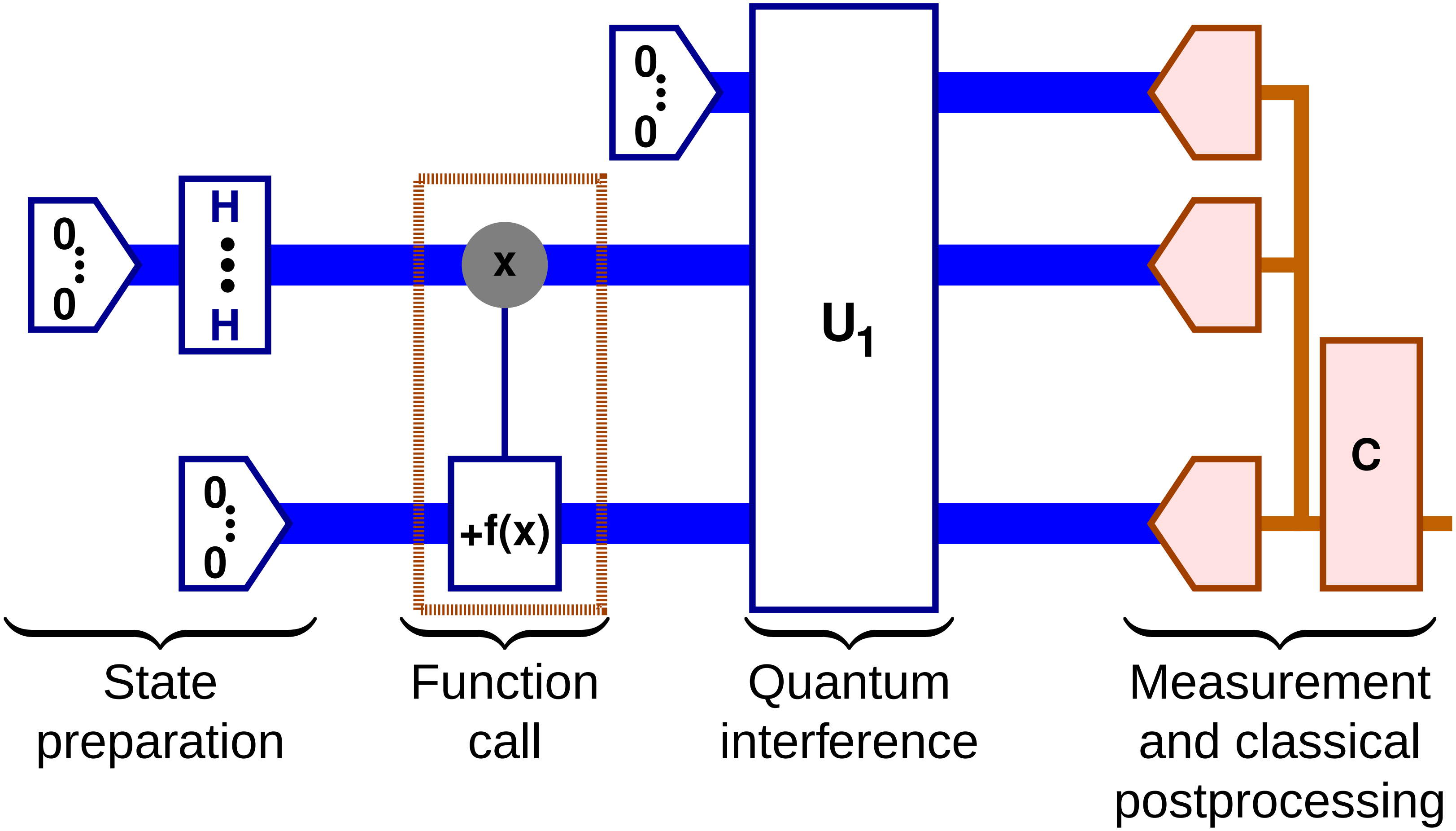
Produced with pdflatex and xfig

- Reversible classical computation.
- From irreversible to reversible computation.

E. “Manny” Knill: `knill@boulder.nist.gov`

Recognizing Patterns in Functions

- Algorithm structure for determining some **Property**(f).
  - One function call:



State preparation

Function call

Quantum interference

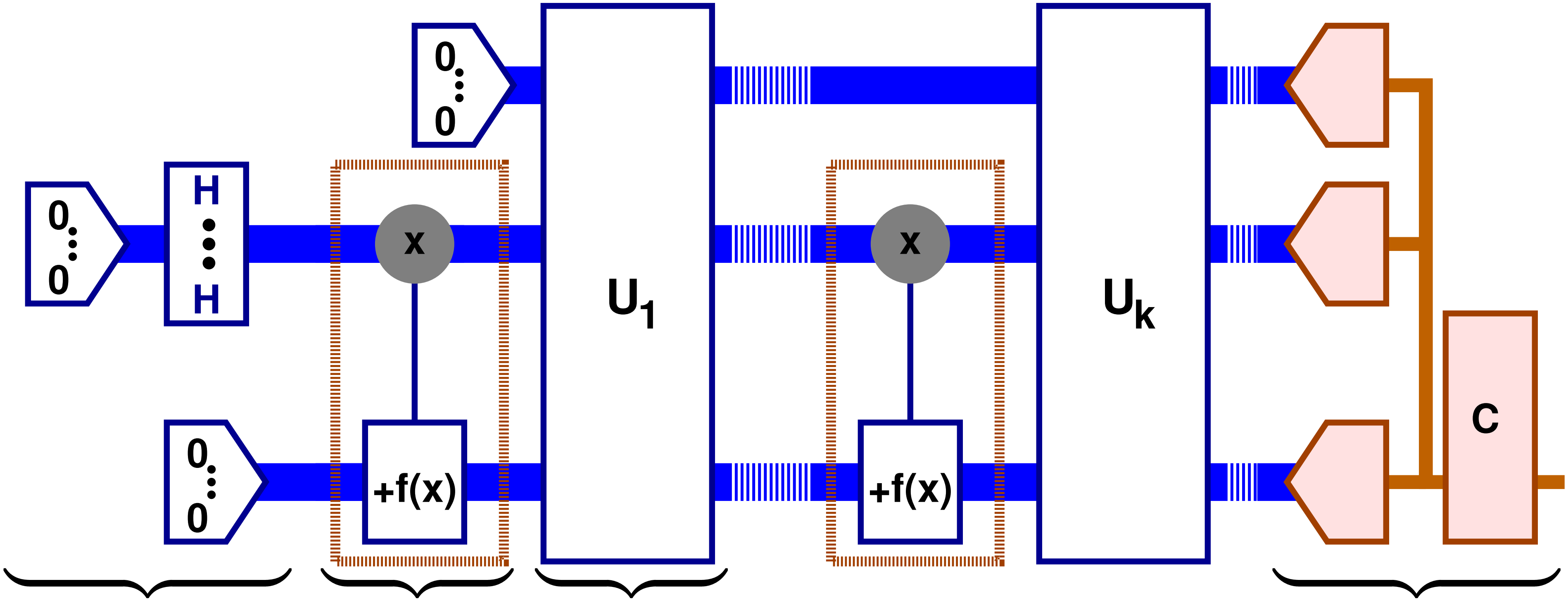
Measurement and classical postprocessing

Recognizing Patterns in Functions

- Algorithm structure for determining some **Property**(f).

Recognizing Patterns in Functions

- Algorithm structure for determining some **Property**(f).
  - Multiple function calls:



State preparation

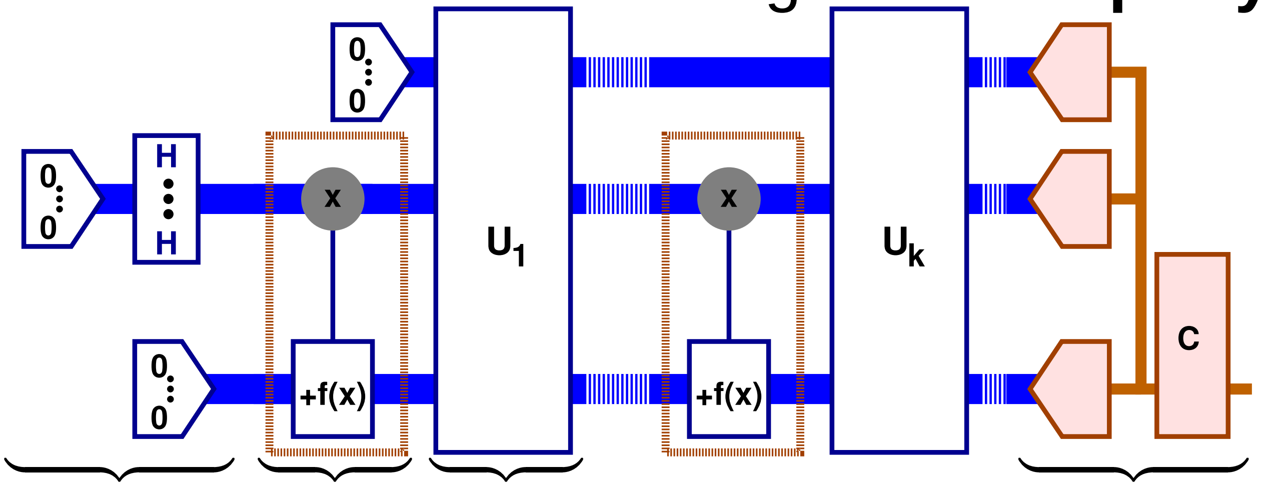
Function call

Quantum interference

Measurement and classical postprocessing

Recognizing Patterns in Functions

- Algorithm structure for determining some **Property**(f).



State preparation

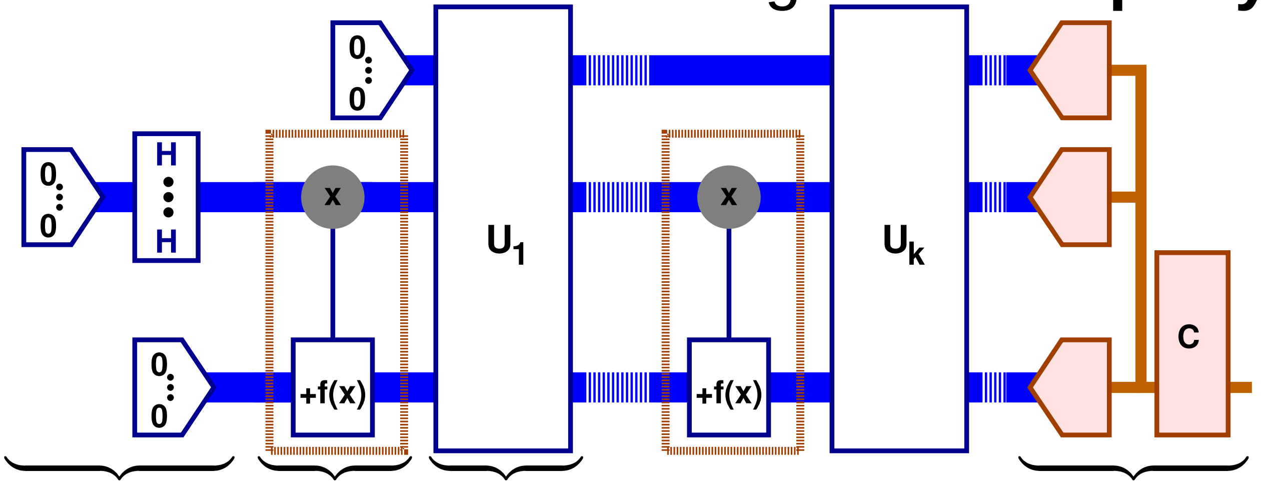
Function call

Quantum interference

Measurement and classical postprocessing

Recognizing Patterns in Functions

- Algorithm structure for determining some **Property**(f).



State preparation

Function call

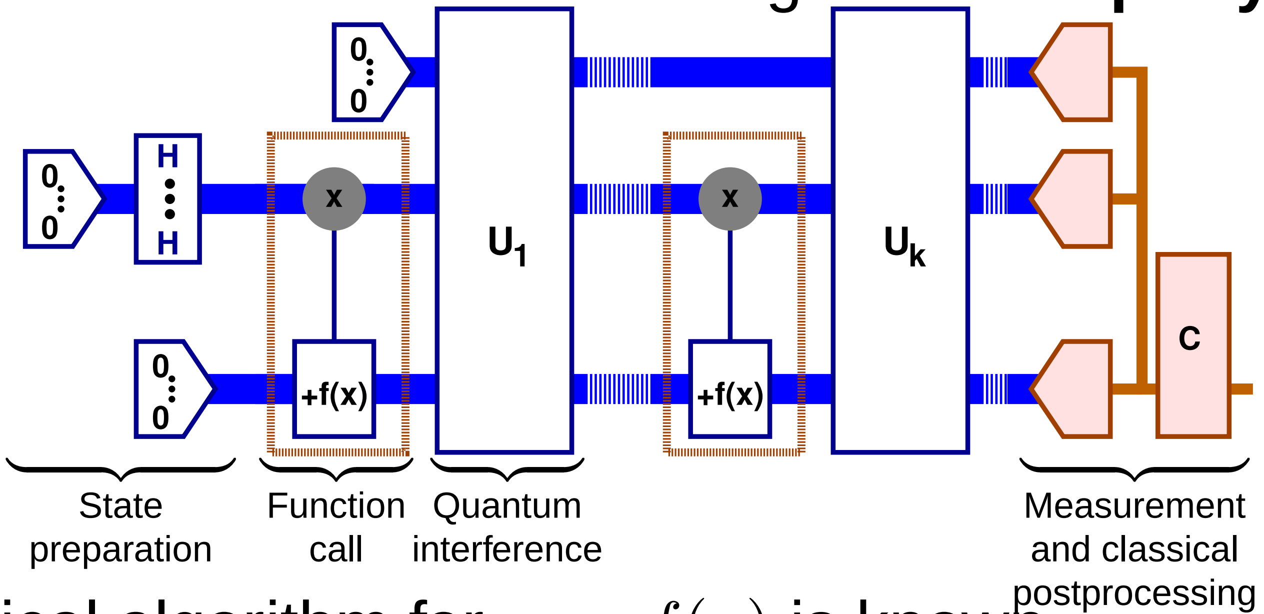
Quantum interference

Measurement and classical postprocessing

- A classical algorithm for $x \mapsto f(x)$ is known.
  - How to implement $\sum_x \alpha_x |xy\rangle \rightarrow \sum_x \alpha_x |x(y \oplus f(x))\rangle$?

Recognizing Patterns in Functions

- Algorithm structure for determining some **Property**(f).

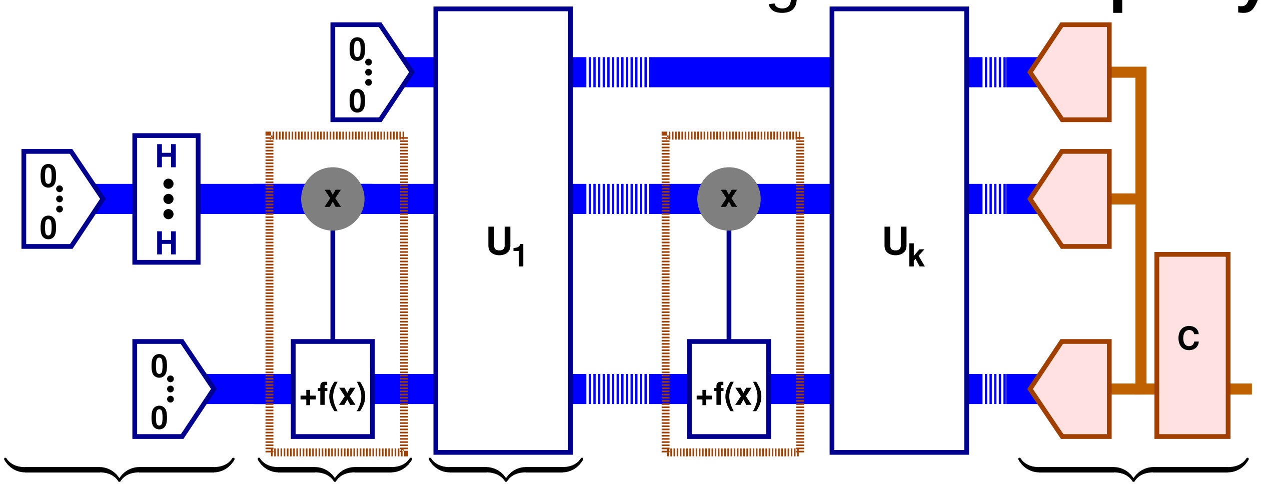


- A classical algorithm for $x \mapsto f(x)$ is known.
  - How to implement $\sum_x \alpha_x|xy\rangle \rightarrow \sum_x \alpha_x|x(y \oplus f(x))\rangle$?
- Solution:

  1. Algorithm → circuit$(|x|)$.

Recognizing Patterns in Functions

- Algorithm structure for determining some **Property**(f).

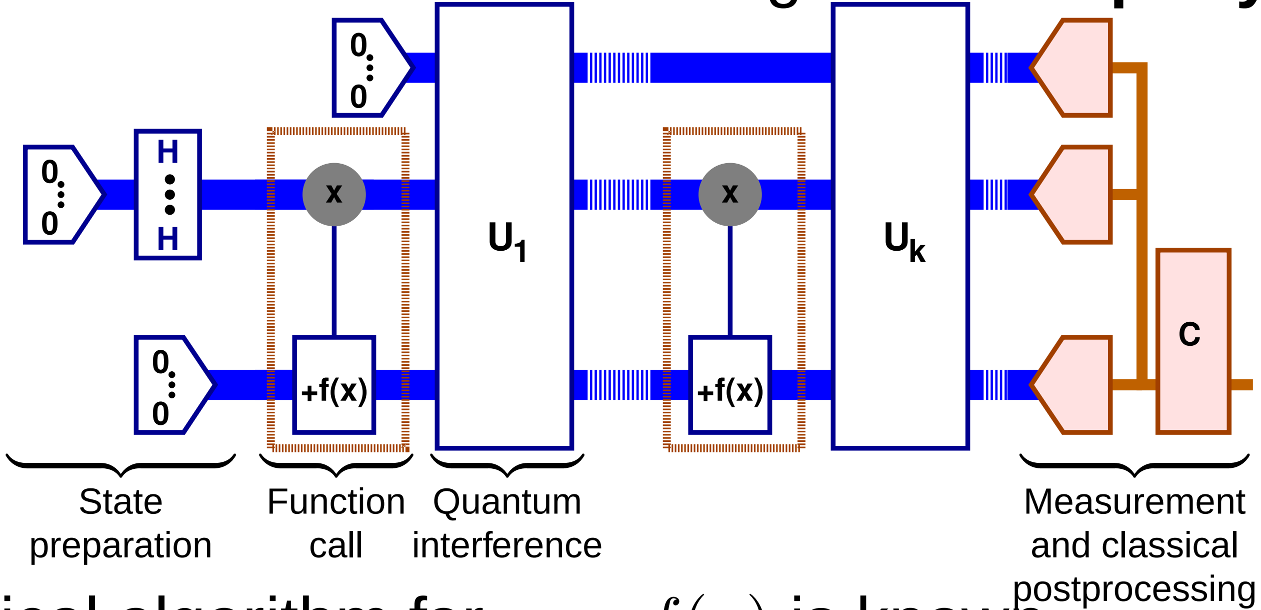


- A classical algorithm for $x \mapsto f(x)$ is known.
  - How to implement $\sum_x \alpha_x |xy\rangle \rightarrow \sum_x \alpha_x |x(y \oplus f(x))\rangle$?
- Solution:

1. Algorithm $\rightarrow$ circuit$(|x|)$.
2. Irreversible gates $\rightarrow$ reversible gates and memory.

Recognizing Patterns in Functions

- Algorithm structure for determining some **Property**(f).

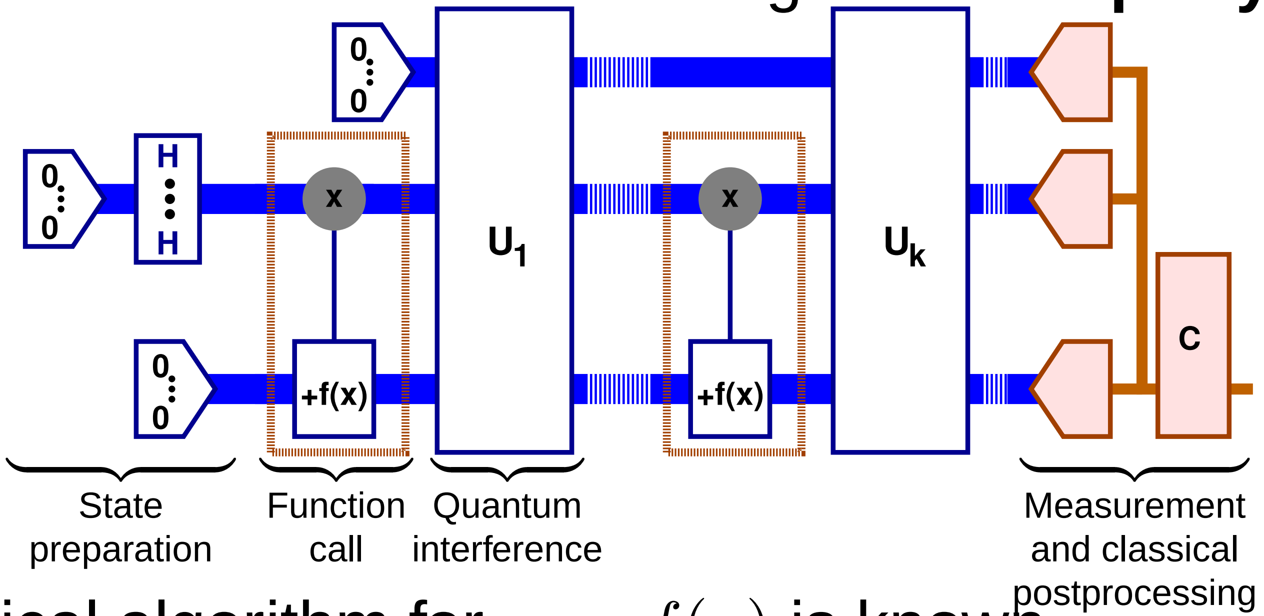


- A classical algorithm for $x \mapsto f(x)$ is known.
  - How to implement $\sum_x \alpha_x|xy\rangle \rightarrow \sum_x \alpha_x|x(y \oplus f(x))\rangle$?
- Solution:

1. Algorithm → circuit$(|x|)$.
2. Irreversible gates → reversible gates and memory.
3. Reversibly erase memory.

Recognizing Patterns in Functions

- Algorithm structure for determining some **Property**(f).



- A classical algorithm for $x \mapsto f(x)$ is known.
  - How to implement $\sum_x \alpha_x|xy\rangle \to \sum_x \alpha_x|x(y \oplus f(x))\rangle$?
- Solution:

1. Algorithm $\to$ circuit$(|x|)$.
2. Irreversible gates $\to$ reversible gates and memory.
3. Reversibly erase memory.
4. Bits $\to$ qubits. Reversible gates $\to$ unitary gates.

Example: Number Comparison

- Algorithm for comparing two binary numbers.

COMP(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \dots x_0$ and $y = y_{n-1} \dots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
$k \leftarrow n - 1$
$c \leftarrow -1$
while $k \geq 0 \ \& \ c < 0$
    if $x_k < y_k$
        $c \leftarrow 0$
    else if $x_k > y_k$
        $c \leftarrow 1$
    end
    $k \leftarrow k - 1$
end
if $c < 0$ then $c \leftarrow 1$
return $c$
```

Example: Number Comparison

- Algorithm for comparing two binary numbers.

COMP(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \ldots x_0$ and $y = y_{n-1} \ldots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
$k \leftarrow n - 1$
$c \leftarrow -1$
while $k \geq 0$ & $c < 0$
    if $x_k < y_k$
        $c \leftarrow 0$
    else if $x_k > y_k$
        $c \leftarrow 1$
    end
    $k \leftarrow k - 1$
end
if $c < 0$ then $c \leftarrow 1$
return $c$
```

k	:						
c	:						
x	:	0	1	0	1	0	1
y	:	0	1	0	0	1	1

Example: Number Comparison

- Algorithm for comparing two binary numbers.

COMP(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \dots x_0$ and $y = y_{n-1} \dots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
$\Rightarrow$  $k \leftarrow n - 1$
    $c \leftarrow -1$
    while $k \geq 0$ & $c < 0$
        if $x_k < y_k$
            $c \leftarrow 0$
        else if $x_k > y_k$
            $c \leftarrow 1$
        end
        $k \leftarrow k - 1$
    end
    if $c < 0$ then $c \leftarrow 1$
    return $c$
```

k : 5

c :

x	:	0	1	0	1	0	1
y	:	0	1	0	0	1	1

Example: Number Comparison

- Algorithm for comparing two binary numbers.

Comp(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \ldots x_0$ and $y = y_{n-1} \ldots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
    $k \leftarrow n - 1$
$\Rightarrow$ $c \leftarrow -1$
    while $k \geq 0$ & $c < 0$
        if $x_k < y_k$
            $c \leftarrow 0$
        else if $x_k > y_k$
            $c \leftarrow 1$
        end
        $k \leftarrow k - 1$
    end
    if $c < 0$ then $c \leftarrow 1$
    return $c$
```

k	:	5					
c	:	-1					
x	:	0	1	0	1	0	1
y	:	0	1	0	0	1	1

Example: Number Comparison

- Algorithm for comparing two binary numbers.

COMP(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \ldots x_0$ and $y = y_{n-1} \ldots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
     $k \leftarrow n - 1$
     $c \leftarrow -1$
$\Longrightarrow$ while $k \geq 0$ & $c < 0$
         if $x_k < y_k$
             $c \leftarrow 0$
         else if $x_k > y_k$
             $c \leftarrow 1$
         end
         $k \leftarrow k - 1$
     end
     if $c < 0$ then $c \leftarrow 1$
     return $c$
```

k	:	5					
c	:	-1					
x	:	0	1	0	1	0	1
y	:	0	1	0	0	1	1

Example: Number Comparison

- Algorithm for comparing two binary numbers.

COMP(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \ldots x_0$ and $y = y_{n-1} \ldots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
    $k \leftarrow n - 1$
    $c \leftarrow -1$
    while $k \geq 0$ & $c < 0$
$\Rightarrow$        if $x_k < y_k$
            $c \leftarrow 0$
        else if $x_k > y_k$
            $c \leftarrow 1$
        end
        $k \leftarrow k - 1$
    end
    if $c < 0$ then $c \leftarrow 1$
    return $c$
```

$$\begin{array}{cc|c|ccccc}
k & : & 5 \\
c & : & -1 \\
\cline{3-3}
x & : & 0 & 1 & 0 & 1 & 0 & 1 \\
y & : & 0 & 1 & 0 & 0 & 1 & 1 \\
\cline{3-3}
\end{array}$$

Example: Number Comparison

- Algorithm for comparing two binary numbers.

COMP(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \ldots x_0$ and $y = y_{n-1} \ldots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
    $k \leftarrow n - 1$
    $c \leftarrow -1$
    **while** $k \geq 0$ & $c < 0$
        **if** $x_k < y_k$
            $c \leftarrow 0$
$\Rightarrow$   **else if** $x_k > y_k$
            $c \leftarrow 1$
        **end**
        $k \leftarrow k - 1$
    **end**
    **if** $c < 0$ **then** $c \leftarrow 1$
    **return** $c$
```

$$\begin{array}{ccl} k & : & 5 \\ c & : & -1 \end{array}$$

$$\begin{array}{cccccccc} x & : & \boxed{0} & 1 & 0 & 1 & 0 & 1 \\ y & : & \boxed{0} & 1 & 0 & 0 & 1 & 1 \end{array}$$

Example: Number Comparison

- Algorithm for comparing two binary numbers.

COMP(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \ldots x_0$ and $y = y_{n-1} \ldots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
    $k \leftarrow n - 1$
    $c \leftarrow -1$
    while $k \geq 0$ & $c < 0$
        if $x_k < y_k$
            $c \leftarrow 0$
        else if $x_k > y_k$
            $c \leftarrow 1$
        end
⇒       $k \leftarrow k - 1$
    end
    if $c < 0$ then $c \leftarrow 1$
    return $c$
```

$$\begin{array}{lcl} k & : & 4 \\ c & : & -1 \end{array}$$

$$\begin{array}{cc|c|ccccc} \cline{3-3} x & : & 0 & 1 & 0 & 1 & 0 & 1 \\ y & : & 0 & 1 & 0 & 0 & 1 & 1 \\ \cline{3-3} \end{array}$$

Example: Number Comparison

- Algorithm for comparing two binary numbers.

COMP(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \dots x_0$ and $y = y_{n-1} \dots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
    $k \leftarrow n - 1$
    $c \leftarrow -1$
$\Longrightarrow$ while $k \geq 0$ & $c < 0$
        if $x_k < y_k$
            $c \leftarrow 0$
        else if $x_k > y_k$
            $c \leftarrow 1$
        end
        $k \leftarrow k - 1$
    end
    if $c < 0$ then $c \leftarrow 1$
    return $c$
```

$$\begin{array}{lcl} k & : & 4 \\ c & : & -1 \end{array}$$

$$\begin{array}{cc|c|ccccc} \cline{3-3} x & : & 0 & 1 & 0 & 1 & 0 & 1 \\ y & : & 0 & 1 & 0 & 0 & 1 & 1 \\ \cline{3-3} \end{array}$$

Example: Number Comparison

- Algorithm for comparing two binary numbers.

COMP(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \ldots x_0$ and $y = y_{n-1} \ldots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
    $k \leftarrow n - 1$
    $c \leftarrow -1$
    while $k \geq 0$ & $c < 0$
⇒       if $x_k < y_k$
            $c \leftarrow 0$
        else if $x_k > y_k$
            $c \leftarrow 1$
        end
        $k \leftarrow k - 1$
    end
    if $c < 0$ then $c \leftarrow 1$
    return $c$
```

$$\begin{array}{ccc} k & : & 4 \\ c & : & -1 \end{array}$$

$$\begin{array}{ccc|c|cccc} x & : & 0 & 1 & 0 & 1 & 0 & 1 \\ y & : & 0 & 1 & 0 & 0 & 1 & 1 \end{array}$$

Example: Number Comparison

- Algorithm for comparing two binary numbers.

COMP(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \dots x_0$ and $y = y_{n-1} \dots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
  $k \leftarrow n - 1$
  $c \leftarrow -1$
  while $k \geq 0$ & $c < 0$
      if $x_k < y_k$
          $c \leftarrow 0$
⇒     else if $x_k > y_k$
          $c \leftarrow 1$
      end
      $k \leftarrow k - 1$
  end
  if $c < 0$ then $c \leftarrow 1$
  return $c$
```

$$
\begin{array}{ccc}
k & : & 4 \\
c & : & -1
\end{array}
$$

$$
\begin{array}{cc c|c|cccc}
x & : & 0 & 1 & 0 & 1 & 0 & 1 \\
y & : & 0 & 1 & 0 & 0 & 1 & 1
\end{array}
$$

Example: Number Comparison

- Algorithm for comparing two binary numbers.

COMP(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \ldots x_0$ and $y = y_{n-1} \ldots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
    $k \leftarrow n - 1$
    $c \leftarrow -1$
    while $k \geq 0$ & $c < 0$
        if $x_k < y_k$
            $c \leftarrow 0$
        else if $x_k > y_k$
            $c \leftarrow 1$
        end
$\Rightarrow$      $k \leftarrow k - 1$
    end
    if $c < 0$ then $c \leftarrow 1$
    return $c$
```

$$k : 3$$

$$c : -1$$

$$\begin{array}{ccc|c|cccc} x & : & 0 & 1 & 0 & 1 & 0 & 1 \\ y & : & 0 & 1 & 0 & 0 & 1 & 1 \end{array}$$

Example: Number Comparison

- Algorithm for comparing two binary numbers.

COMP(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \ldots x_0$ and $y = y_{n-1} \ldots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
         $k \leftarrow n - 1$
         $c \leftarrow -1$
$\Rightarrow$   while $k \geq 0$ & $c < 0$
             if $x_k < y_k$
                 $c \leftarrow 0$
             else if $x_k > y_k$
                 $c \leftarrow 1$
             end
             $k \leftarrow k - 1$
         end
         if $c < 0$ then $c \leftarrow 1$
         return $c$
```

$$\begin{array}{ccl} k & : & 3 \\ c & : & -1 \end{array}$$

$$\begin{array}{ccc|c|cccc} x & : & 0 & 1 & 0 & 1 & 0 & 1 \\ y & : & 0 & 1 & 0 & 0 & 1 & 1 \end{array}$$

Example: Number Comparison

- Algorithm for comparing two binary numbers.

COMP(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \ldots x_0$ and $y = y_{n-1} \ldots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
    $k \leftarrow n - 1$
    $c \leftarrow -1$
    while $k \geq 0$ & $c < 0$
$\Rightarrow$    if $x_k < y_k$
            $c \leftarrow 0$
        else if $x_k > y_k$
            $c \leftarrow 1$
        end
        $k \leftarrow k - 1$
    end
    if $c < 0$ then $c \leftarrow 1$
    return $c$
```

$$\begin{array}{lcl} k & : & 3 \\ c & : & -1 \end{array}$$

$$\begin{array}{ccccc|c|ccc} x & : & 0 & 1 & 0 & 1 & 0 & 1 \\ y & : & 0 & 1 & 0 & 0 & 1 & 1 \end{array}$$

Example: Number Comparison

- Algorithm for comparing two binary numbers.

Comp(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \dots x_0$ and $y = y_{n-1} \dots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
    $k \leftarrow n - 1$
    $c \leftarrow -1$
    while $k \geq 0$ & $c < 0$
        if $x_k < y_k$
            $c \leftarrow 0$
⇒       else if $x_k > y_k$
            $c \leftarrow 1$
        end
        $k \leftarrow k - 1$
    end
    if $c < 0$ then $c \leftarrow 1$
    return $c$
```

$$\begin{array}{ccl} k & : & 3 \\ c & : & -1 \end{array}$$

$$\begin{array}{cccc|c|ccc} x & : & 0 & 1 & 0 & 1 & 0 & 1 \\ y & : & 0 & 1 & 0 & 0 & 1 & 1 \end{array}$$

Example: Number Comparison

- Algorithm for comparing two binary numbers.

COMP(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \ldots x_0$ and $y = y_{n-1} \ldots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
$k \leftarrow n - 1$
$c \leftarrow -1$
while $k \geq 0$ & $c < 0$
    if $x_k < y_k$
        $c \leftarrow 0$
    else if $x_k > y_k$
        $c \leftarrow 1$
    end
⇒   $k \leftarrow k - 1$
end
if $c < 0$ then $c \leftarrow 1$
return $c$
```

$$\begin{array}{ccc}
k & : & 2 \\
c & : & -1
\end{array}$$

$$\begin{array}{cccc|c|ccc}
x & : & 0 & 1 & 0 & 1 & 0 & 1 \\
y & : & 0 & 1 & 0 & 0 & 1 & 1
\end{array}$$

Example: Number Comparison

- Algorithm for comparing two binary numbers.

COMP(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \ldots x_0$ and $y = y_{n-1} \ldots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
    $k \leftarrow n - 1$
    $c \leftarrow -1$
$\Rightarrow$ while $k \geq 0 \ \& \ c < 0$
        if $x_k < y_k$
            $c \leftarrow 0$
        else if $x_k > y_k$
            $c \leftarrow 1$
        end
        $k \leftarrow k - 1$
    end
    if $c < 0$ then $c \leftarrow 1$
    return $c$
```

$k \ : \ 2$

$c \ : \ -1$

$$\begin{array}{cccccccc} x & : & 0 & 1 & \boxed{0} & 1 & 0 & 1 \\ y & : & 0 & 1 & \boxed{0} & 0 & 1 & 1 \end{array}$$

Example: Number Comparison

- Algorithm for comparing two binary numbers.

COMP(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \dots x_0$ and $y = y_{n-1} \dots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
    $k \leftarrow n - 1$
    $c \leftarrow -1$
    while $k \geq 0$ & $c < 0$
⇒       if $x_k < y_k$
            $c \leftarrow 0$
        else if $x_k > y_k$
            $c \leftarrow 1$
        end
        $k \leftarrow k - 1$
    end
    if $c < 0$ then $c \leftarrow 1$
    return $c$
```

$$\begin{array}{ccl} k & : & 2 \\ c & : & -1 \end{array}$$

$$\begin{array}{cccccccc} x & : & 0 & 1 & 0 & \multirow{2}{*}{$\boxed{\begin{array}{c} 1 \\ 0 \end{array}}$} & 0 & 1 \\ y & : & 0 & 1 & 0 & & 1 & 1 \end{array}$$

Example: Number Comparison

- Algorithm for comparing two binary numbers.

```
COMP($x, y$)
Input: Zero-filled $n$-bit numbers $x = x_{n-1} \ldots x_0$ and $y = y_{n-1} \ldots y_0$.
Output:  $0$ if $x < y$ and $1$ if $x \geq y$.

    $k \leftarrow n - 1$
    $c \leftarrow -1$
    while $k \geq 0$ & $c < 0$
        if $x_k < y_k$
            $c \leftarrow 0$
⇒       else if $x_k > y_k$
            $c \leftarrow 1$
        end
        $k \leftarrow k - 1$
    end
    if $c < 0$ then $c \leftarrow 1$
    return $c$
```

$$\begin{array}{ccl} k & : & 2 \\ c & : & -1 \end{array}$$

$$\begin{array}{cccccc|c|cc} x & : & 0 & 1 & 0 & 1 & 0 & 1 \\ y & : & 0 & 1 & 0 & 0 & 1 & 1 \end{array}$$

Example: Number Comparison

- Algorithm for comparing two binary numbers.

COMP(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \dots x_0$ and $y = y_{n-1} \dots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
    $k \leftarrow n - 1$
    $c \leftarrow -1$
    while $k \geq 0$ & $c < 0$
        if $x_k < y_k$
            $c \leftarrow 0$
        else if $x_k > y_k$
$\Rightarrow$       $c \leftarrow 1$
        end
        $k \leftarrow k - 1$
    end
    if $c < 0$ then $c \leftarrow 1$
    return $c$
```

$$\begin{array}{ccc} k & : & 2 \\ c & : & 1 \end{array}$$

$$\begin{array}{cccccc|c|cc} x & : & 0 & 1 & 0 & & 1 & 0 & 1 \\ y & : & 0 & 1 & 0 & & 0 & 1 & 1 \end{array}$$

Example: Number Comparison

- Algorithm for comparing two binary numbers.

COMP(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \ldots x_0$ and $y = y_{n-1} \ldots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
   $k \leftarrow n - 1$
   $c \leftarrow -1$
   while $k \geq 0$ & $c < 0$
       if $x_k < y_k$
           $c \leftarrow 0$
       else if $x_k > y_k$
           $c \leftarrow 1$
       end
$\Rightarrow$  $k \leftarrow k - 1$
   end
   if $c < 0$ then $c \leftarrow 1$
   return $c$
```

$$\begin{array}{ccl} k & : & 1 \\ c & : & 1 \end{array}$$

$$\begin{array}{cccccc|c|cc} x & : & 0 & 1 & 0 & 1 & 0 & 1 \\ y & : & 0 & 1 & 0 & 0 & 1 & 1 \end{array}$$

Example: Number Comparison

- Algorithm for comparing two binary numbers.

COMP(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \ldots x_0$ and $y = y_{n-1} \ldots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
     $k \leftarrow n - 1$
     $c \leftarrow -1$
$\Rightarrow$   while $k \geq 0$ & $c < 0$
         if $x_k < y_k$
             $c \leftarrow 0$
         else if $x_k > y_k$
             $c \leftarrow 1$
         end
         $k \leftarrow k - 1$
     end
     if $c < 0$ then $c \leftarrow 1$
     return $c$
```

k : 1

c : 1

x	:	0	1	0	$\boxed{1}$	0	1
y	:	0	1	0	$\boxed{0}$	1	1

Example: Number Comparison

- Algorithm for comparing two binary numbers.

COMP(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \dots x_0$ and $y = y_{n-1} \dots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
    $k \leftarrow n - 1$
    $c \leftarrow -1$
    **while** $k \geq 0$ & $c < 0$
        **if** $x_k < y_k$
            $c \leftarrow 0$
        **else if** $x_k > y_k$
            $c \leftarrow 1$
        **end**
        $k \leftarrow k - 1$
    **end**
⇒   **if** $c < 0$ **then** $c \leftarrow 1$
    **return** $c$
```

$$\begin{array}{lcl} k & : & 1 \\ c & : & 1 \end{array}$$

$$\begin{array}{lcccc|c|cc} x & : & 0 & 1 & 0 & 1 & 0 & 1 \\ y & : & 0 & 1 & 0 & 0 & 1 & 1 \end{array}$$

Example: Number Comparison

- Algorithm for comparing two binary numbers.

COMP(x, y)
Input: Zero-filled n-bit numbers $x = x_{n-1} \ldots x_0$ and $y = y_{n-1} \ldots y_0$.
Output: 0 if $x < y$ and 1 if $x \geq y$.

```
    $k \leftarrow n - 1$
    $c \leftarrow -1$
    while $k \geq 0$ & $c < 0$
        if $x_k < y_k$
            $c \leftarrow 0$
        else if $x_k > y_k$
            $c \leftarrow 1$
        end
        $k \leftarrow k - 1$
    end
    if $c < 0$ then $c \leftarrow 1$
$\Rightarrow$ return $c$
```

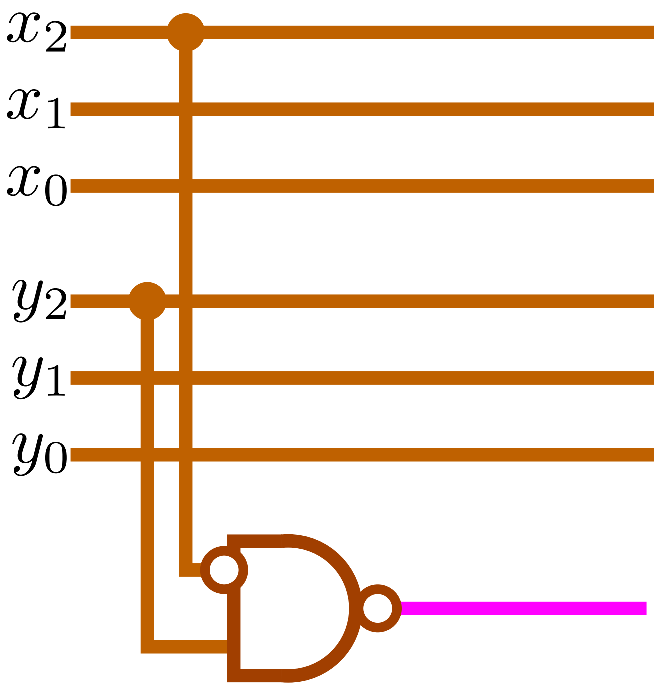
k : 1

c : 1

x	:	0	1	0	$\boxed{1}$	0	1
y	:	0	1	0	$\boxed{0}$	1	1

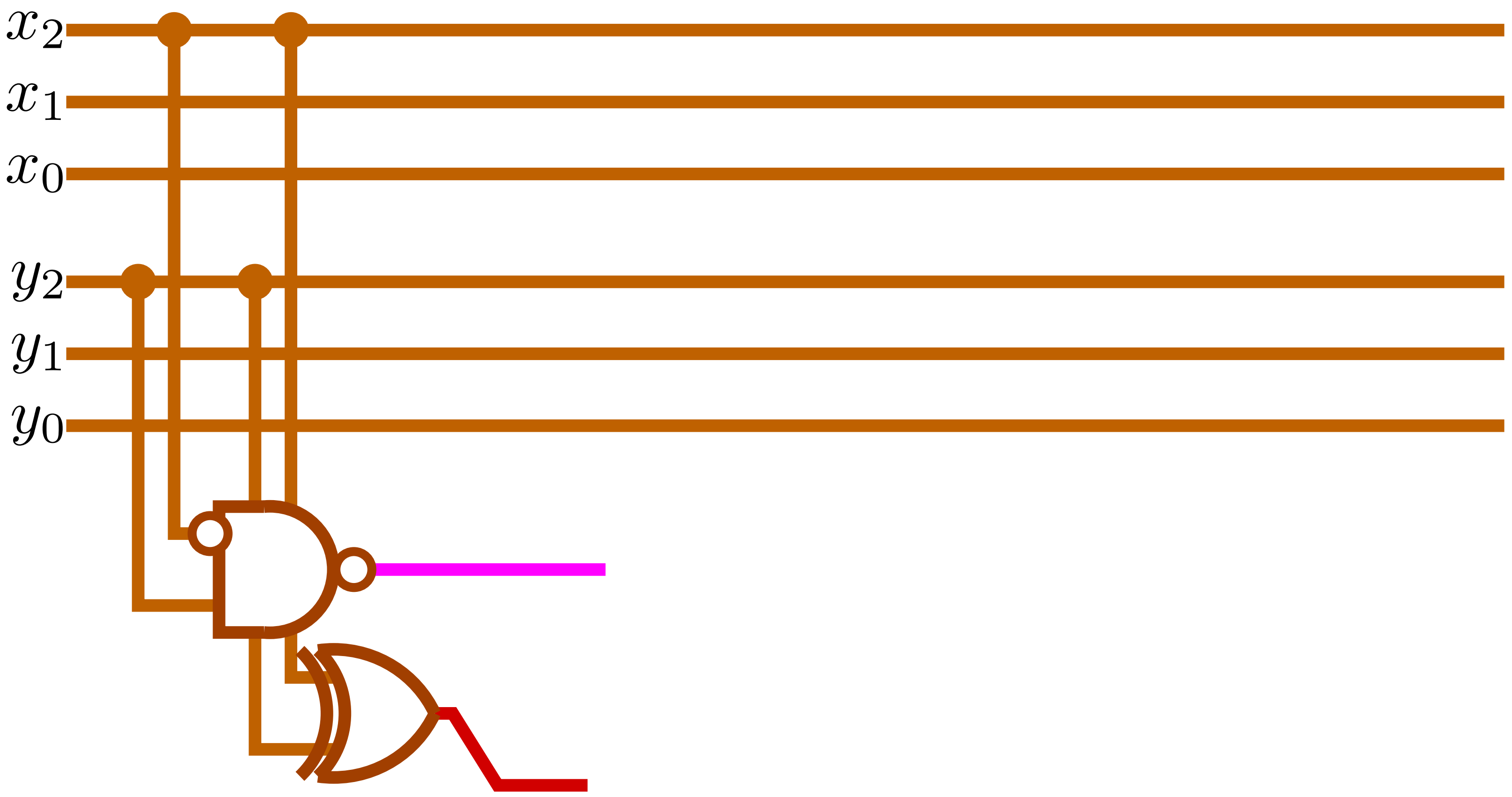
Classical Comparison Circuit

- 3 bit comparison circuit for “$\textbf{if } x < y \textbf{ then } 0 \textbf{ else } 1$”.



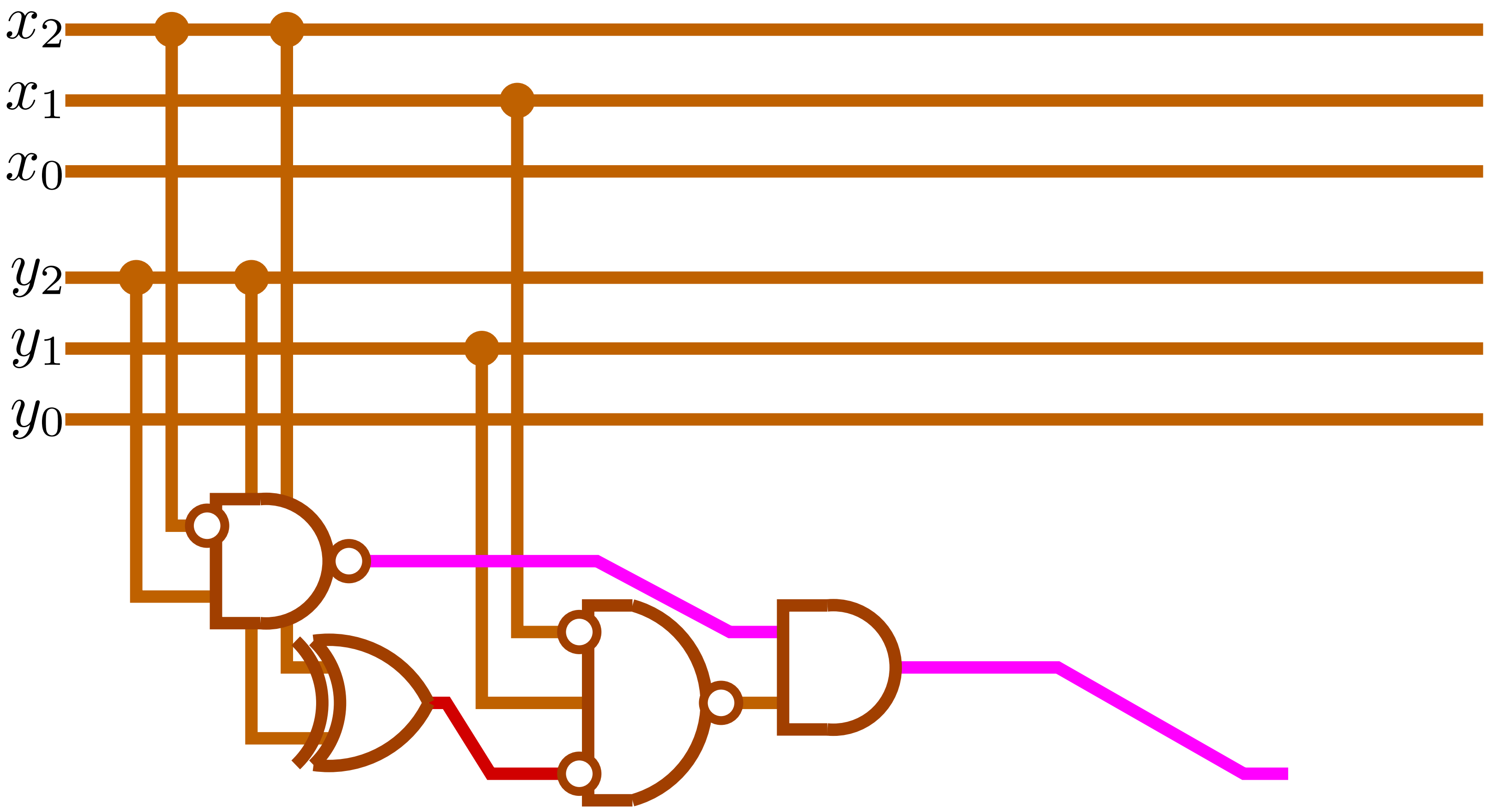
Classical Comparison Circuit

- 3 bit comparison circuit for "**if** $x < y$ **then** 0 **else** 1".



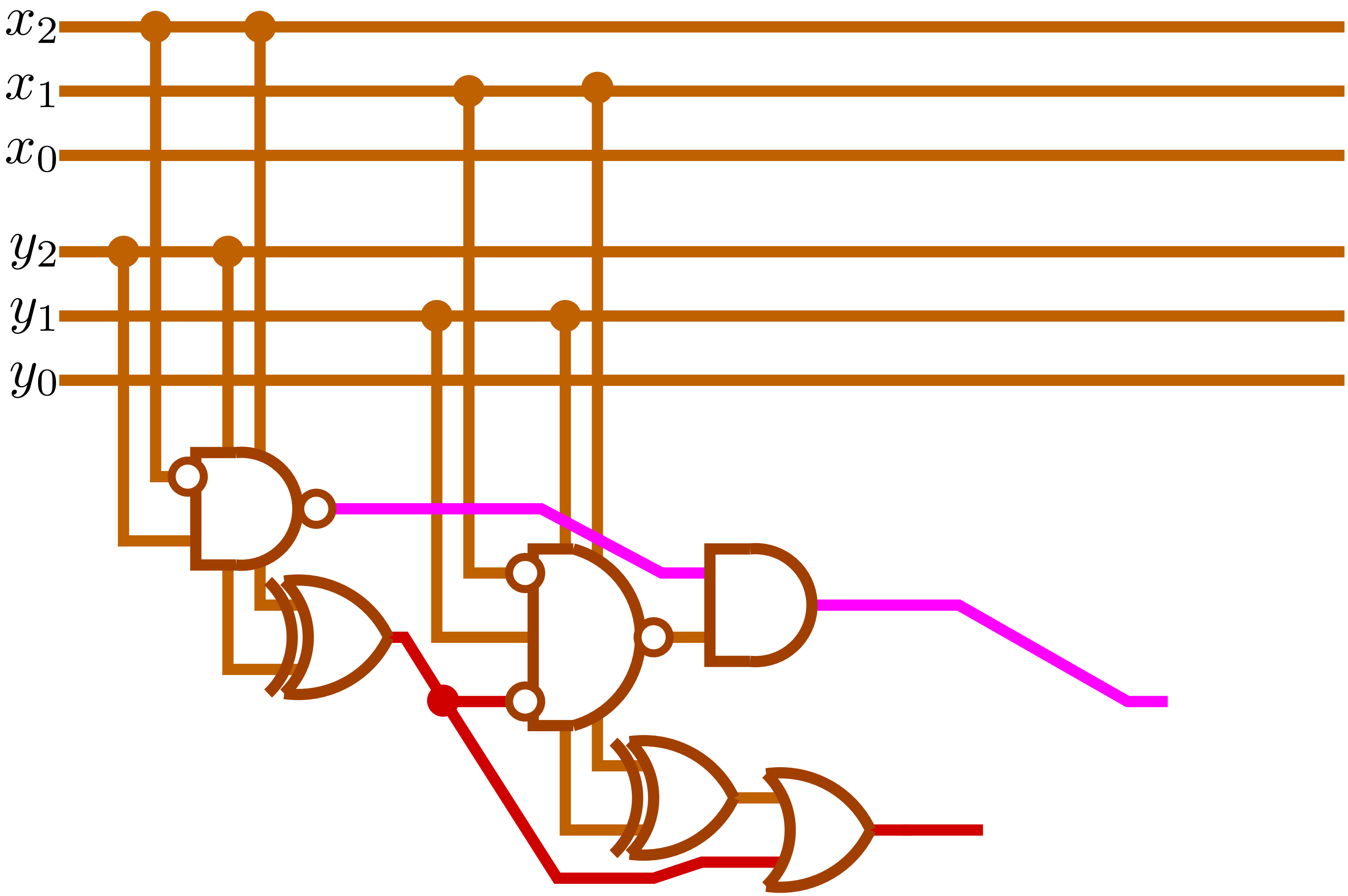
Classical Comparison Circuit

- 3 bit comparison circuit for “**if** $x < y$ **then** 0 **else** 1”.



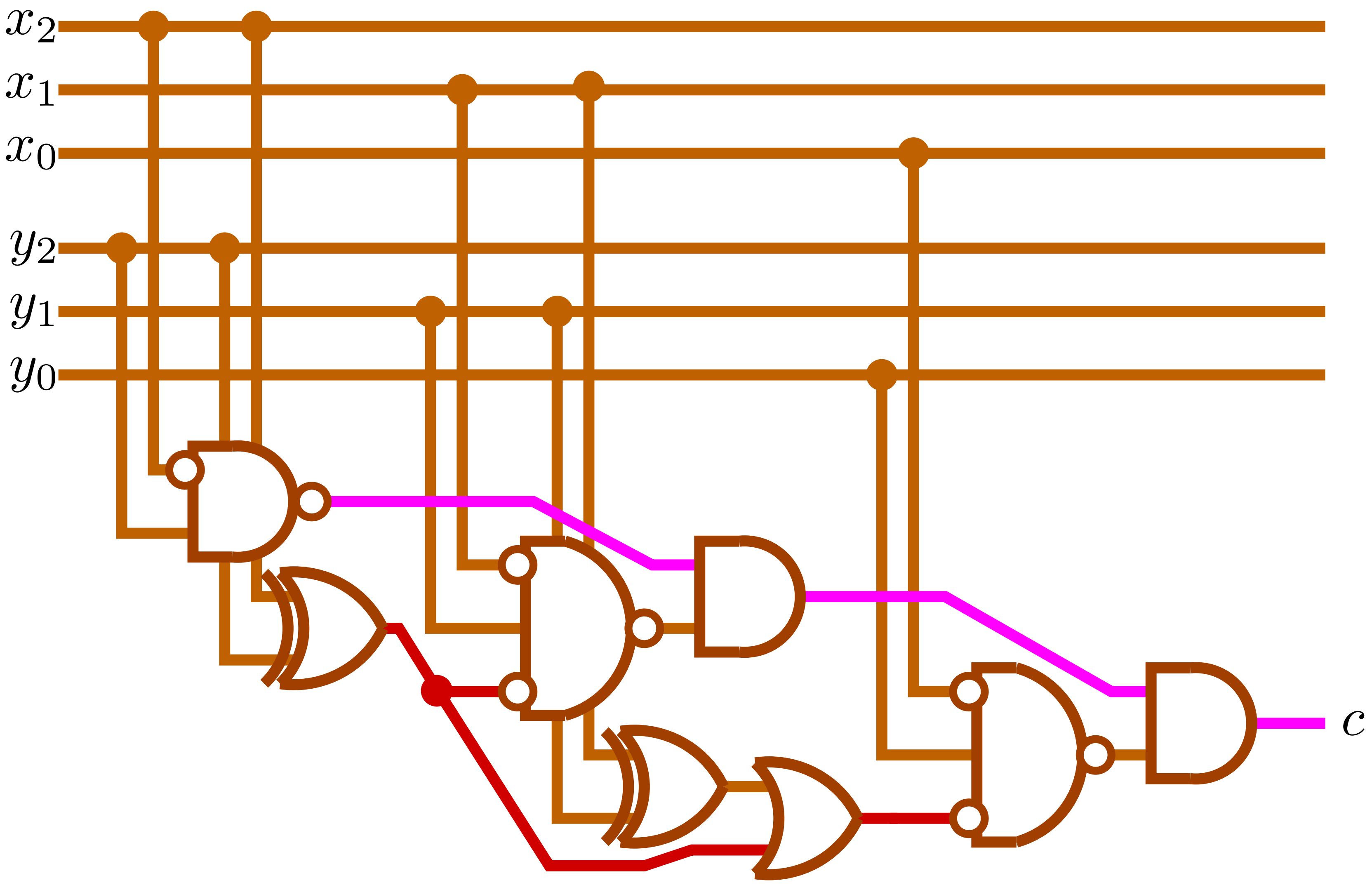
Classical Comparison Circuit

- 3 bit comparison circuit for “$\textbf{if } x < y \textbf{ then } 0 \textbf{ else } 1$”.



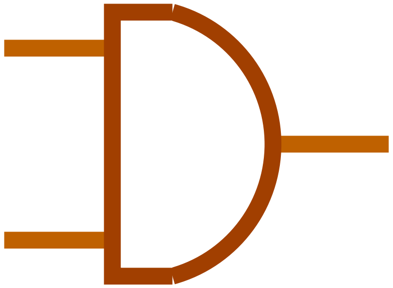
Classical Comparison Circuit

- 3 bit comparison circuit for "**if** $x < y$ **then** 0 **else** 1".



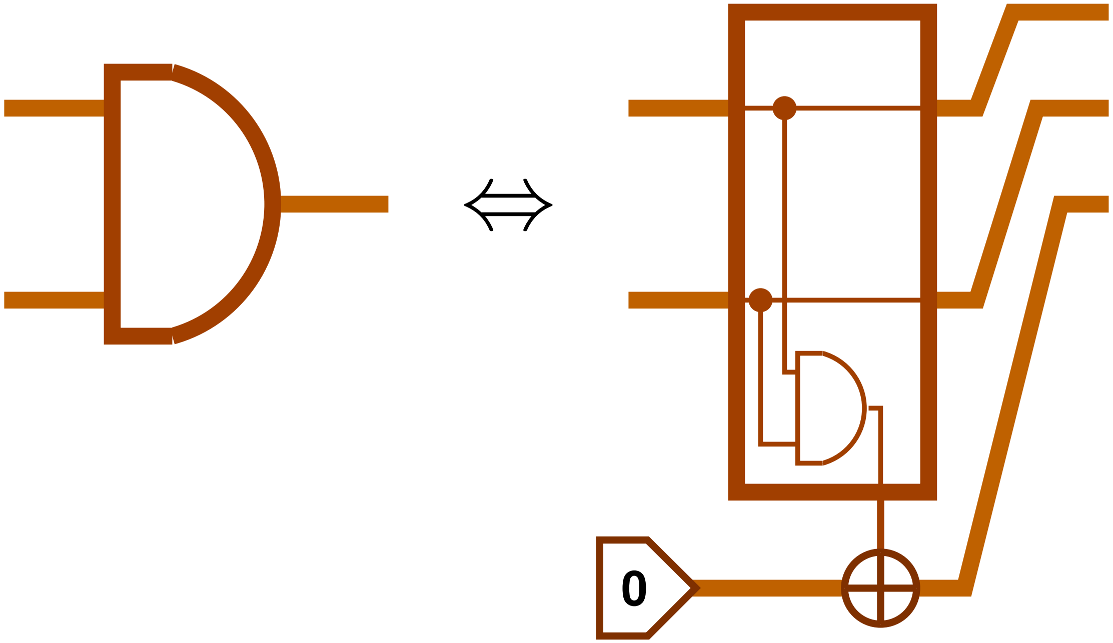
Converting to Reversible Logic

- Reversifying the and gate.



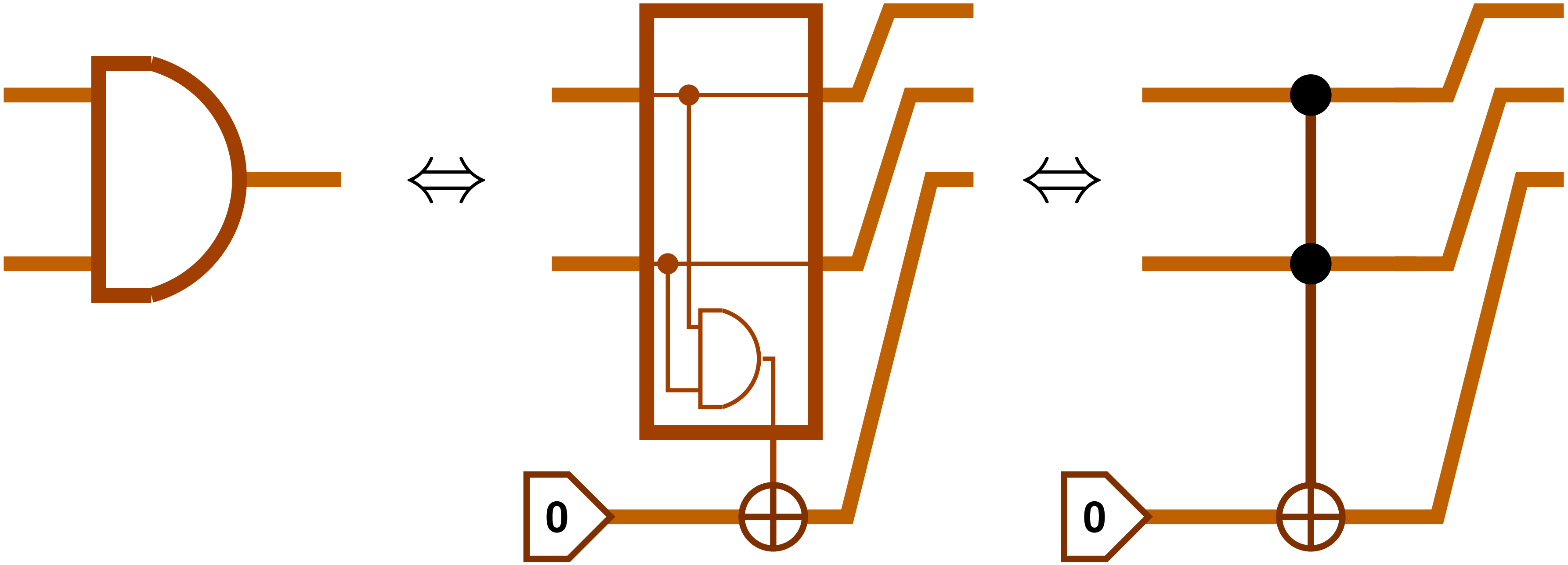
Converting to Reversible Logic

- Reversifying the **and** gate.



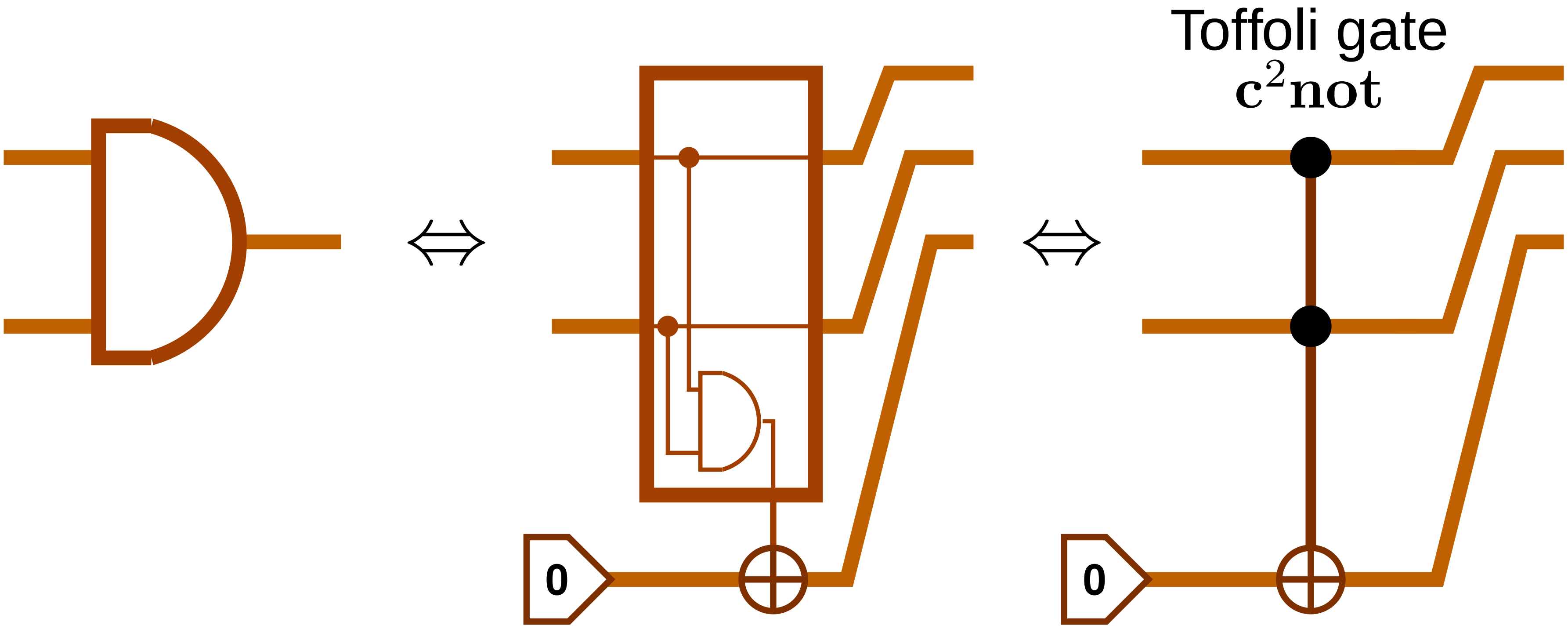
Converting to Reversible Logic

- Reversifying the **and** gate.



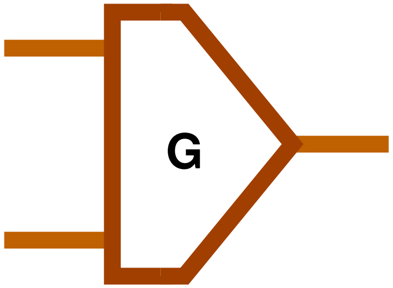
Converting to Reversible Logic

- Reversifying the **and** gate.



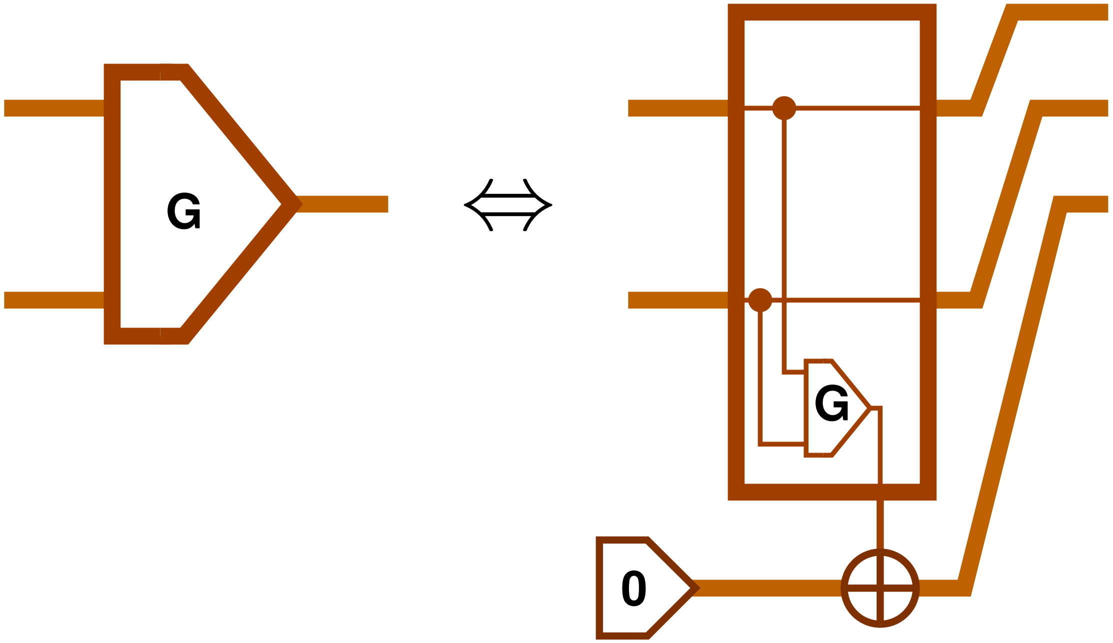
Converting to Reversible Logic

- Reversifying a general gate.



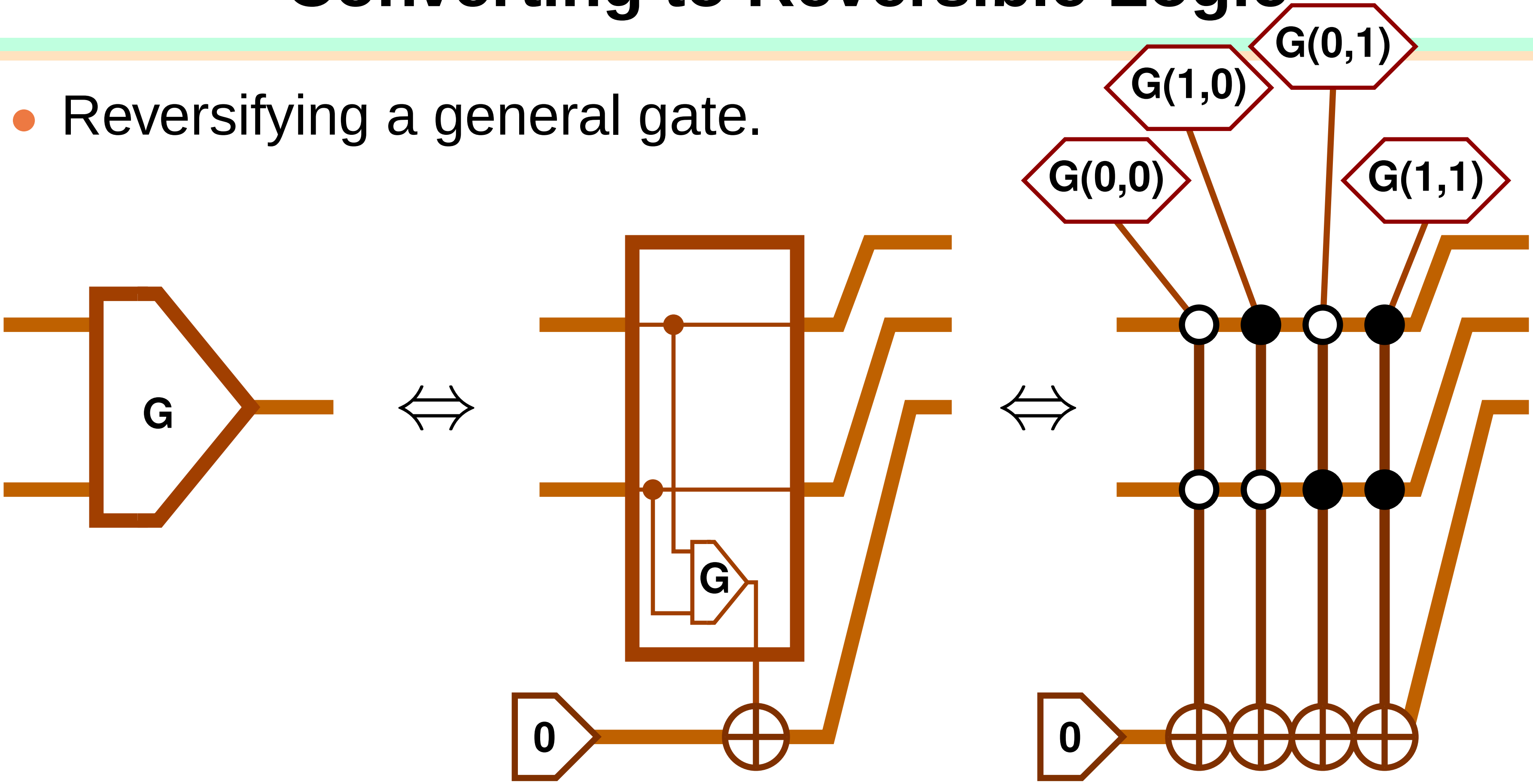
Converting to Reversible Logic

- Reversifying a general gate.



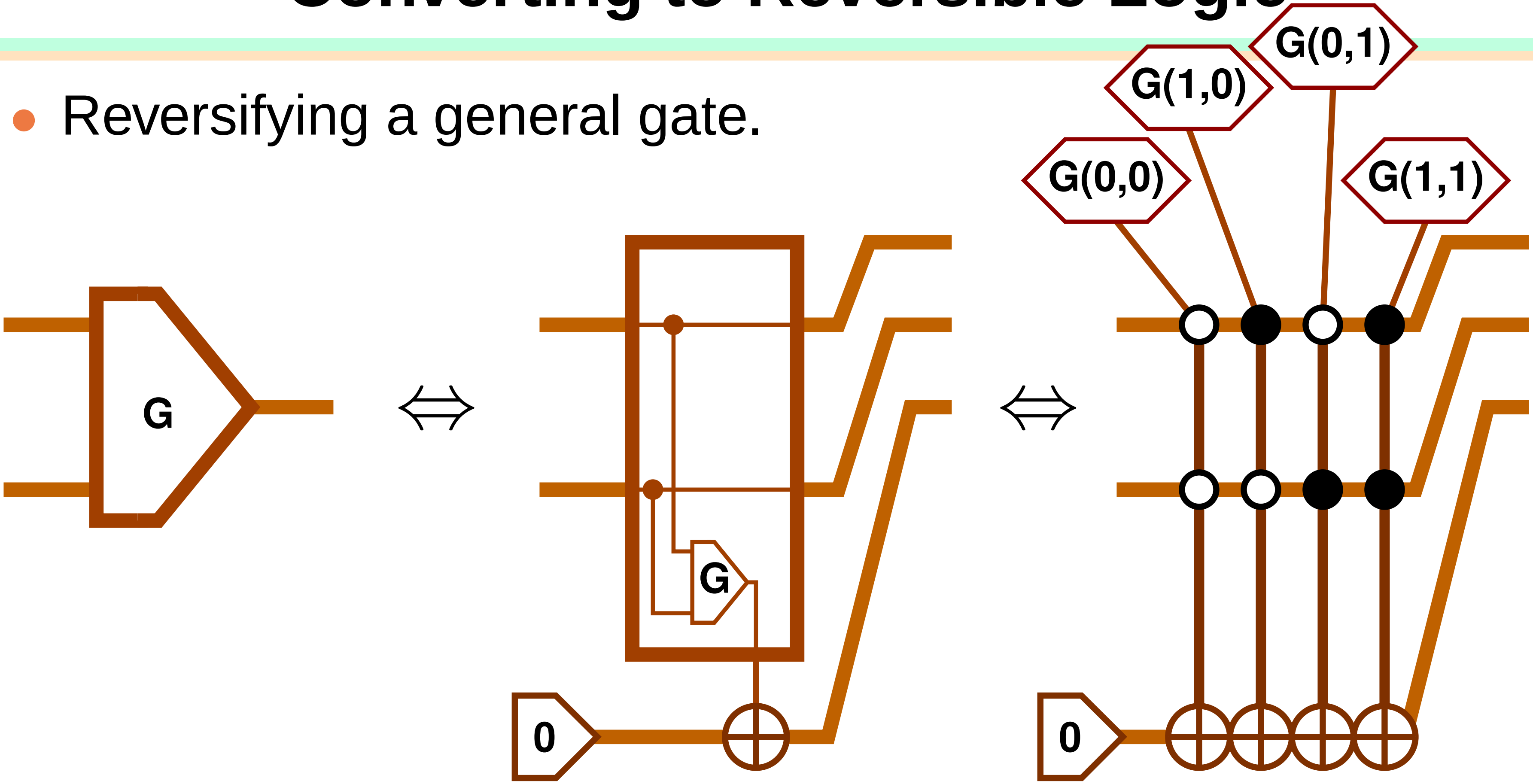
Converting to Reversible Logic

- Reversifying a general gate.



Converting to Reversible Logic

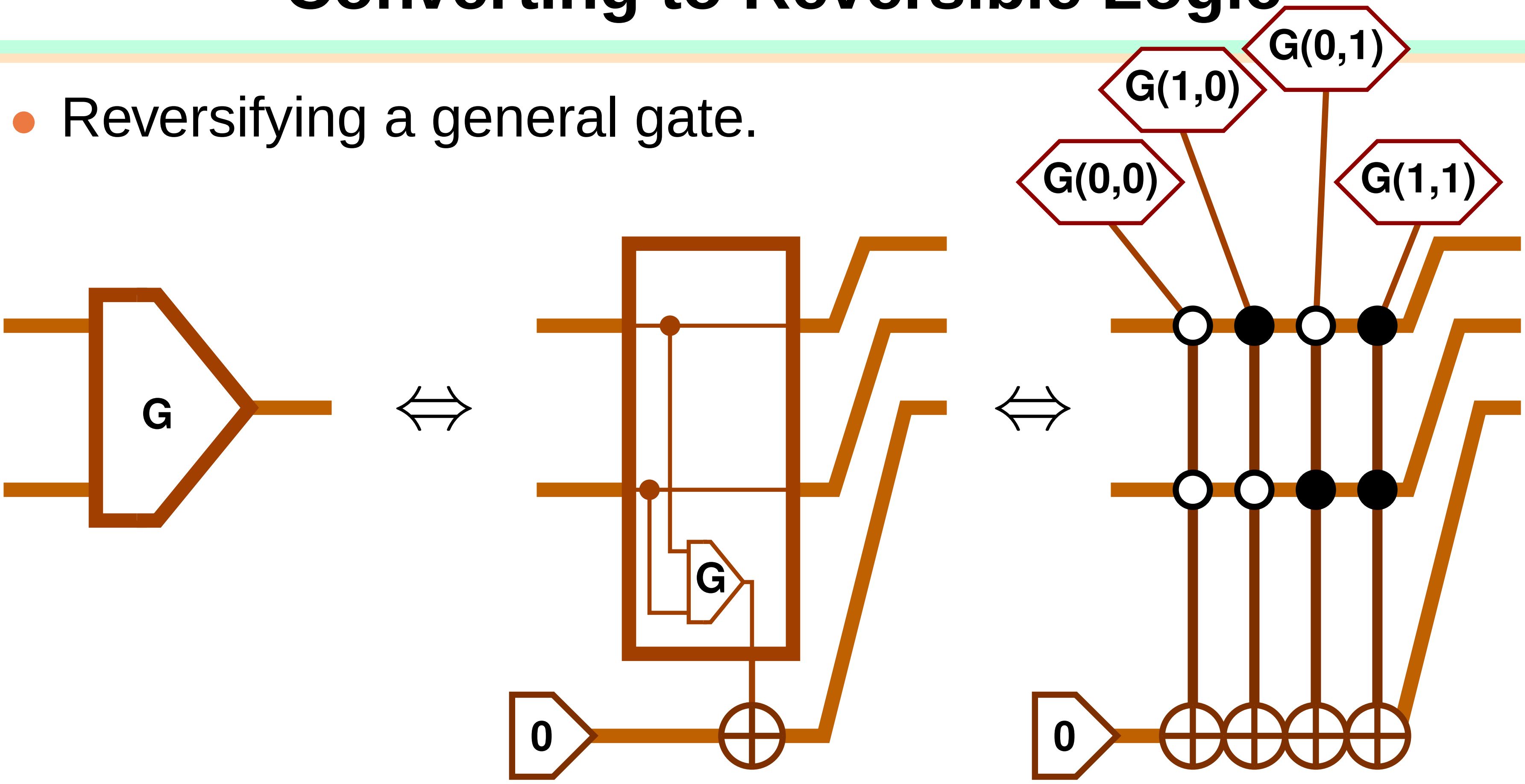
- Reversifying a general gate.



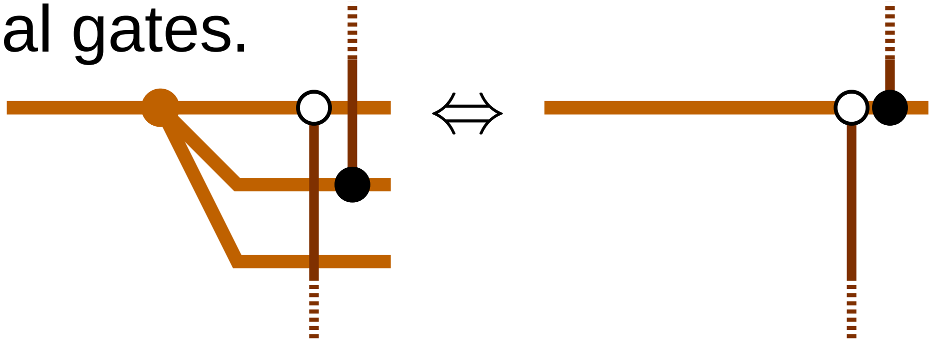
- . . . then optimize the conditional gates.

Converting to Reversible Logic

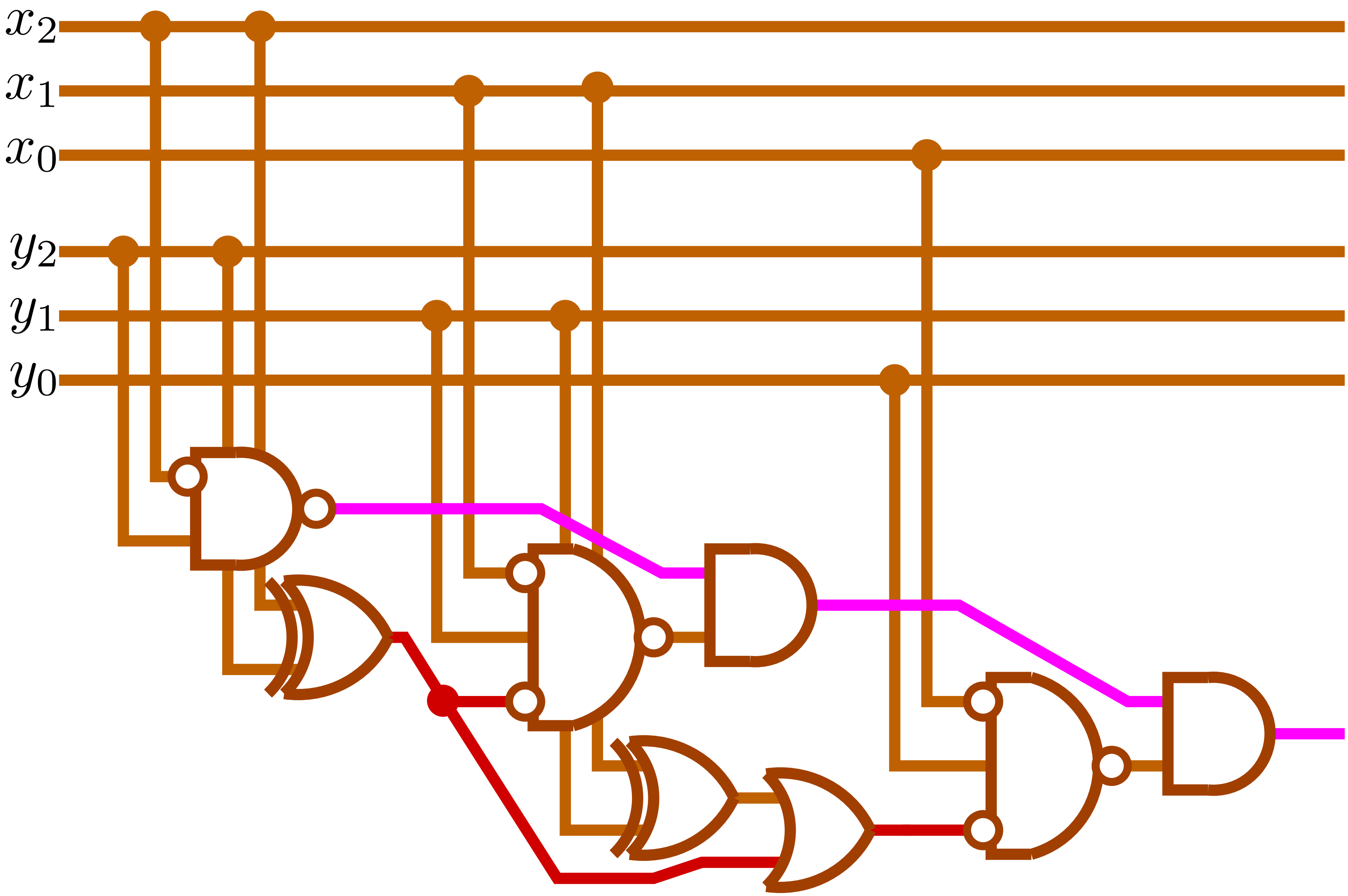
- Reversifying a general gate.



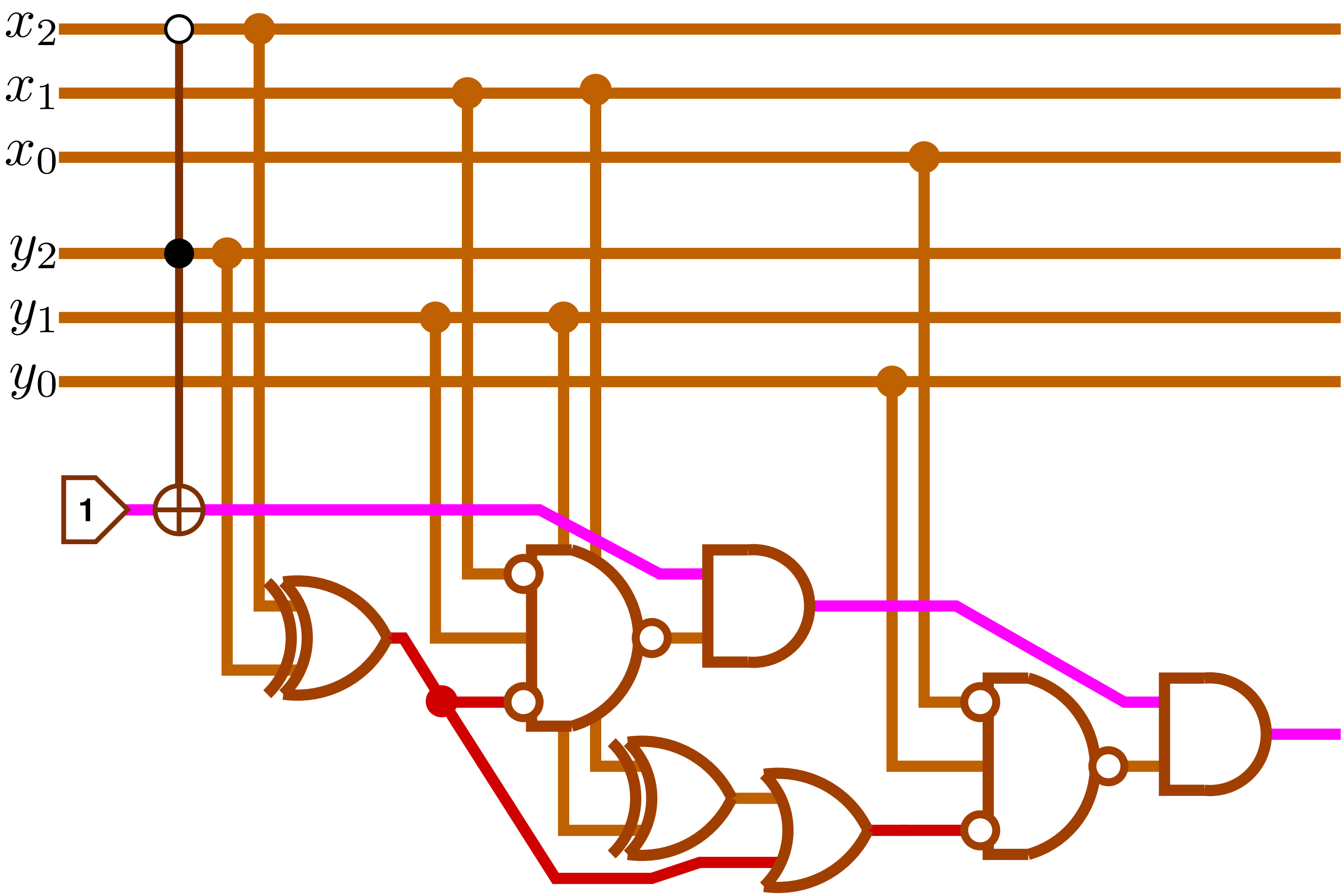
- . . . then optimize the conditional gates.
- Remove redundant fanout.



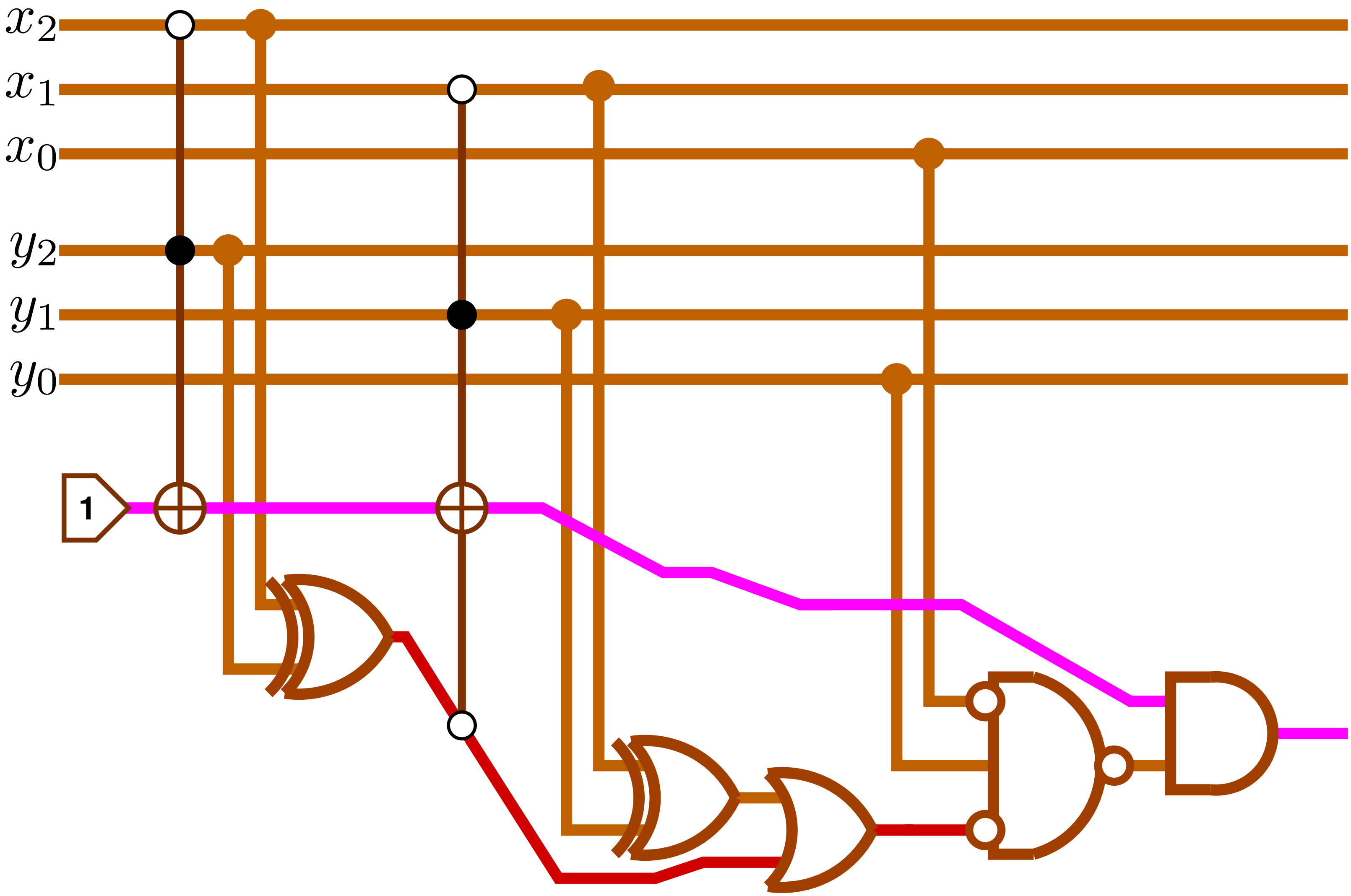
Reversifying the Comparison Circuit



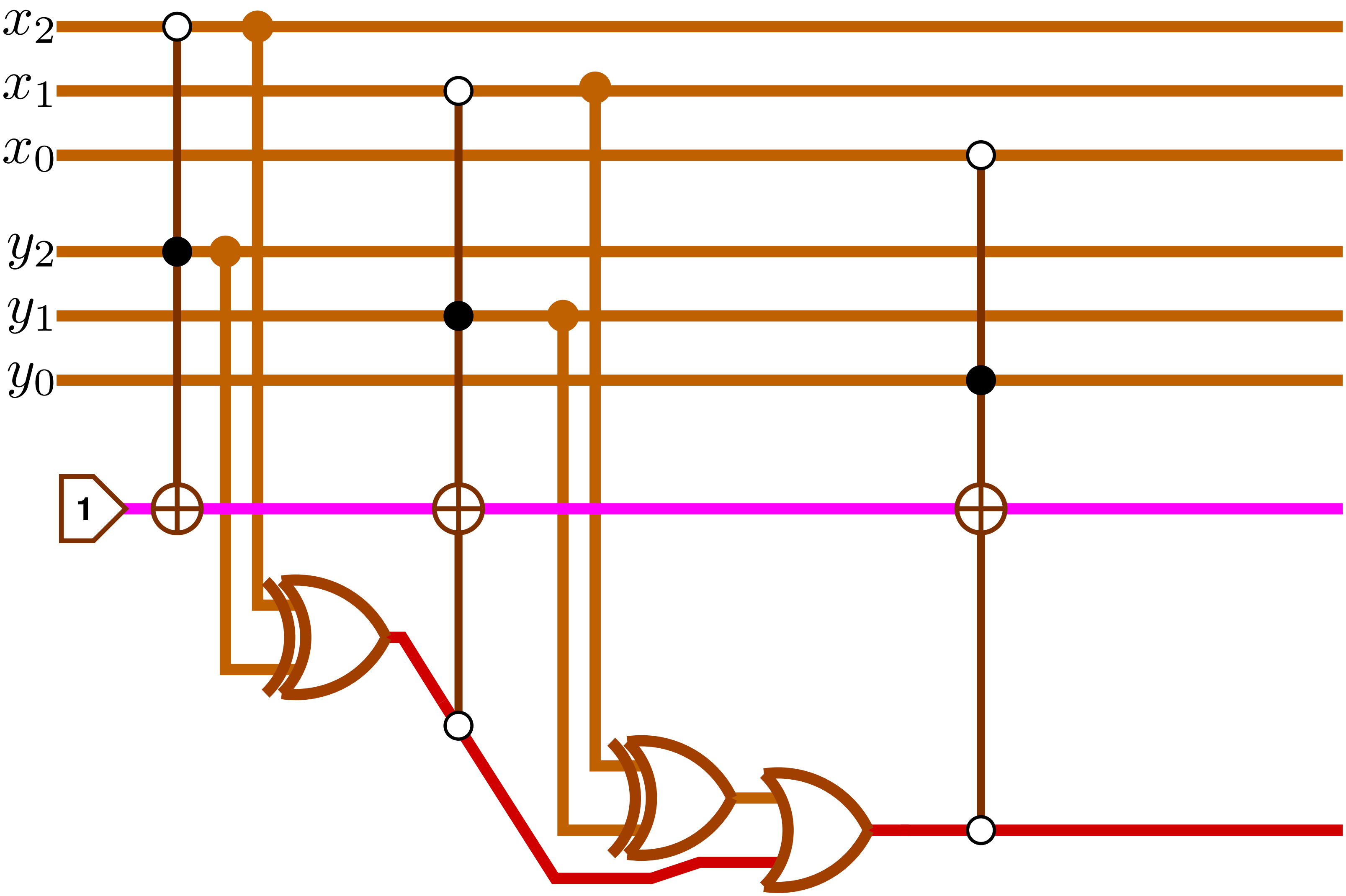
Reversifying the Comparison Circuit



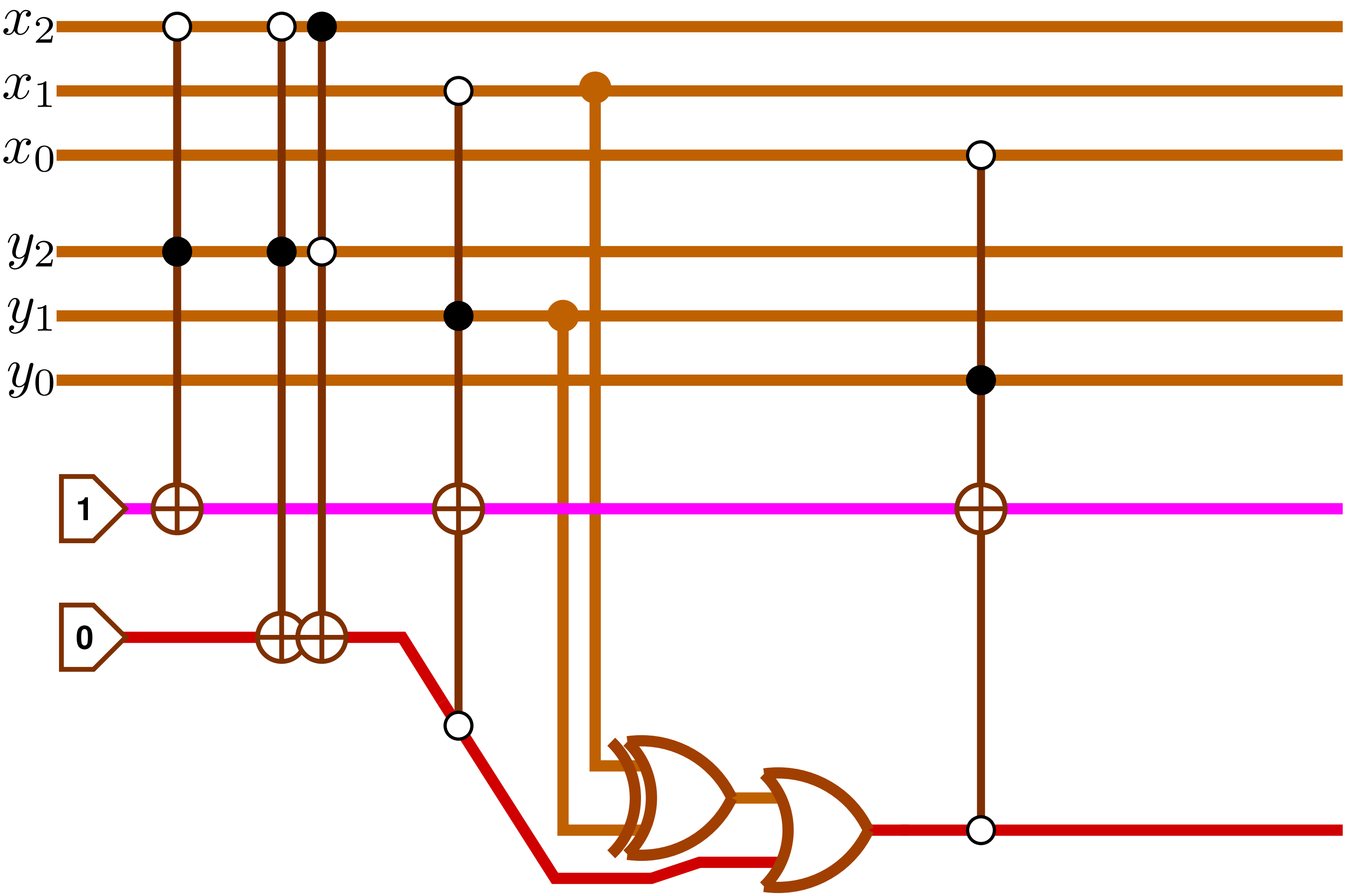
Reversifying the Comparison Circuit



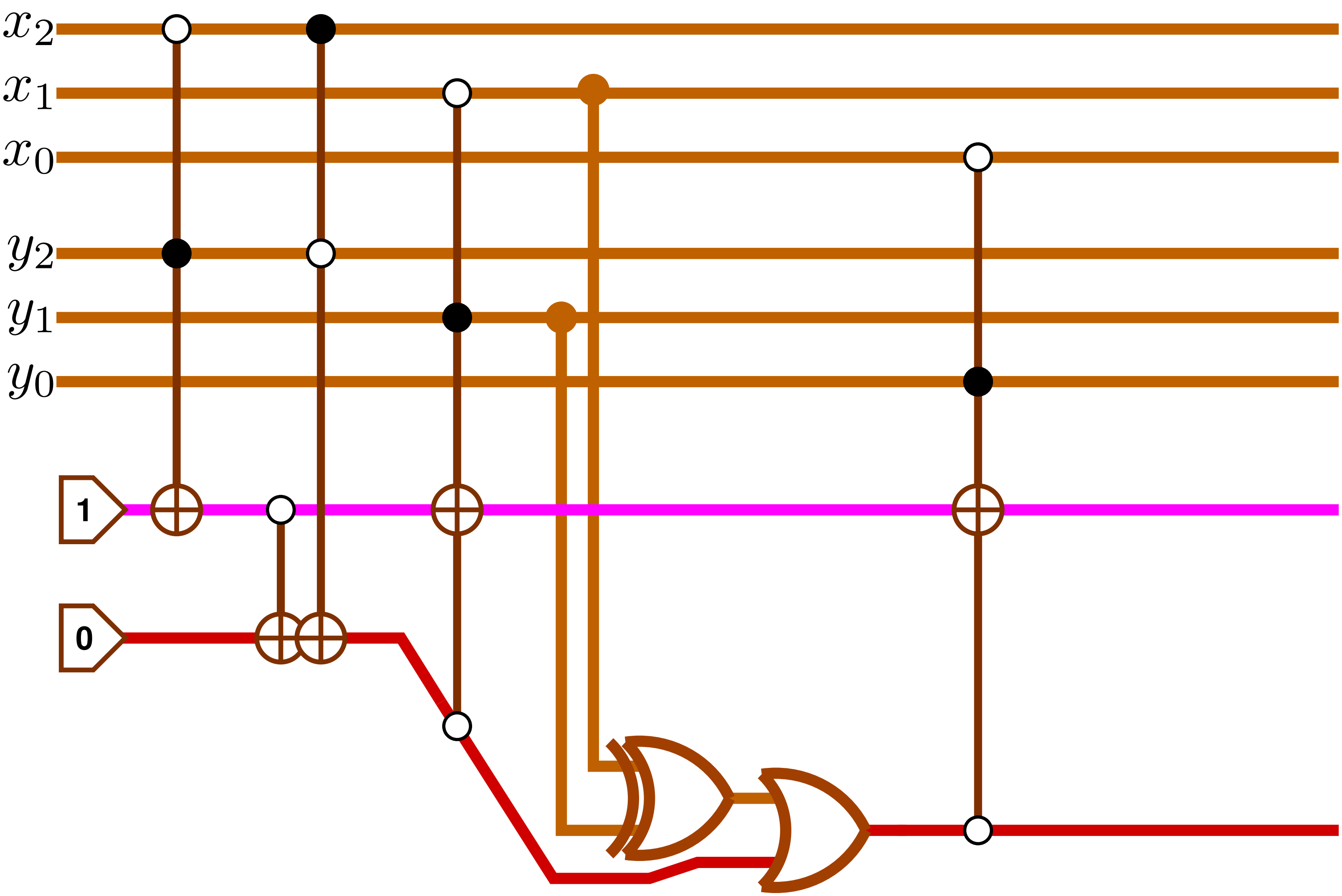
Reversifying the Comparison Circuit



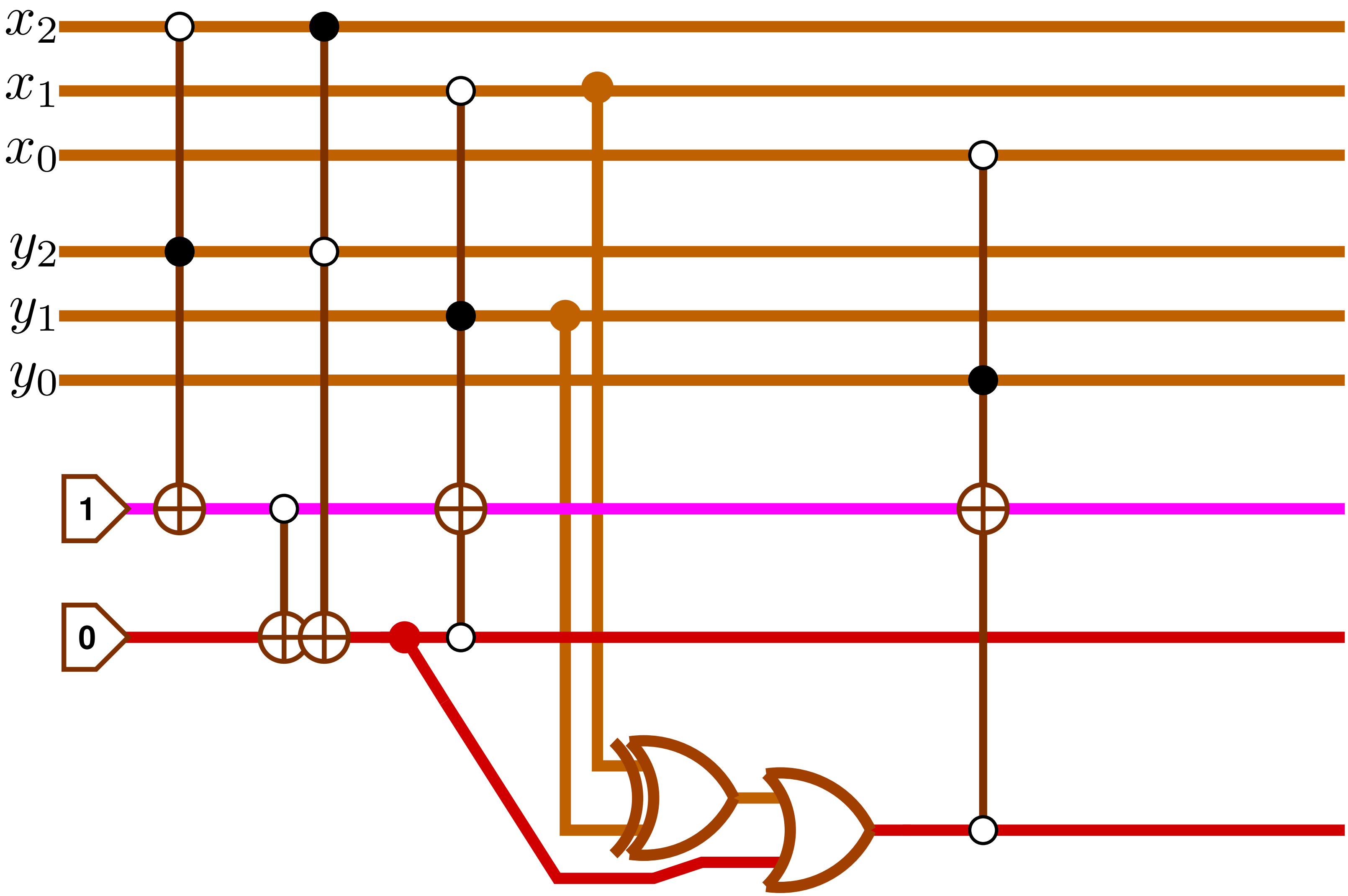
Reversifying the Comparison Circuit



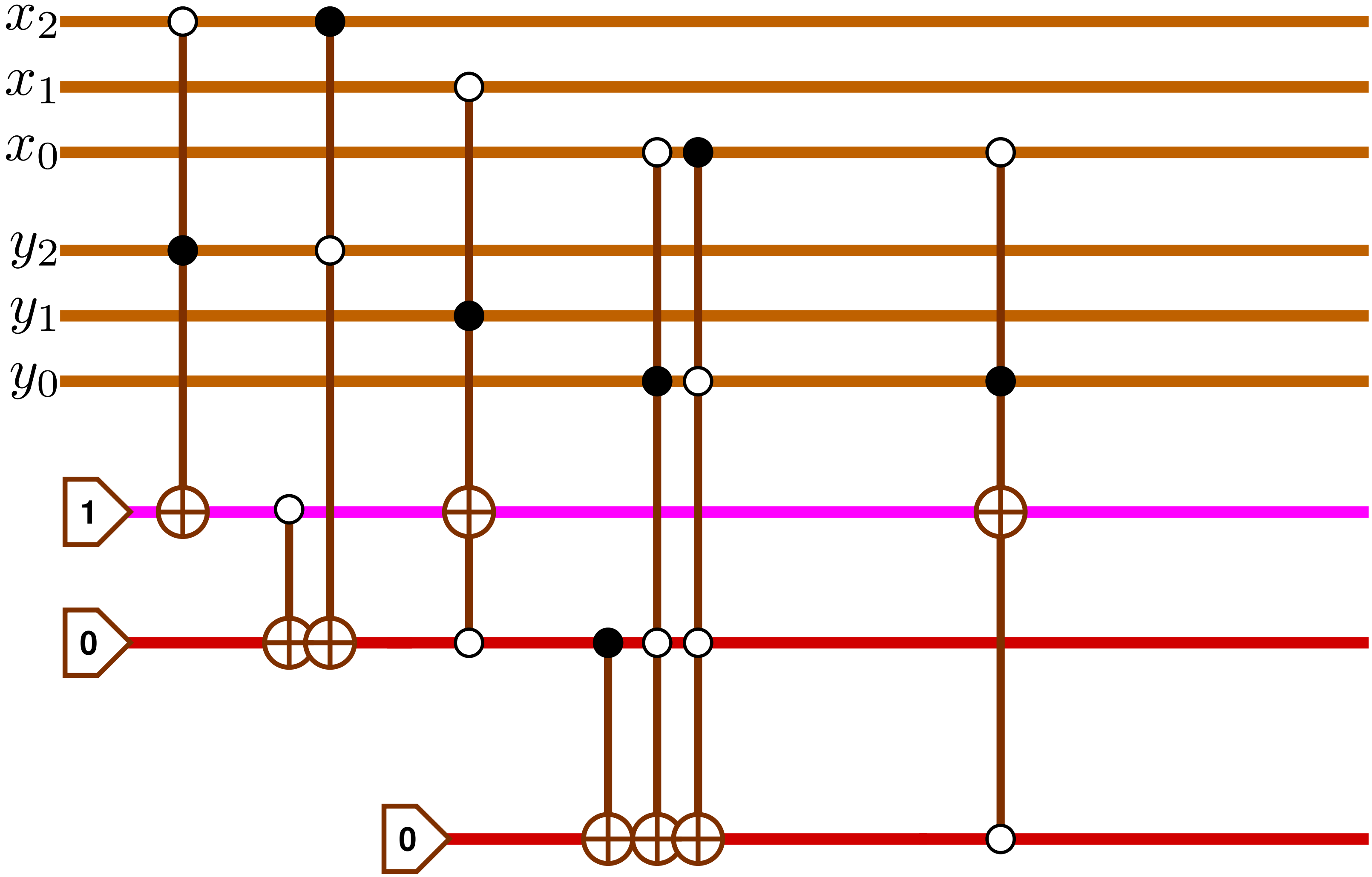
Reversifying the Comparison Circuit



Reversifying the Comparison Circuit

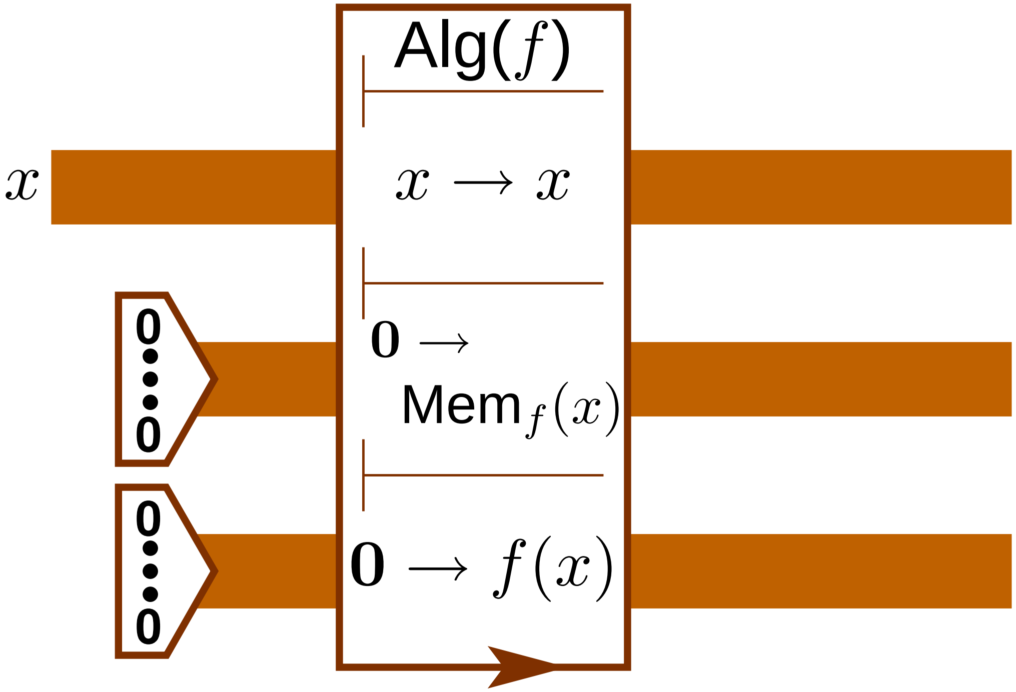


Reversifying the Comparison Circuit



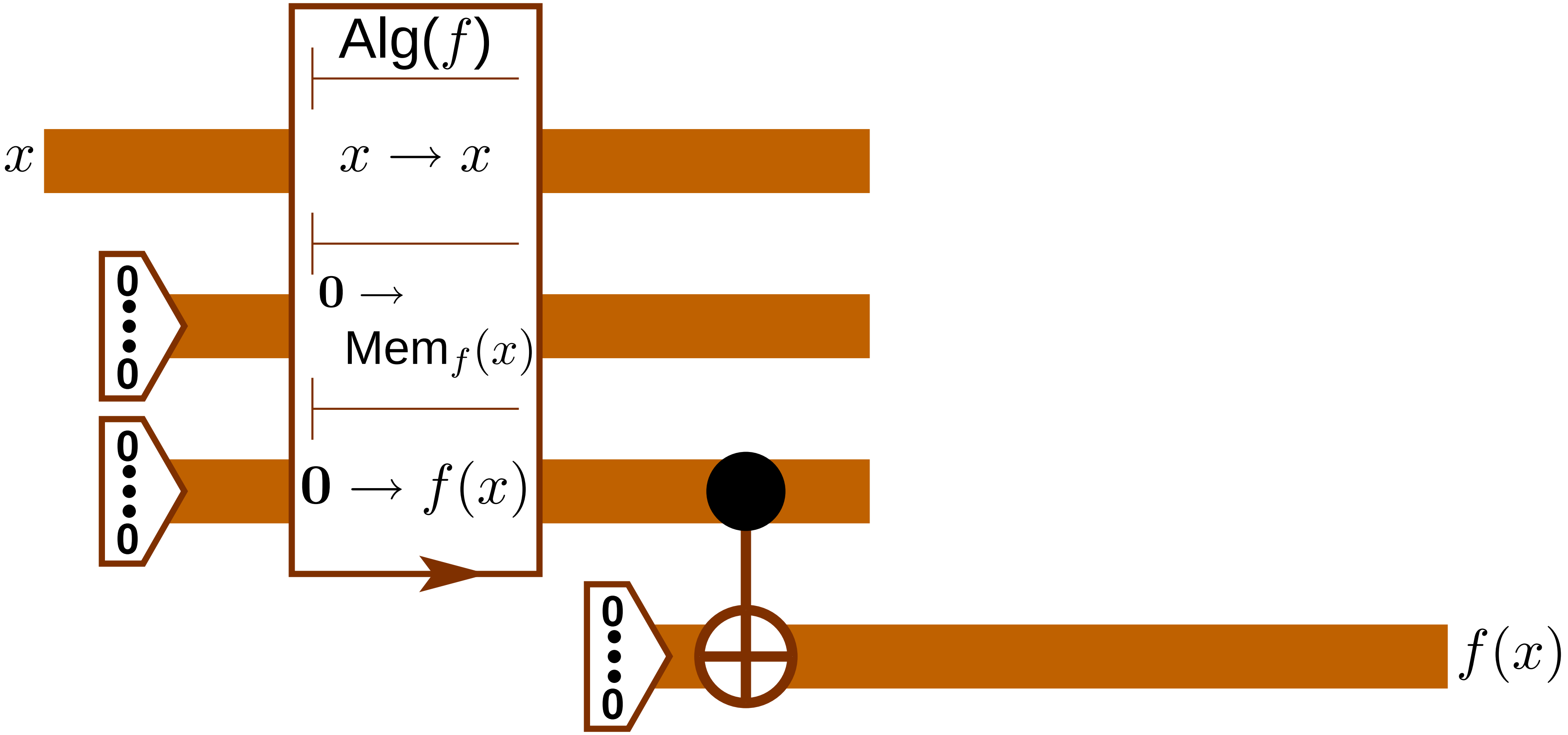
Erasing Memory

- Erasing memory by copying output and reversing.



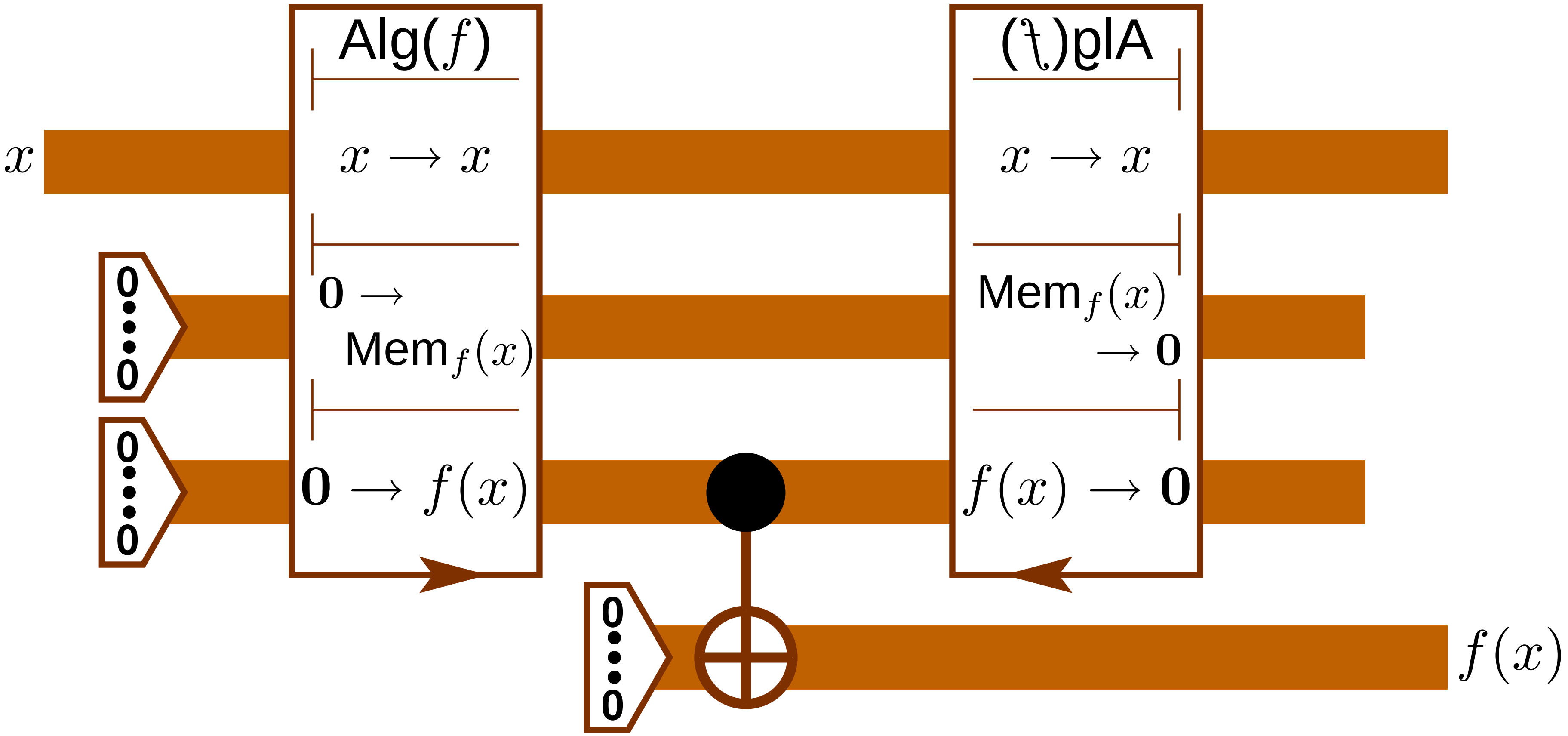
Erasing Memory

- Erasing memory by copying output and reversing.


Alg(f)
x
$x \rightarrow x$
0
⋮
0
$\mathbf{0} \rightarrow$ Mem$_f(x)$
0
⋮
0
$\mathbf{0} \rightarrow f(x)$
0
⋮
0
$f(x)$

Erasing Memory

- Erasing memory by copying output and reversing.



x

Alg(f)

$x \to x$

$\mathbf{0} \to$ Mem$_f(x)$

$\mathbf{0} \to f(x)$

Alg(f)

$x \to x$

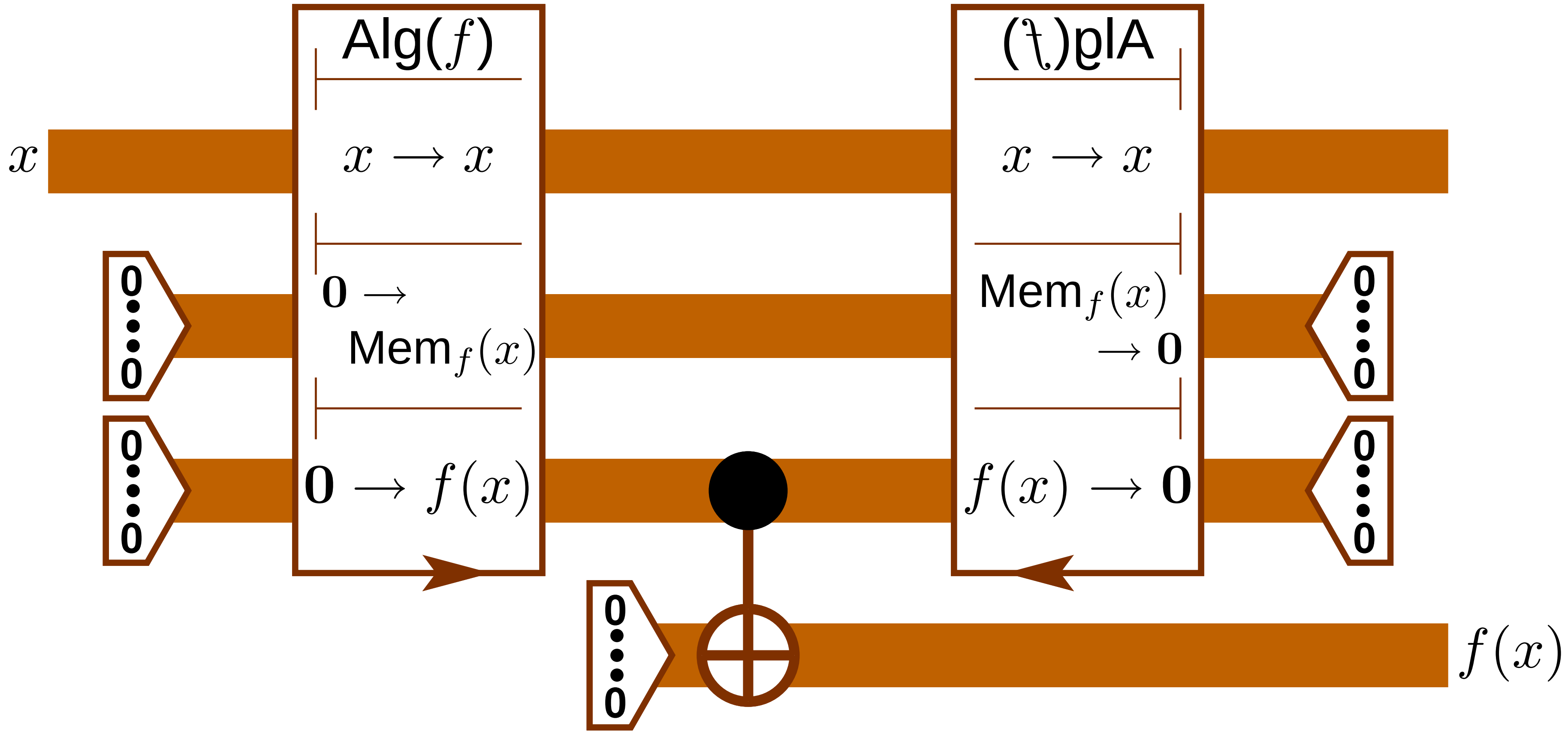
Mem$_f(x) \to \mathbf{0}$

$f(x) \to \mathbf{0}$

$f(x)$

Erasing Memory

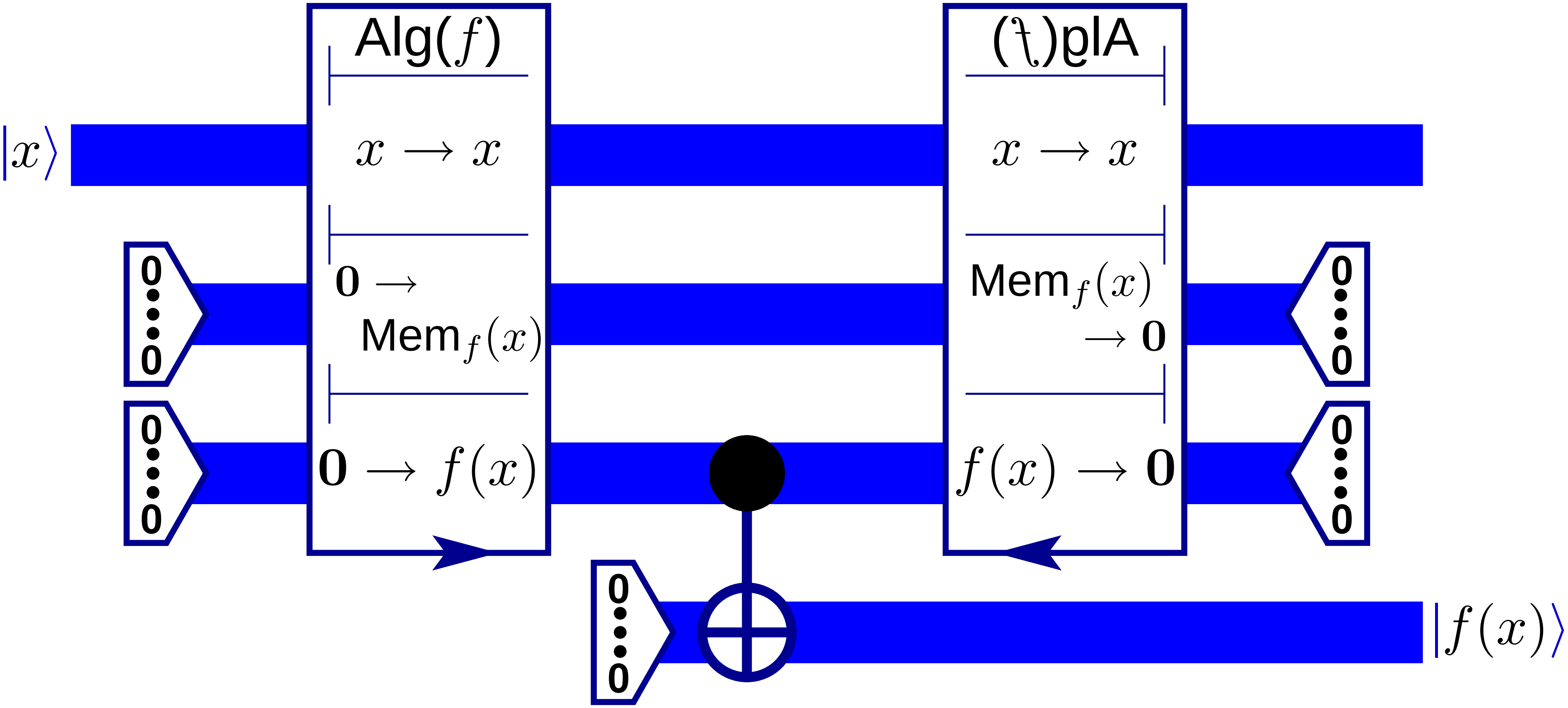
- Erasing memory by copying output and reversing.



Alg(f)
x
$x \rightarrow x$
$\mathbf{0} \rightarrow$ Mem$_f(x)$
$\mathbf{0} \rightarrow f(x)$
0 • • • • 0
0 • • • • 0
0 • • • • 0
(f)glA
$x \rightarrow x$
Mem$_f(x)$ $\rightarrow \mathbf{0}$
$f(x) \rightarrow \mathbf{0}$
0 • • • • 0
0 • • • • 0
$f(x)$

Erasing Memory

- Erasing memory by copying output and reversing.

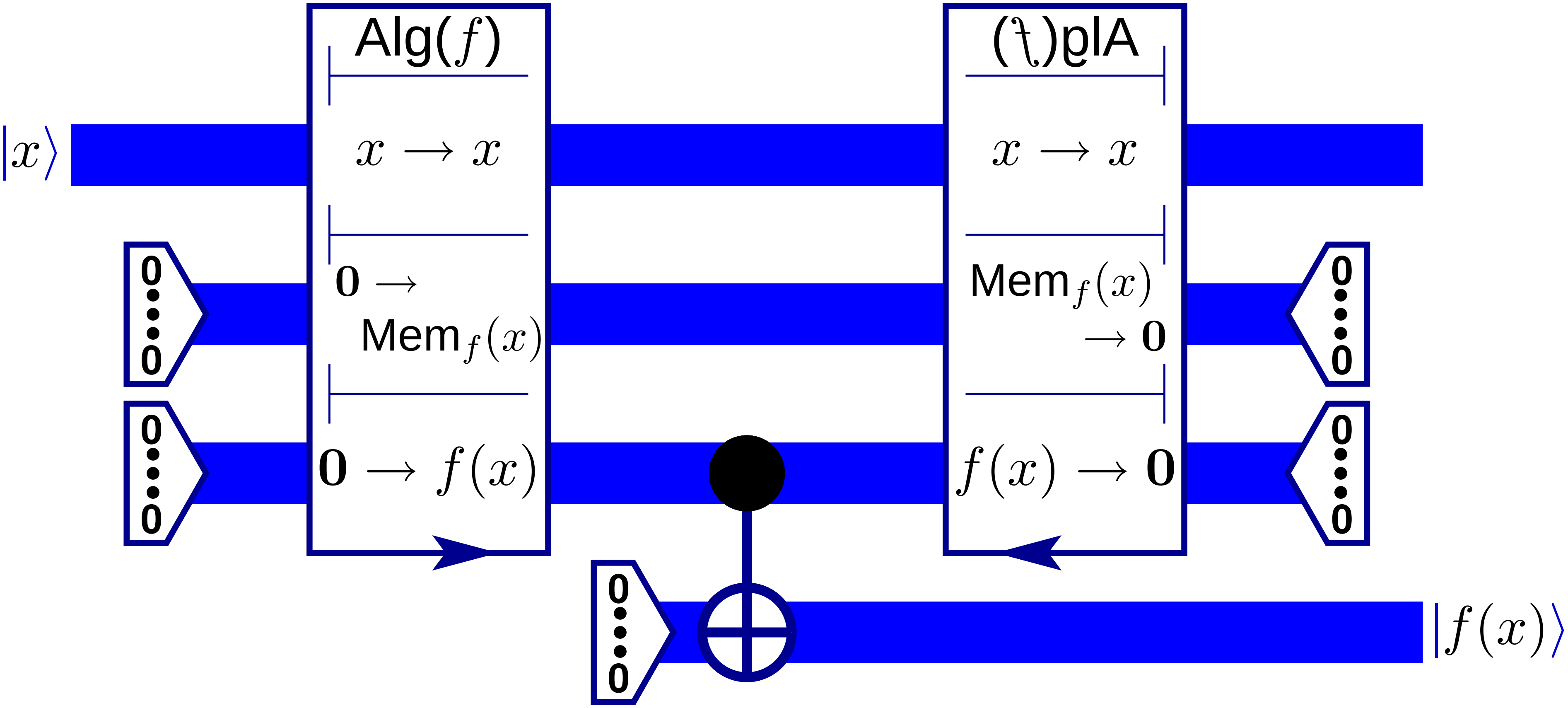


Alg(f)
(f)glA
$|x\rangle$
$x \rightarrow x$
$x \rightarrow x$
$\mathbf{0} \rightarrow$ Mem$_f(x)$
Mem$_f(x)$ $\rightarrow \mathbf{0}$
$\mathbf{0} \rightarrow f(x)$
$f(x) \rightarrow \mathbf{0}$
0 ⋮ 0
0 ⋮ 0
0 ⋮ 0
0 ⋮ 0
0 ⋮ 0
$|f(x)\rangle$

- Classical reversible gates → quantum gates.

Erasing Memory

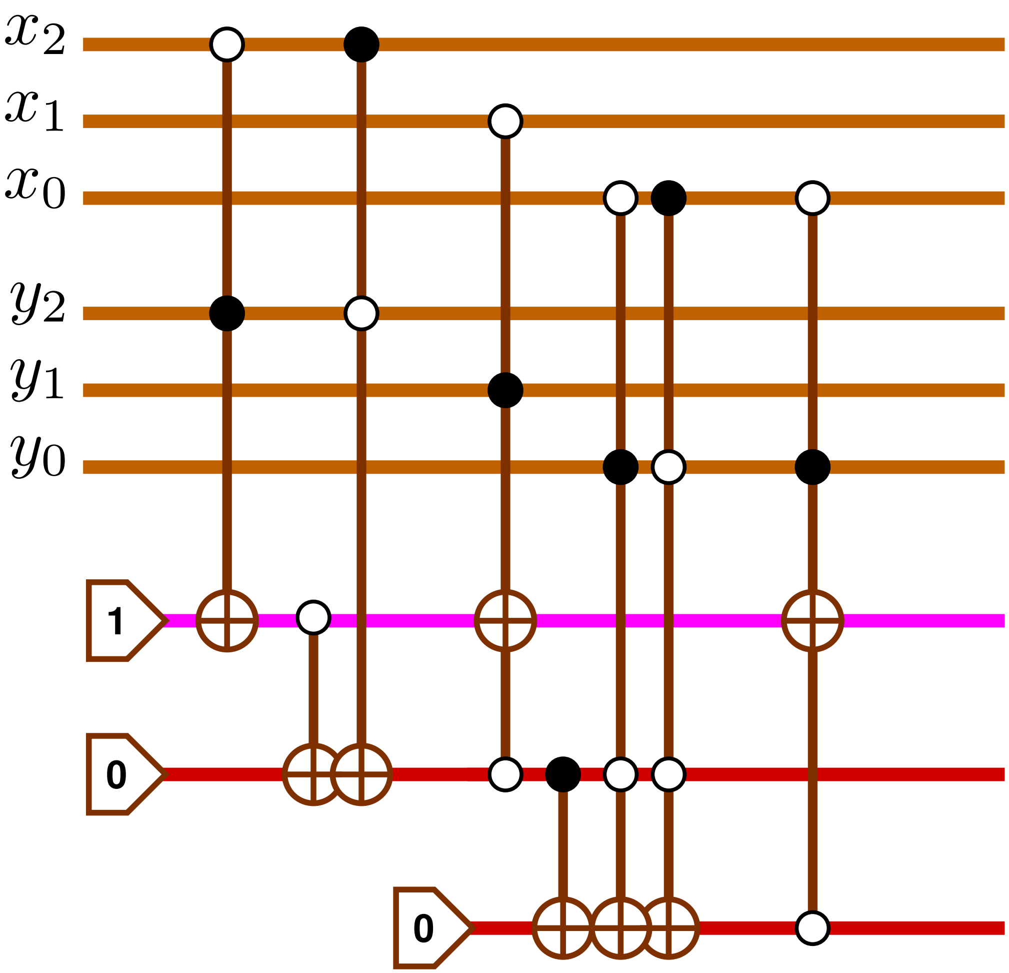
- Erasing memory by copying output and reversing.



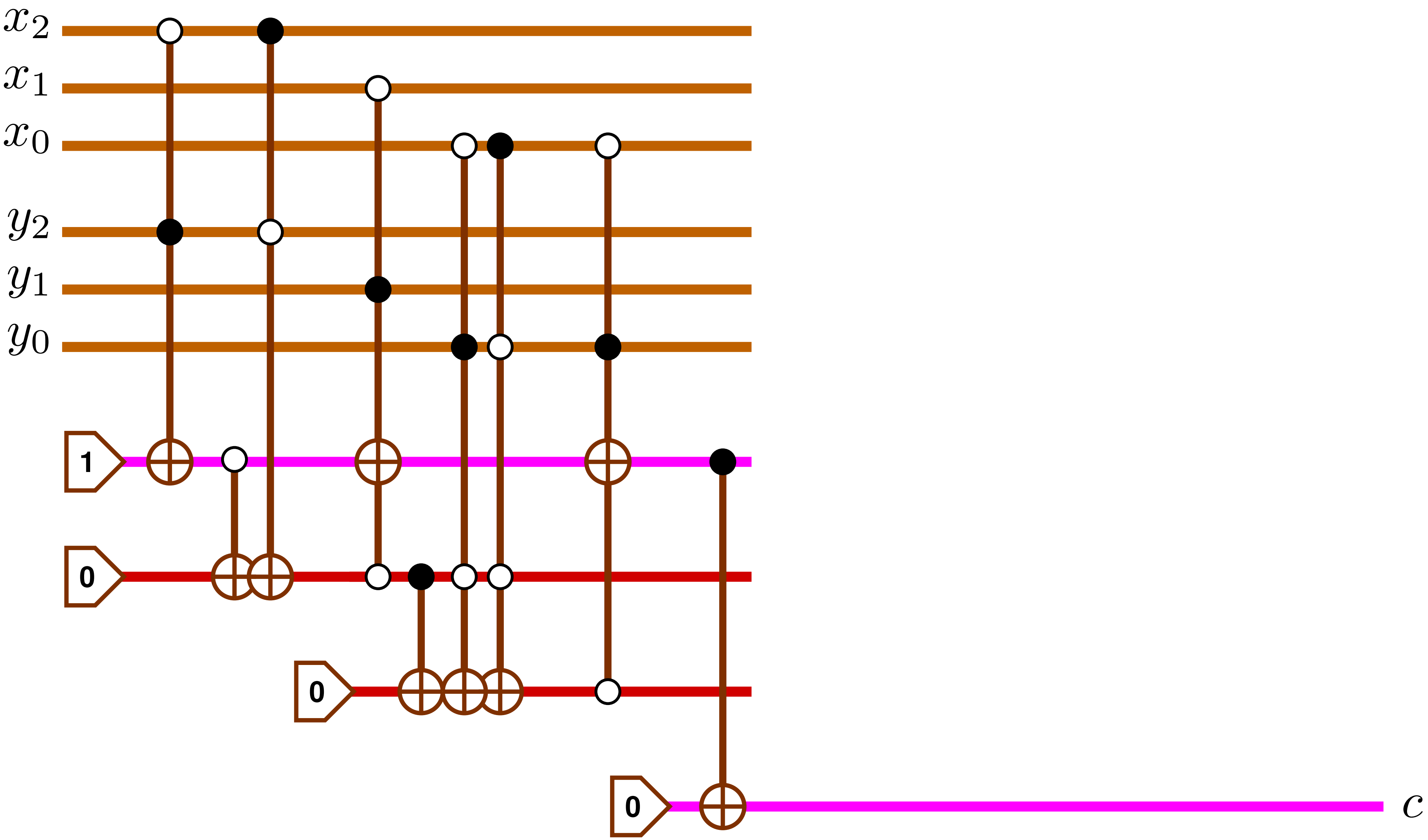
- Classical reversible gates → quantum gates.

$$\sum_x \alpha_x |x\rangle_{\mathsf{I}} \rightarrow \sum_x \alpha_x |x\rangle_{\mathsf{I}} |f(x)\rangle_{\mathsf{O}}$$

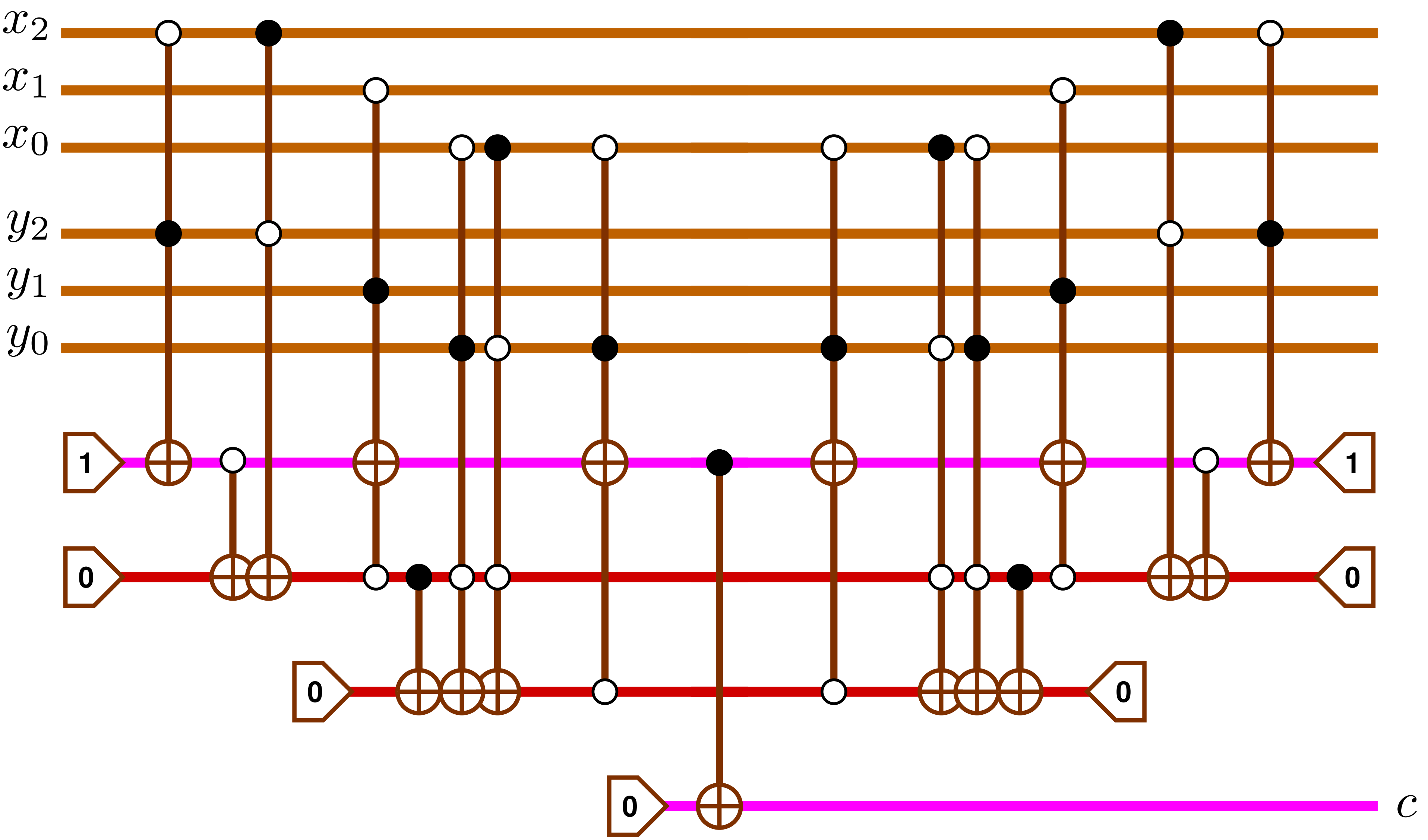
Coherent Comparison



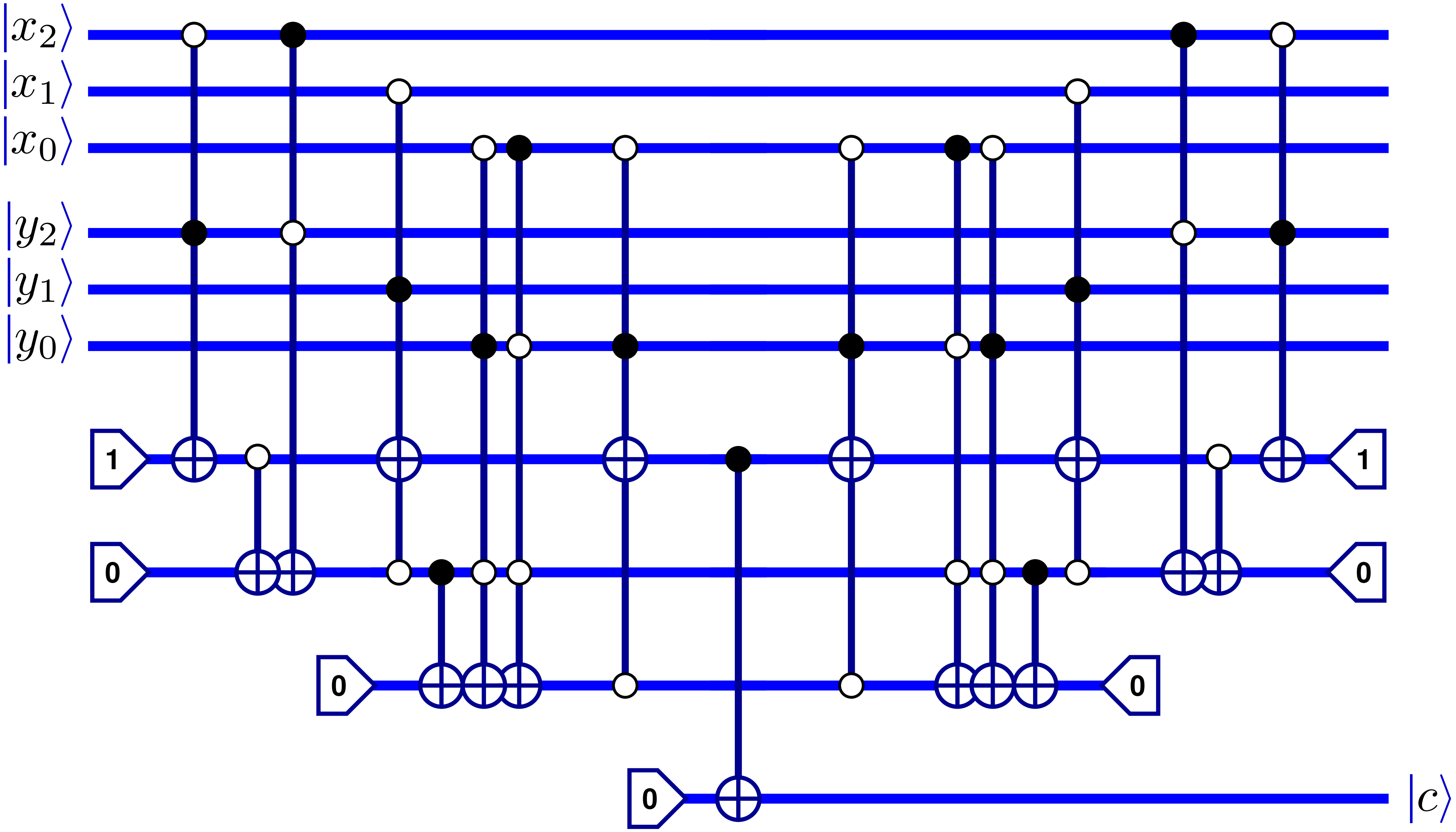
Coherent Comparison



Coherent Comparison



Coherent Comparison



Invertible Functions

- When can one coherently implement $\sum_x \alpha_x|x\rangle \to \sum_x \alpha_x|f(x)\rangle$?

Invertible Functions

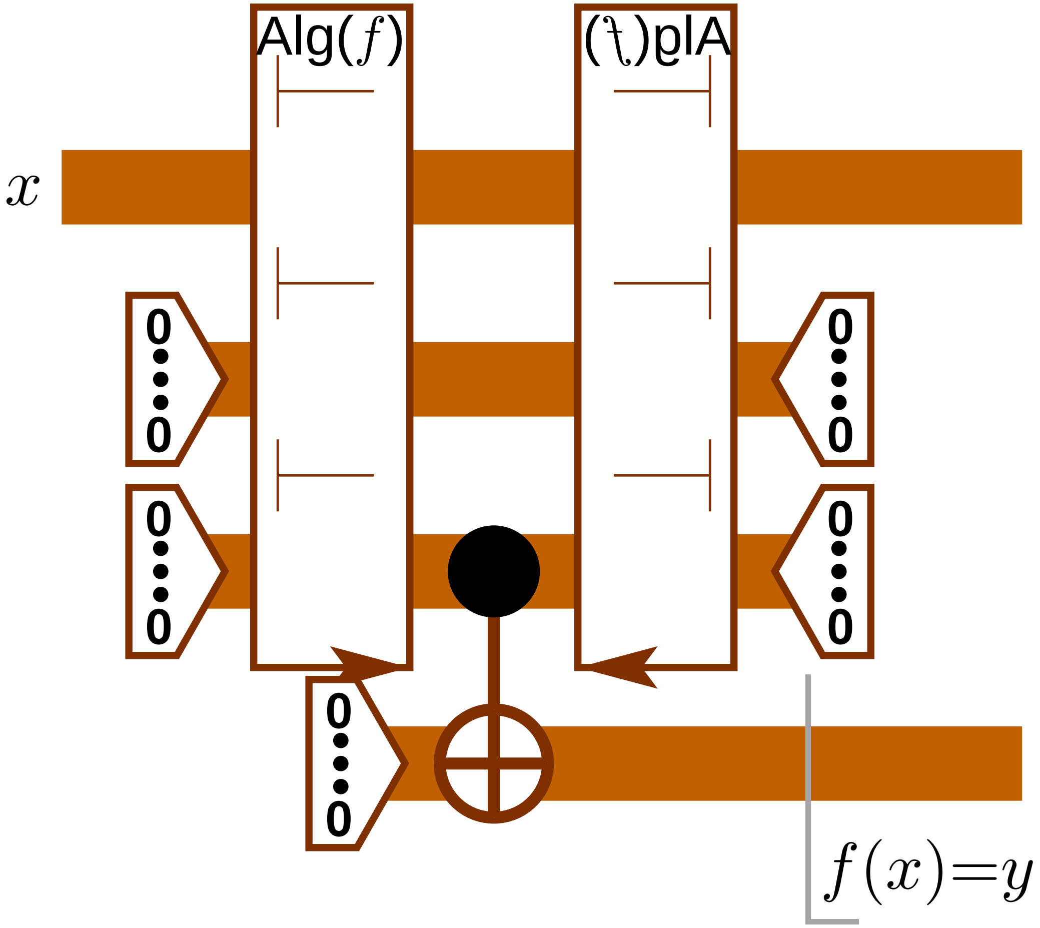
- When can one coherently implement $\sum_x \alpha_x|x\rangle \rightarrow \sum_x \alpha_x|f(x)\rangle$?
  - If there exists g such that $g(f(x)) = x$ (f is invertible).

Invertible Functions

- When can one coherently implement $\sum_x \alpha_x|x\rangle \rightarrow \sum_x \alpha_x|f(x)\rangle$?
  - If there exists g such that $g(f(x)) = x$ (f is invertible).
- Network implementation given algorithms for f and $g = f^{-1}$.

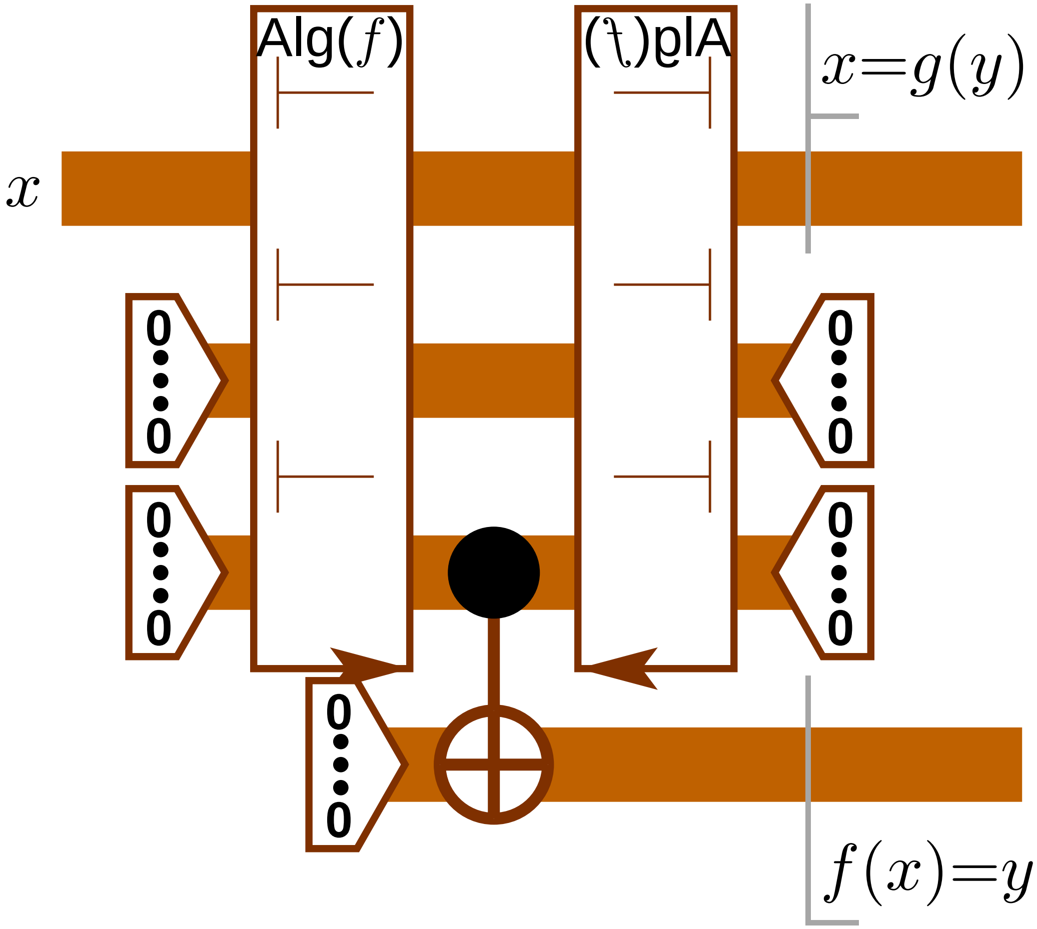
Invertible Functions

- When can one coherently implement $\sum_x \alpha_x|x\rangle \rightarrow \sum_x \alpha_x|f(x)\rangle$?
  - If there exists g such that $g(f(x)) = x$ (f is invertible).
- Network implementation given algorithms for f and $g = f^{-1}$.



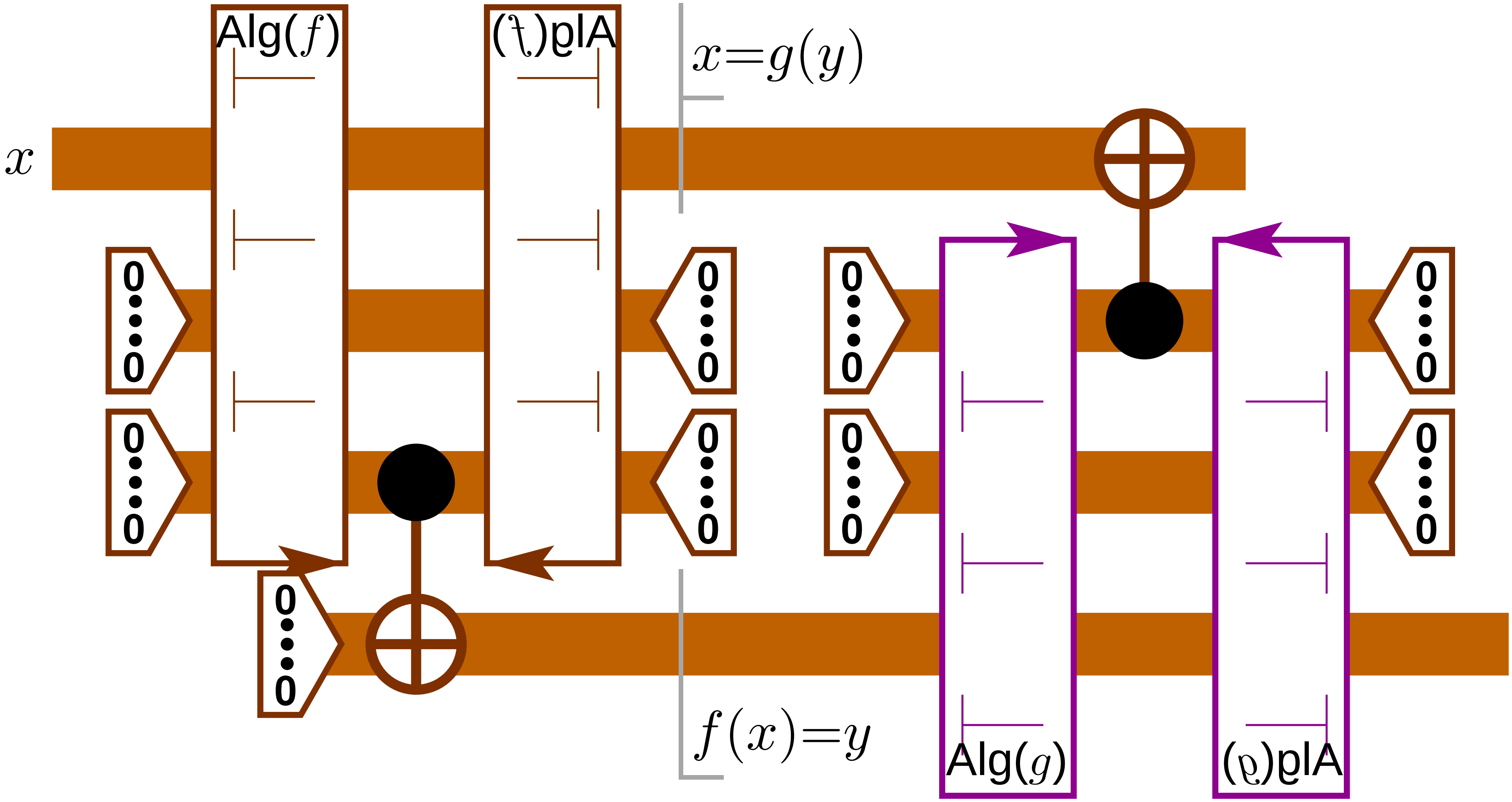
Invertible Functions

- When can one coherently implement $\sum_x \alpha_x|x\rangle \rightarrow \sum_x \alpha_x|f(x)\rangle$?
  - If there exists g such that $g(f(x)) = x$ (f is invertible).
- Network implementation given algorithms for f and $g = f^{-1}$.



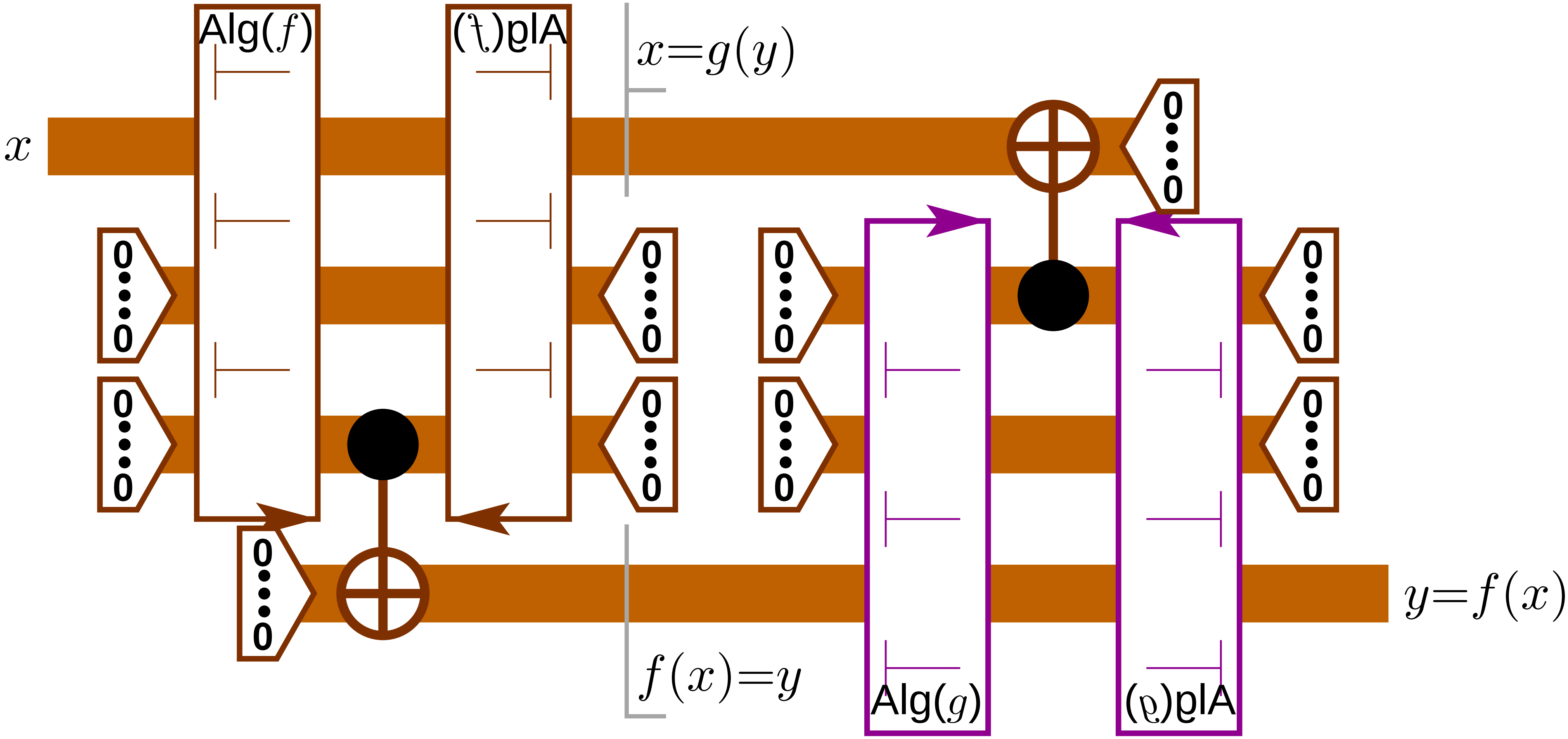
Invertible Functions

- When can one coherently implement $\sum_x \alpha_x|x\rangle \rightarrow \sum_x \alpha_x|f(x)\rangle$?
  - If there exists g such that $g(f(x)) = x$ (f is invertible).
- Network implementation given algorithms for f and $g = f^{-1}$.



Invertible Functions

- When can one coherently implement $\sum_x \alpha_x|x\rangle \rightarrow \sum_x \alpha_x|f(x)\rangle$?
  - If there exists g such that $g(f(x)) = x$ (f is invertible).
- Network implementation given algorithms for f and $g = f^{-1}$.



Contents

References

[1] C. H. Bennett, G. Brassard, S. Breidbart, and S. Wiesner. Quantum cryptography, or unforgeable subway tokens. In *Advances in Cryptology: Proceedings of Crypto'82*, pages 267–275. Plenum Press, 1982.

[2] C. H. Bennett. Time/space trade-offs for reversible computation. *SIAM J. Comput.*, 18:766–776, 1989.