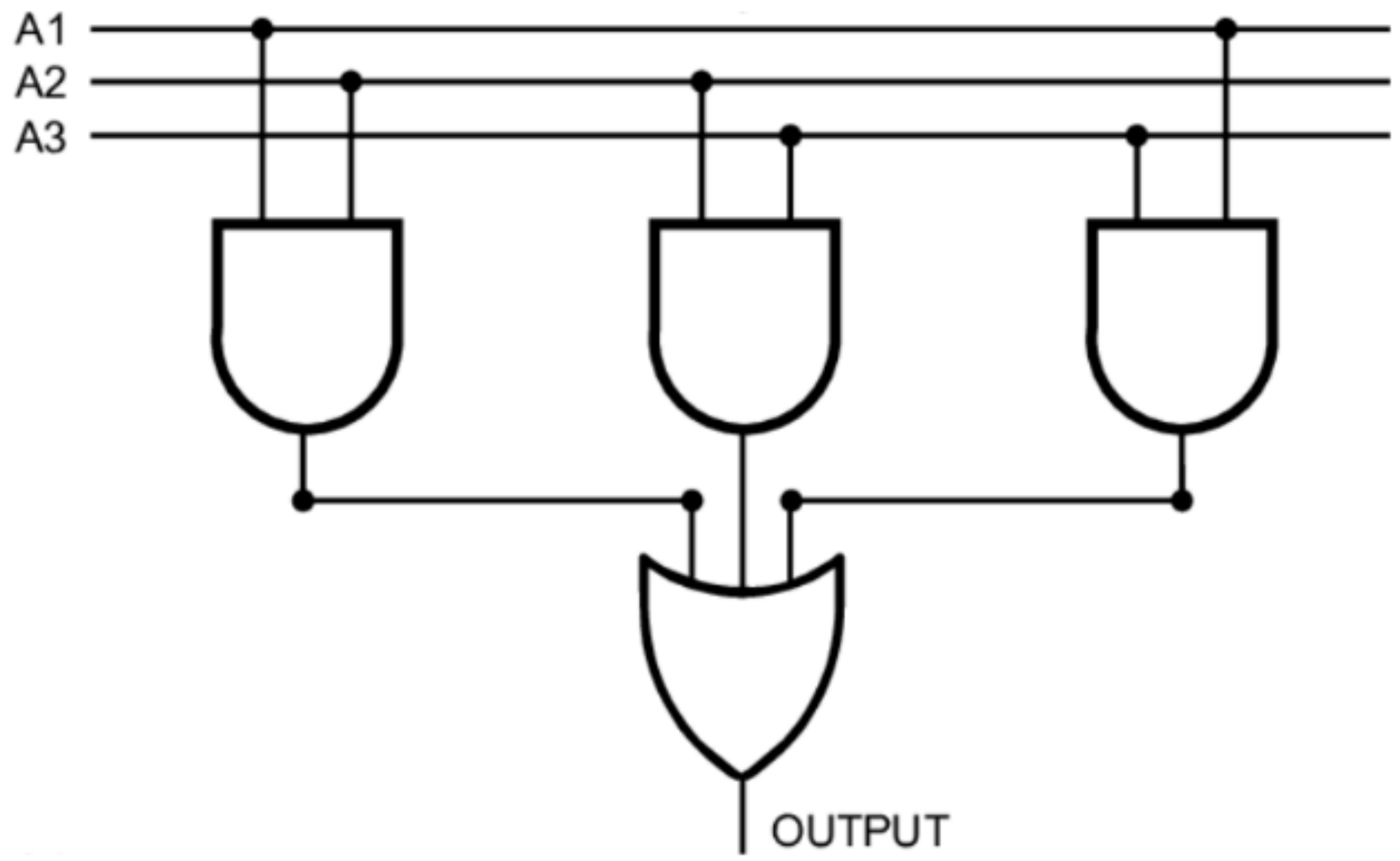
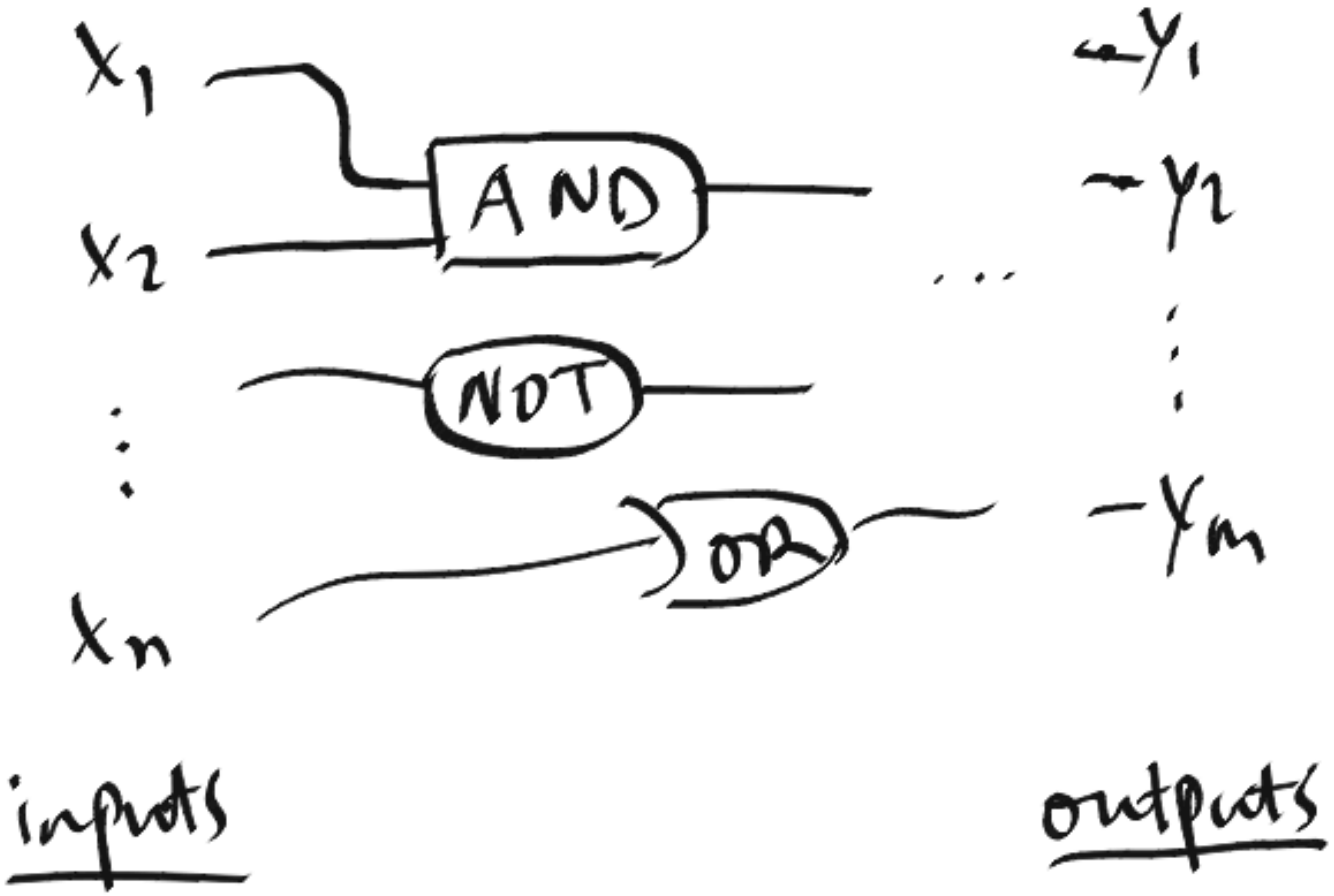
Review of classical circuits

- Computes a Boolean function: $F: \{0,1\}^n \rightarrow \{0,1\}^m$
- Complexity / efficiency = number of gates and how it scales with n
- EG:





Various computational models

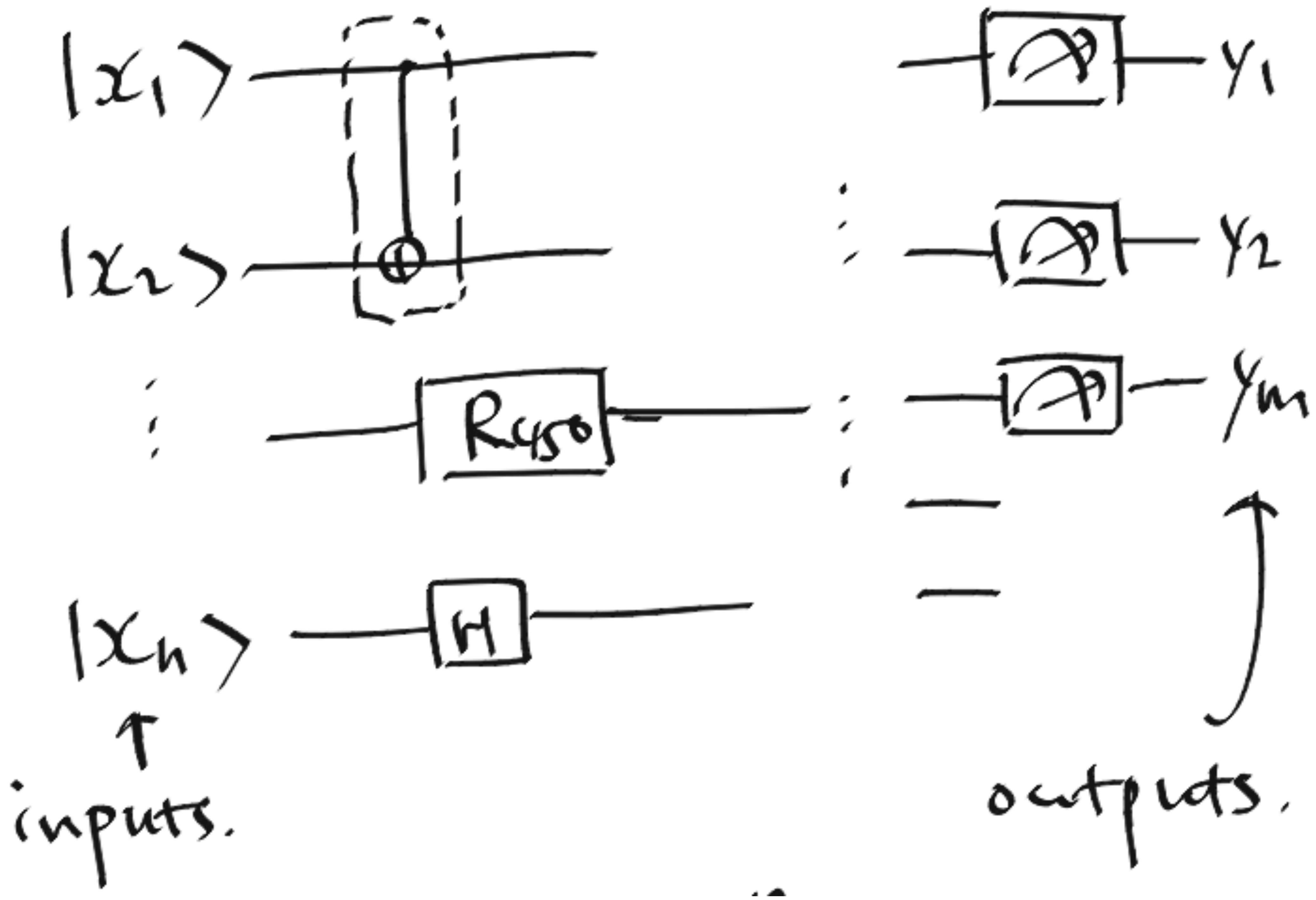
- Turing machine
- Programming language of your choice
- Complexity: time / steps
- Fact: Given C++ code computing $F: \{0,1\}^n \to \{0,1\}^m$ in $T(n)$ steps, can produce circuit computing F with $O(T(n)\log T(n)) = \tilde{O}(T(n))$ gates

Probabilistic circuits

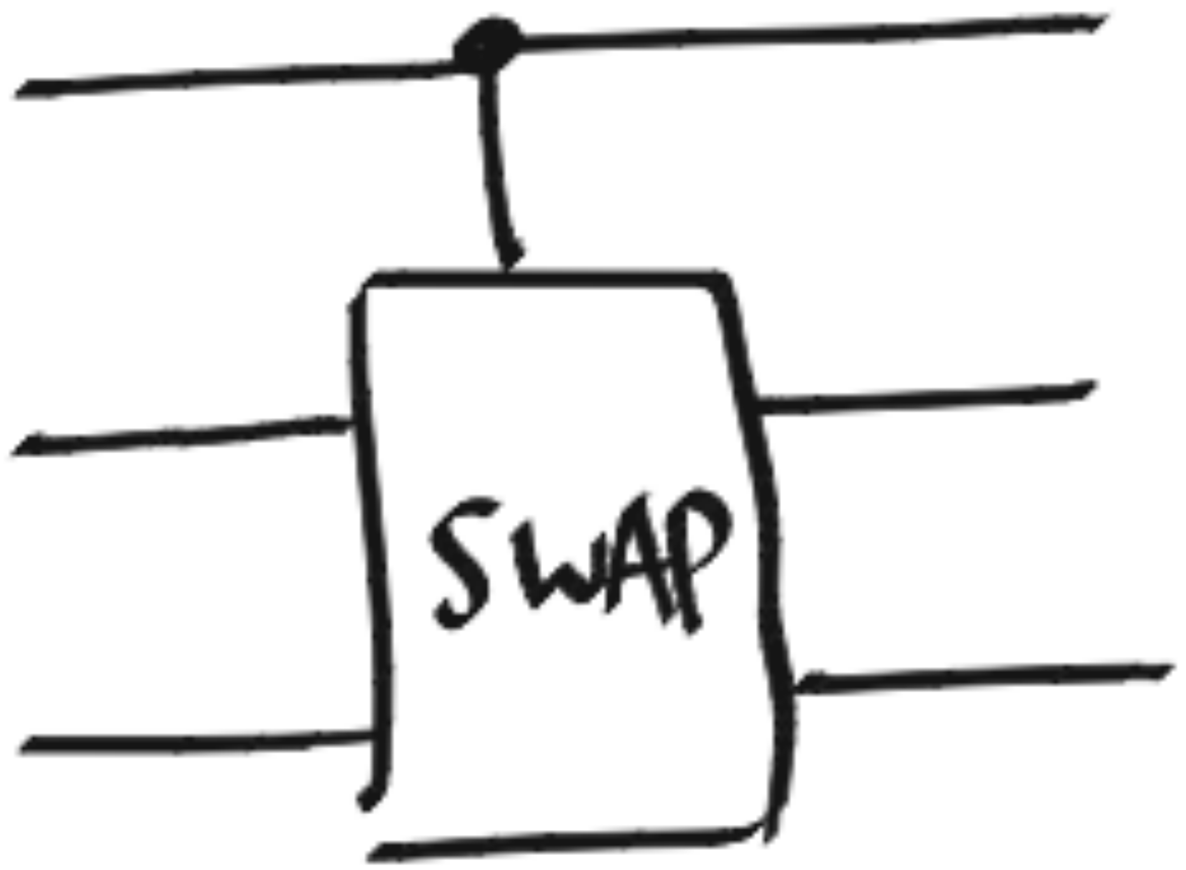
- Adds a [COIN FLIP] gate, which has no input, and outputs 0 or 1 equally likely
- Probabilistic circuit C "computes" $F: \{0,1\}^n \to \{0,1\}^m$ if for all $x \in \{0,1\}^n$, the probability that $C(x) \neq F(x)$ is small (say at most 1%).

Quantum circuits

- Input $x \in (x_1, \ldots, x_n) \in \{0,1\}^n$ (for now)
- Remark: There seems to exist a Boolean function which quantum circuits compute more efficiently than probabilistic circuits (Shor's factorization algorithm)



- Question: Is there a Boolean function which quantum circuits can compute much more efficiently than quantum ones?
- Simpler question: Can a quantum circuit even compute the AND gate?
- Recall: quantum gates are unitary: $U^{-1} = U^\dagger$, hence invertible / reversible
- AND gate is not reversible: 00, 01, 10 all get mapped to 0
- NOT gate is reversible though
- The controlled swap gate
  - Use CSWAP to simulate AND, OR, NOT
  - Also FANOUT gate



Conclusion

- Theorem: Any classical circuit C computing $F: \{0,1\}^n \to \{0,1\}^m$ can be efficiently converted to a reversible quantum circuit $QC: \{0,1\}^{n+a} \to \{0,1\}^{m+g}$, where a is the number of ancillas, and g is the number of garbage qubits.

- What about [COIN FLIP] gate?

- Remark: "Principle of deferred measurement" can move measurements to the end of the quantum circuits

- Conclusion: Quantum computation $\geq$ classical / probabilistic computing

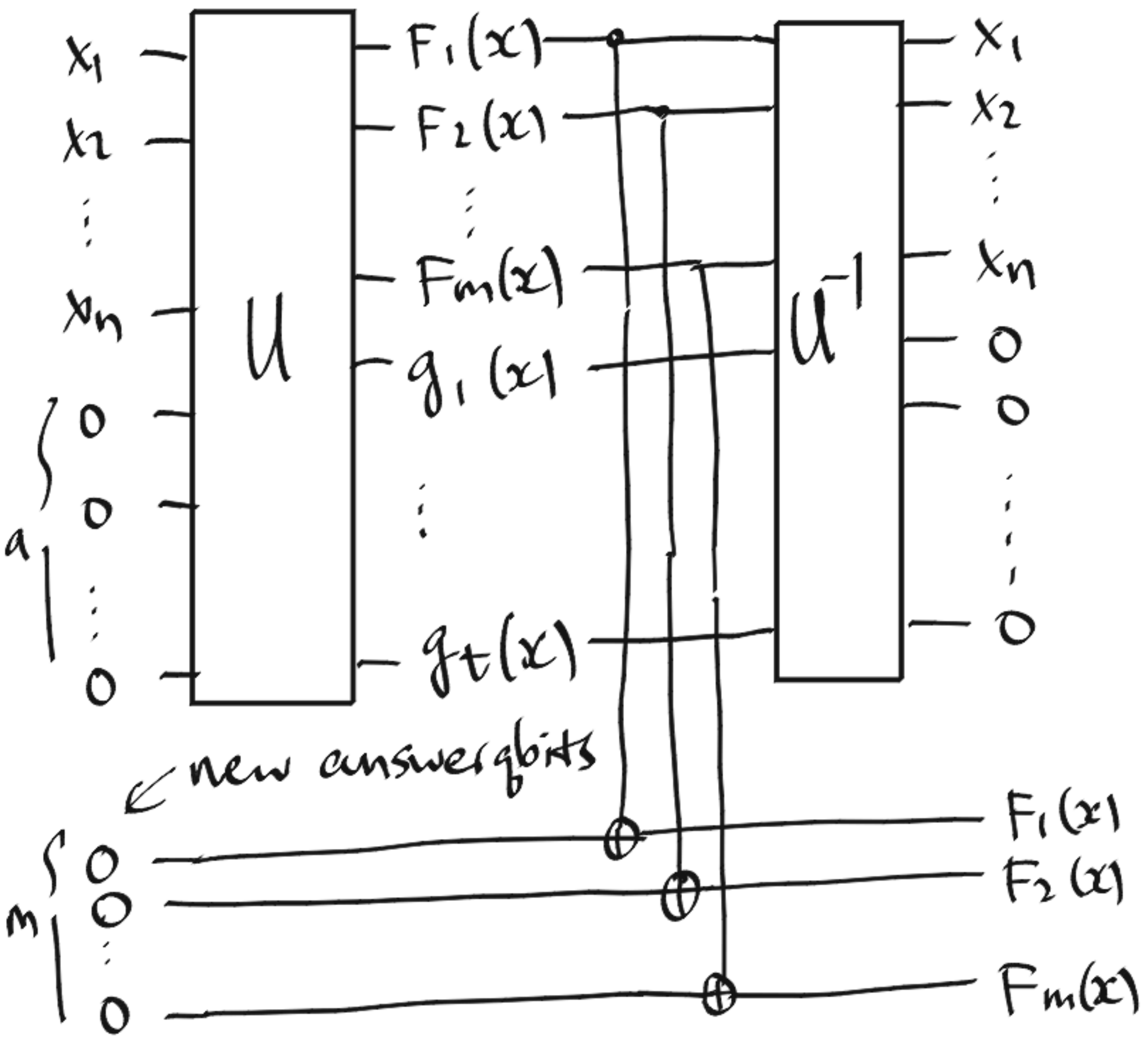
Paradox

- There exists an efficient classical multiplication circuits $(a, b) \mapsto ab$
- Build the reversible version C
- Then reverse it to get an efficient classical circuit for factoring!

Un-computing garbage

[Bennett 80s]

- If the answer qubits are initialized to $|b\rangle$, where $b \in \{0,1\}^m$, then CNOT converts it to $|b \oplus F(x)\rangle$, where $\oplus$ is the bitwise XOR.



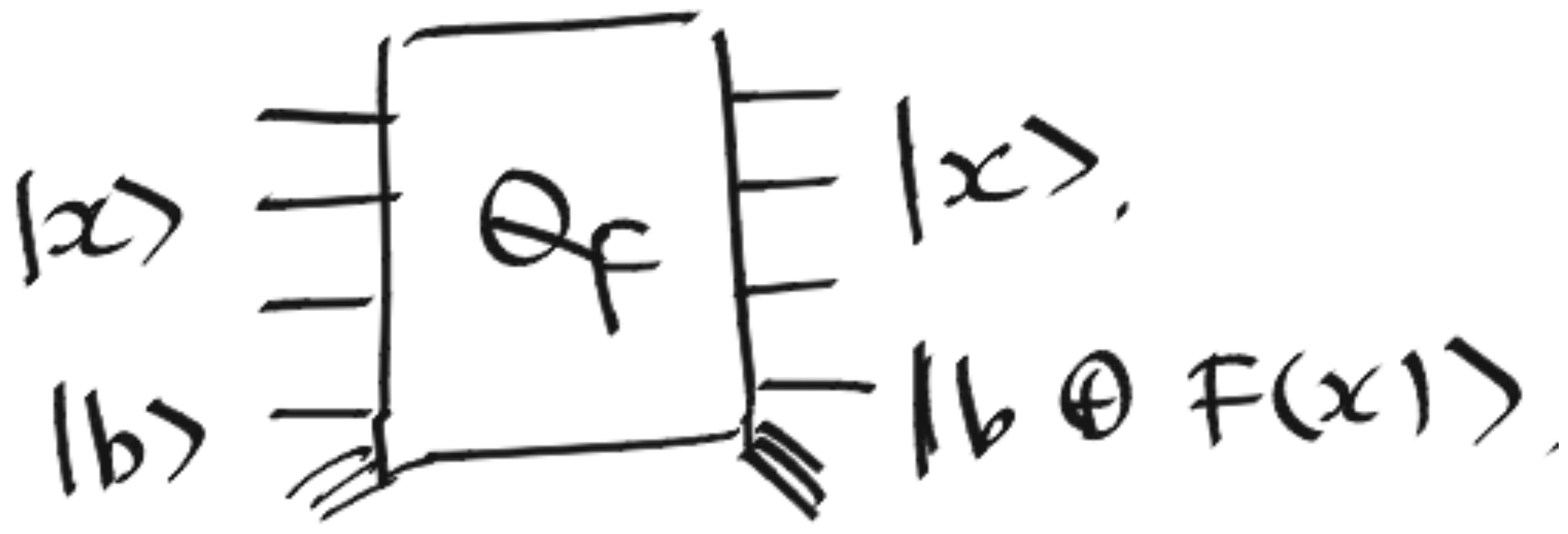
$$|x\rangle \otimes |0\rangle^{\otimes m} \otimes |0\rangle^{\otimes a}$$

$$\longmapsto |x\rangle \otimes |F(x)\rangle \otimes |0\rangle^{\otimes a}$$

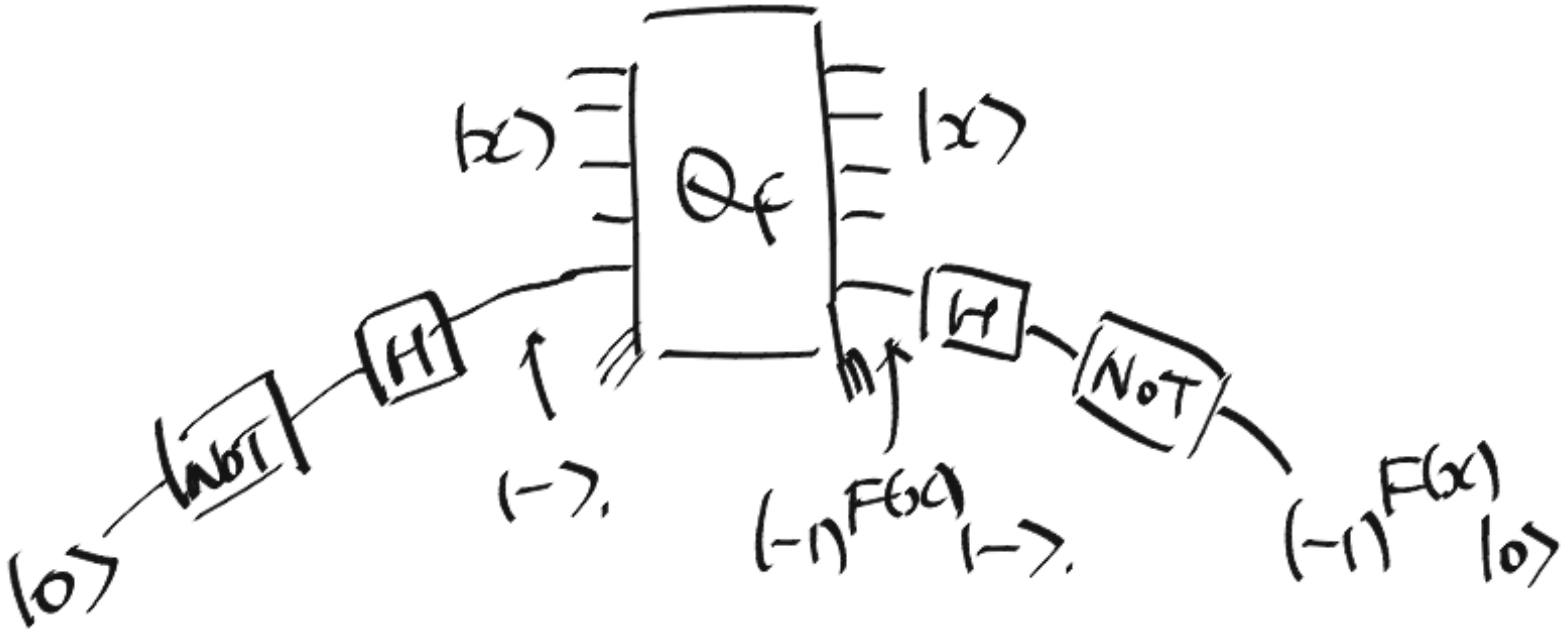
- Overall transformation $|x\rangle|b\rangle|0\rangle^a \mapsto |x\rangle|b \oplus F(x)\rangle|0\rangle^a$
    - Often $\otimes$ is omitted
    - $x \in \{0,1\}^n$ (inputs)
    - $b \in \{0,1\}^m$ (for outputs)
    - $|0\rangle^a$ ancillas
- Definition: A quantum circuit implements $F\colon \{0,1\}^n \to \{0,1\}^m$ if it computes it in the above garbage-free manner.

Special case m = 1

Boolean function $F\colon \{0,1\}^n \to \{0,1\}$



- What happens when $|b\rangle = |-\rangle$

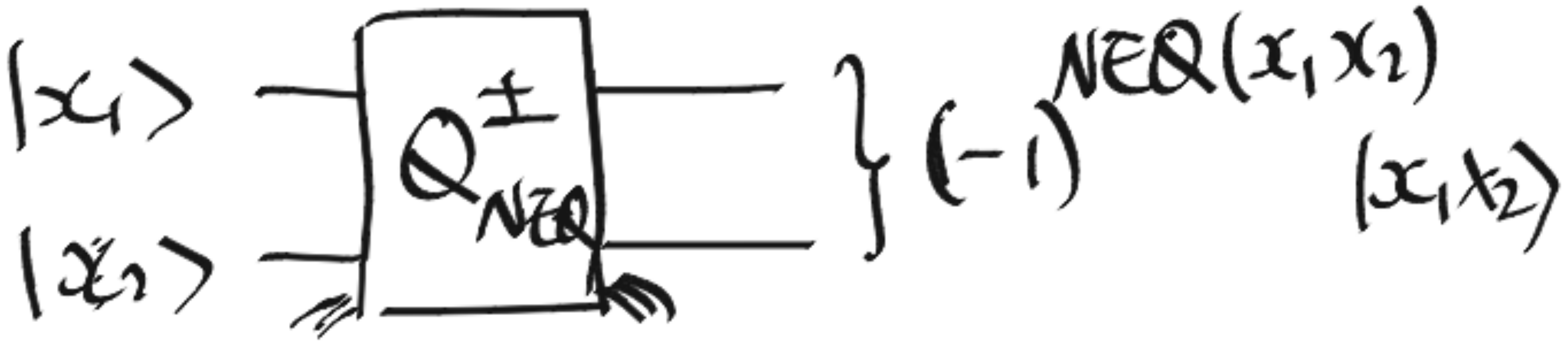


- Definition: $Q_F^{\pm}$ *sign-implements* $F\colon \{0,1\}^n \to \{0,1\}$ if
$$|x\rangle|0\rangle^a \mapsto (-1)^{F(x)}|x\rangle|0\rangle^a$$

Example NEQ

{0,1}^2 -> {0,1}

- Classical circuit C: (x1 AND ¬x2) OR (¬x1 OR x2)
- Sign implementation



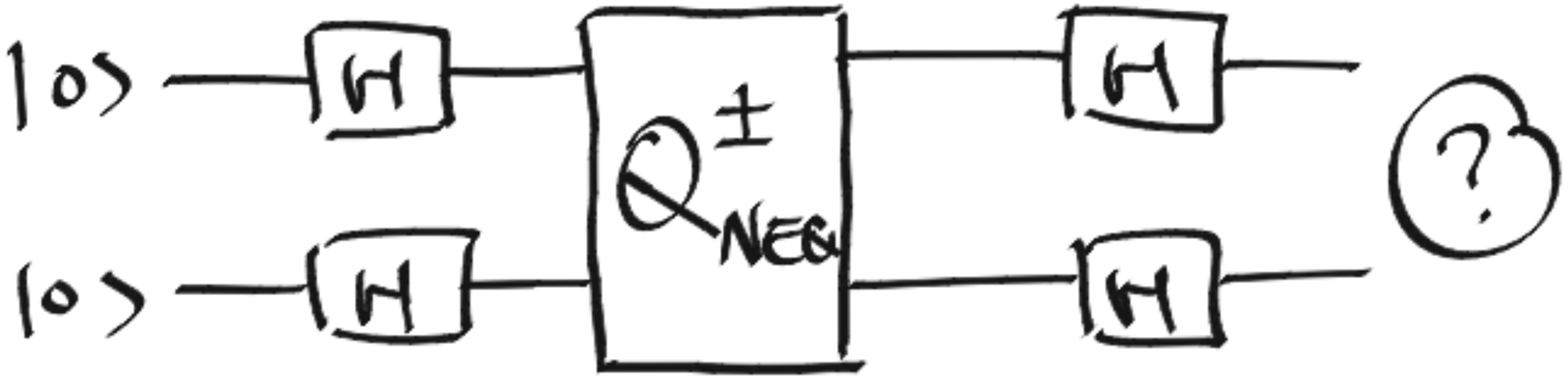
- How does it transform the standard basis?

Let's plug in superpositions!

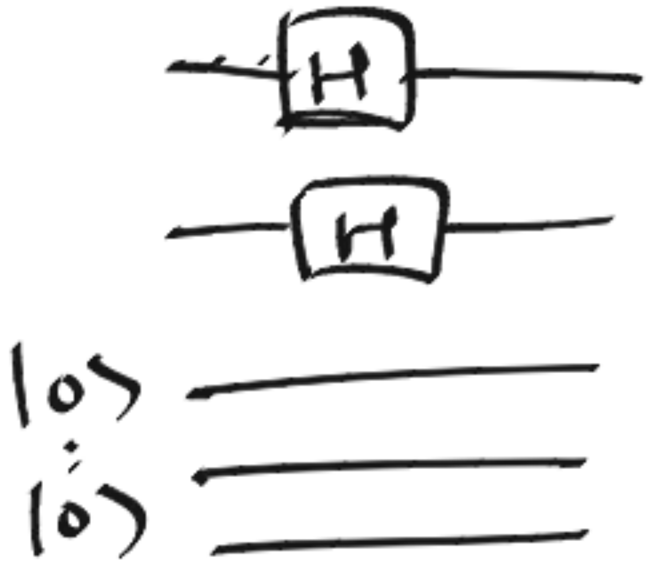
What happens if we plug in …

- $\frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$?
- How do we prepare $\frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$?
- Hadamard transform or Boolean Fourier transform.
- Measure now?

Rotate, compute, rotate

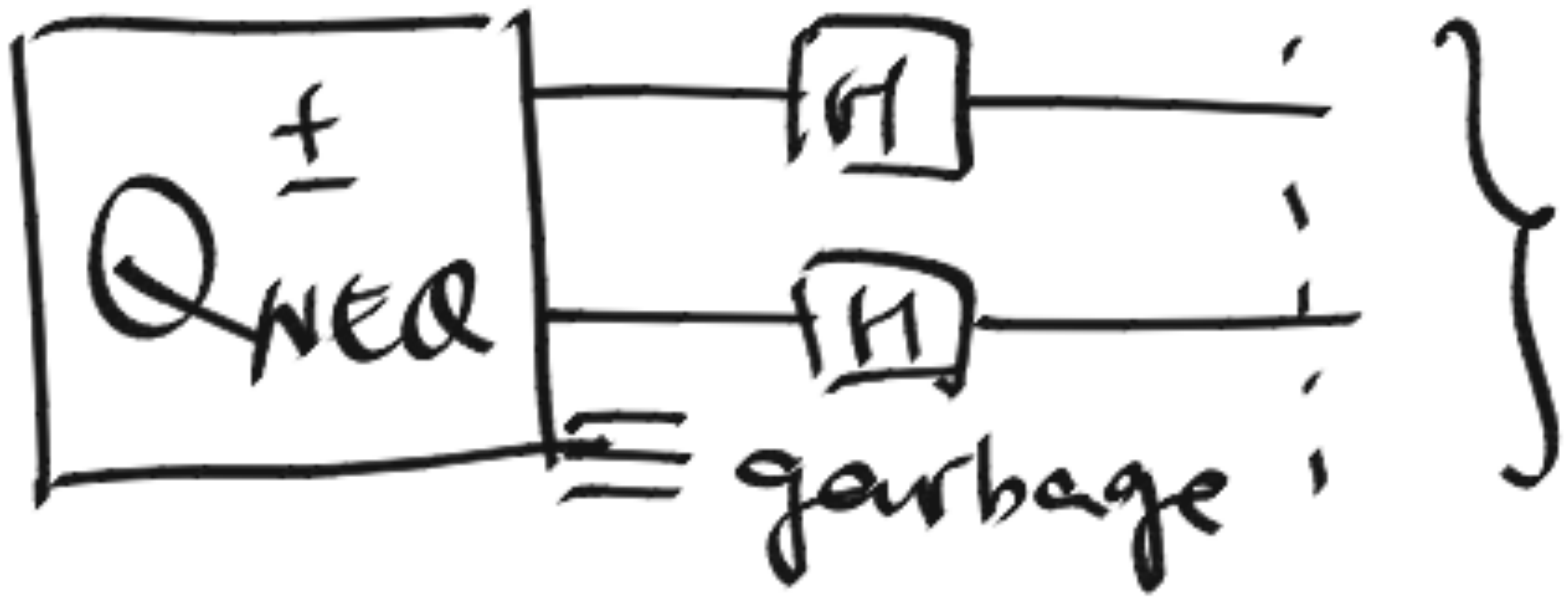


- Final state?
- Reason: Hidden “XOR-pattern” in truth table of NEQ.
- Remark: What really happened is



unentangled ancillas.

- Suppose $Q^{\pm}_{\mathsf{NEQ}}$ produces garbage



- Final state is …?
- Conclusion: un-computing garbage is important.