ΝΑΝΟΗΛΕΚΤΡΟΝΙΚΗ & ΚΒΑΝΤΙΚΕΣ ΠΥΛΕΣ
6η Διάλεξη

Βιβλιογραφία: EXPLORATIONS IN QUANTUM COMPUTING, Colin P. Williams (2nd edition, Springer-Verlag, 2011), chapter 2.

Reversible Gates

- NOT, SWAP, and CNOT
- Universal Reversible Gates: FREDKIN and TOFFOLI
- Reversible Gates Expressed as Permutation Matrices
- Cost of Simulating Irreversible Computations Reversibly
- Ancillae in Reversible Computing
- Universal Reversible Basis
- Can All Boolean Circuits Be Simulated Reversibly?

Reversible Gates

- ✓ Any classical computation can always be decomposed into a sequence of logic gates that act on only a few bits at a time.
- ✓ *Universal Gates: NAND and NOR*, are sufficient to express any desired computation.
- ✓ Restricting circuits to using a single type of universal gate does *not necessarily lead to the smallest circuit* for computing a desired Boolean function but it does allow chip manufacturers to perfect the design and manufacturing process for the universal gate.
- ✓ Today, the microprocessor industry pursues this strategy by basing their circuits on the NAND ("NOT AND") gates: $a\text{NAND}b \equiv \neg(a \wedge b)$, often written as $a|b$, and is universal for classical irreversible computing.
- ✓ Logical irreversibility comes at a price: energy must be dissipated when information is erased, in the amount *kT ln2* per bit erased ($k=1.3805 \times 10^{-23}$ JK^{-1}). Miniaturization of computer technology, implementing nanoelectronics, will be impeded by the difficulty of removing this unwanted waste heat from deep within the irreversible circuitry.

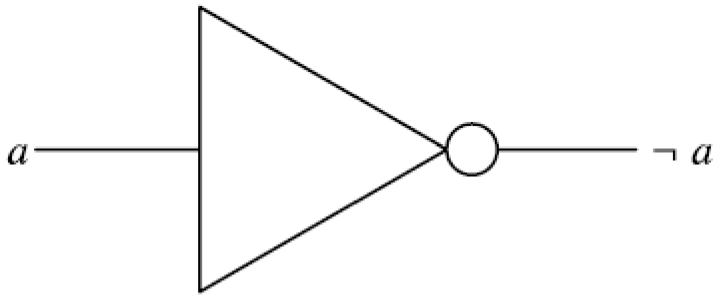
- ❖ One way chip manufacturers can suppress the unwanted heat produced as a side effect of running irreversible logic gates is to modify their chip designs to use only reversible logic gates. In a reversible logic gate there is a unique input for a unique output and vice versa.
- ❖ Reversible gates never erase any information when they act, and a computation based on reversible logic can be run forward to obtain an answer, the answer copied, and then the whole computation undone to recover all the energy expended apart from the small amount used to copy the answer at the mid-way point.

Reversible Gates: NOT and SWAP

Truth table of NOT

a	$\neg a$
0	1
1	0

Icon for the XOR gate—**a 1-bit** logically reversible gate

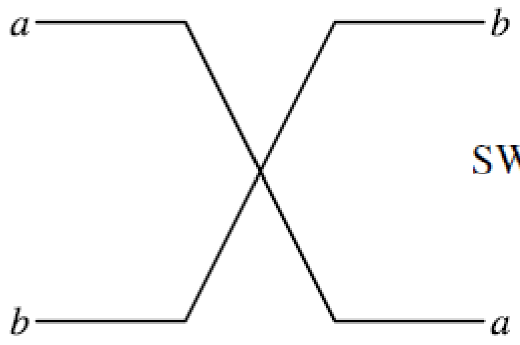


$$\text{NOT} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Truth table of SWAP

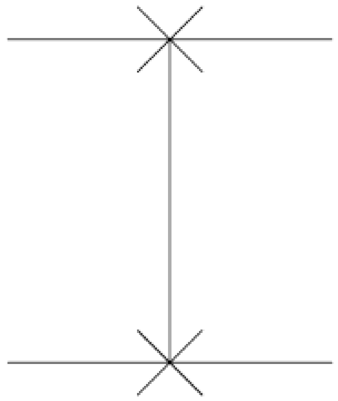
a	b	a'	b'
0	0	0	0
0	1	1	0
1	0	0	1
1	1	1	1

Icon for the SWAP gate—**a 2-bit** logically reversible gate



$$\text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Alternative icon for a SWAP gate that is more common in quantum circuit diagrams. The reason for having a different icon for SWAP in quantum circuits compared to classical circuits is that many **implementations of quantum circuits do not have physical wires as such**. Hence, it could be misleading to depict a SWAP operation as a crossing of wires. Instead, a SWAP operation can be achieved as **the result of a sequence of applied fields**.

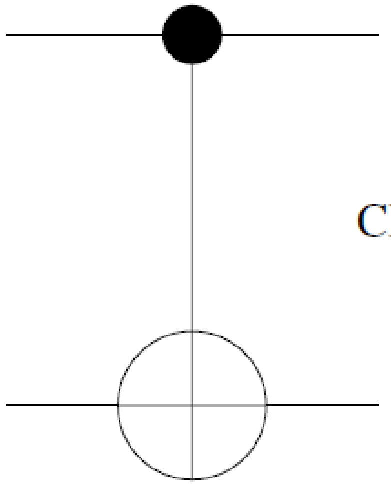


Reversible Gates: CNOT

Truth table of CNOT

a	b	a'	b'
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

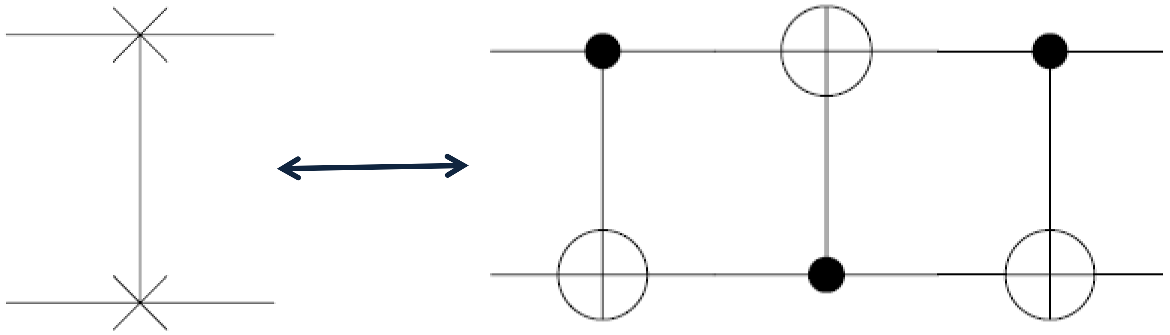
Icon for the CNOT gate—**a 2-bit** logically reversible gate



$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

"controlled-NOT"

- ✓ The decision to negate or not negate the second bit is controlled by the value of the first bit.
- ✓ The effect of CNOT is to flip the bit value of the 2nd bit if and only if the 1st bit is set to 1.



The SWAP gate can be obtained from a sequence of three CNOT gates.

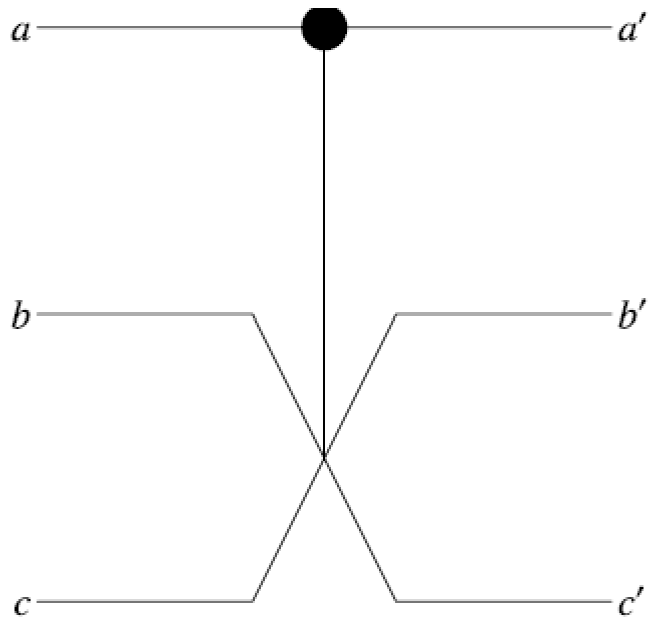
Universal Reversible Gates: FREDKIN

The smallest gates that are both reversible and universal require three inputs and three outputs

Truth table of FREDKIN

a	b	c	a'	b'	c'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	1	1

Icon for the FREDKIN



FREDKIN (controlled-SWAP) gate

- ✓ The decision to SWAP the second and third bit is controlled by the value of the first bit.
- ✓ The effect of FREDKIN is to swap the values of 2nd and 3rd bits only if the 1st bit is set to 1.

Universal Reversible Gates: TOFFOLI

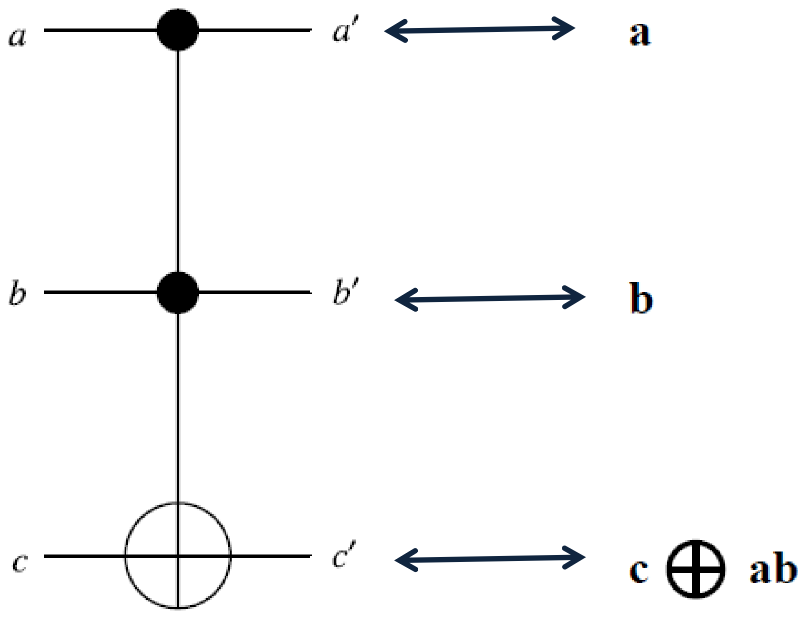
We can make any irreversible classical gate into an equivalent but reversible gate by using the Toffoli gate, a reversible classical gate.

Truth table of TOFFOLI

a	b	c	a'	b'	c'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

Icon for the TOFFOLI

Output for reversible NAND



- ✓ The values of the first two input bits control whether the third input bit is flipped.
- ✓ It is flipping the third input bit if, and only if, the first two input bits are both 1.
- ✓ The TOFFOLI (controlled-controlled NOT) gate can simulate a reversible NAND: by choosing c = 1, we get: $a' = a,\ b' = b$ and $c' = 1 \oplus ab = \neg ab.$

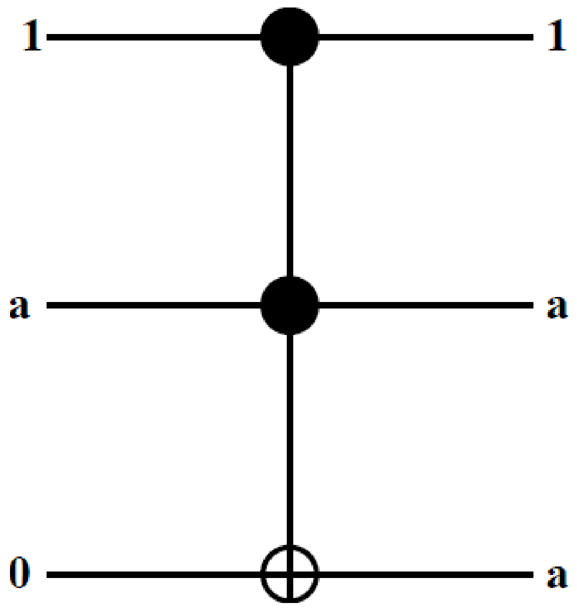
TOFFOLI gate can simulate NAND and FANOUT

- ✓ With NAND and FANOUT it is possible to construct any classical gate.
- ✓ The Toffoli gate can thus be used to construct, in a reversible way, any classical gate *f*.

Truth table of TOFFOLI

a	b	c	a'	b'	c'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

Icon for the TOFFOLI acting as a FANOUT



The Toffoli gate transformation is unitary, and thus can be implemented as a quantum gate.

Reversible Gates Expressed as Permutation Matrices

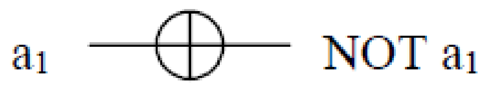
- ✓ Any n-bit reversible gate must specify how to map each distinct bit string input into a distinct bit string output of the same length.
- ✓ Thus no two inputs are allowed to be mapped to the same output and vice versa.
- ✓ This ensures the mapping is reversible.

Since reversible logic gates are symmetric with respect to the number of inputs and outputs, we can represent them in ways other than the truth table, in the form of a matrix or as a graphic.

Example

	NOT	⟨0⟩	⟨1⟩	INPUT
OUTPUT	⟨0⟩	0	1	
	⟨1⟩	1	0	

$$\text{NOT} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$



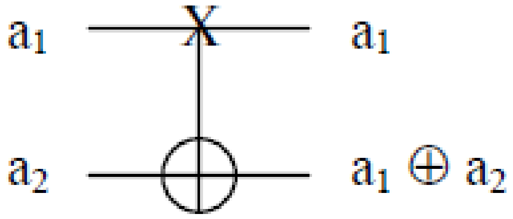
Thus, a natural way to represent an *n*-bit reversible gate is as an array whose rows and columns are indexed by the 2^n possible bit strings expressible in *n*-bits.

Permutation Matrices: CNOT and SWAP gates

- ✓ The resulting array will contain a single 1 in each row and column and zeroes everywhere else, and will therefore be a permutation matrix.
- ✓ The matrices corresponding to classical reversible gates are always permutation matrices, i.e., 0/1 matrices having a single 1 in each row and column, and permutation matrices are always unitary matrices.

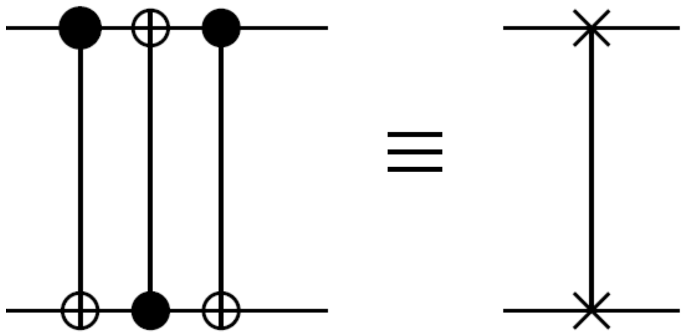
C-NOT	⟨00⟩	⟨01⟩	⟨10⟩	⟨11⟩
⟨00⟩	1	0	0	0
⟨01⟩	0	1	0	0
⟨10⟩	0	0	0	1
⟨11⟩	0	0	1	0

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

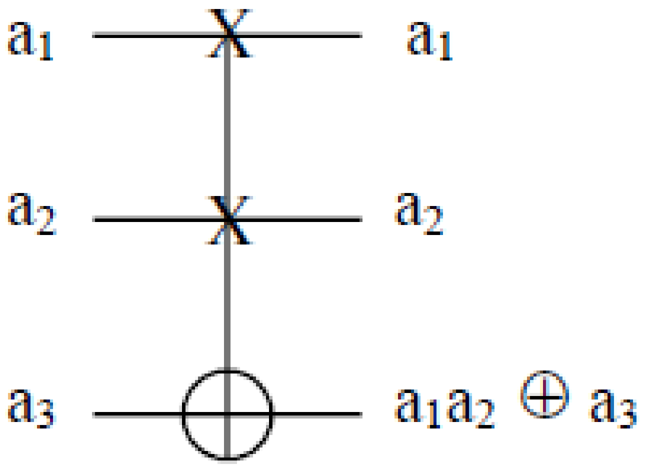


SWAP	⟨00⟩	⟨01⟩	⟨10⟩	⟨11⟩
⟨00⟩	1	0	0	0
⟨01⟩	0	0	1	0
⟨10⟩	0	1	0	0
⟨11⟩	0	0	0	1

$$\text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

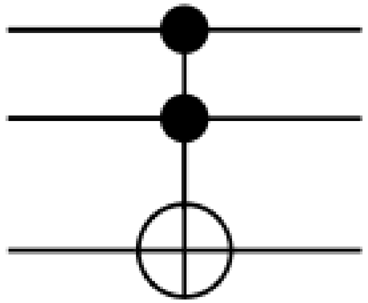


Permutation Matrix of TOFFOLI gate



CC-NOT	⟨000⟩	…					⟨110⟩	⟨111⟩
⟨000⟩	1	0	0	0	0	0	0	0
⟨001⟩	0	1	0	0	0	0	0	0
⟨010⟩	0	0	1	0	0	0	0	0
⟨011⟩	0	0	0	1	0	0	0	0
⟨100⟩	0	0	0	0	1	0	0	0
⟨101⟩	0	0	0	0	0	1	0	0
⟨110⟩	0	0	0	0	0	0	0	1
⟨111⟩	0	0	0	0	0	0	1	0

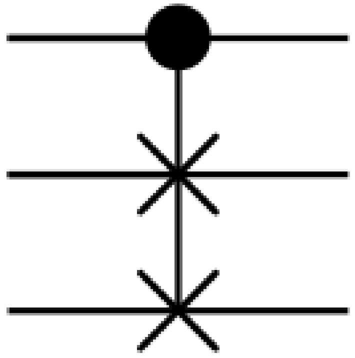
Circuit representation of Toffoli gate



$$\text{TOFFOLI:}\quad \begin{array}{c|cccccccc} & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\ \hline 000 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 001 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 010 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 011 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 100 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 101 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 110 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 111 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}$$

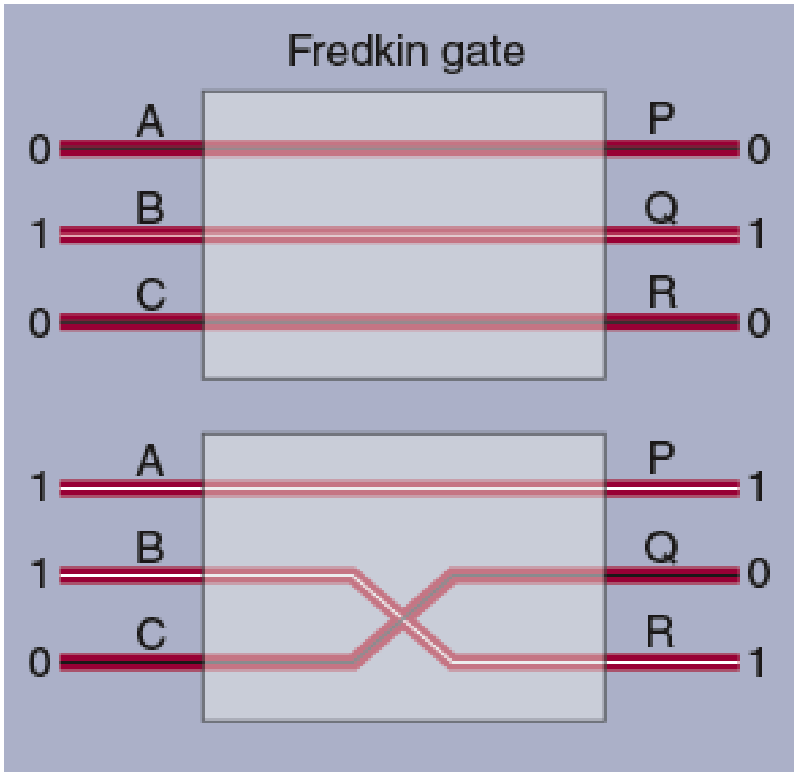
Permutation Matrix of FREDKIN gate

Circuit representation of Fredkin gate



$$\text{FREDKIN:}\quad \begin{array}{c|cccccccc} & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\ \hline 000 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 001 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 010 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 011 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 100 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 101 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 110 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 111 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

$$\mathrm{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



FREDKIN gate performs a controlled-SWAP operation.

$$|000\rangle \xrightarrow{\text{FREDKIN}} |000\rangle \qquad |100\rangle \xrightarrow{\text{FREDKIN}} |100\rangle$$

$$|001\rangle \xrightarrow{\text{FREDKIN}} |001\rangle \qquad |101\rangle \xrightarrow{\text{FREDKIN}} |110\rangle$$

$$|010\rangle \xrightarrow{\text{FREDKIN}} |010\rangle \qquad |110\rangle \xrightarrow{\text{FREDKIN}} |101\rangle$$

$$|011\rangle \xrightarrow{\text{FREDKIN}} |011\rangle \qquad |111\rangle \xrightarrow{\text{FREDKIN}} |111\rangle$$

The effect of TOFFOLI gate on an input bit string

TOFFOLI gates act on three bits, and every column vector consists of $2^3 = 8$ slots:

$$000 \equiv \begin{pmatrix}1\\0\\0\\0\\0\\0\\0\\0\end{pmatrix}, \quad 001 \equiv \begin{pmatrix}0\\1\\0\\0\\0\\0\\0\\0\end{pmatrix}, \quad 010 \equiv \begin{pmatrix}0\\0\\1\\0\\0\\0\\0\\0\end{pmatrix}, \quad \ldots \quad 111 \equiv \begin{pmatrix}0\\0\\0\\0\\0\\0\\0\\1\end{pmatrix}$$

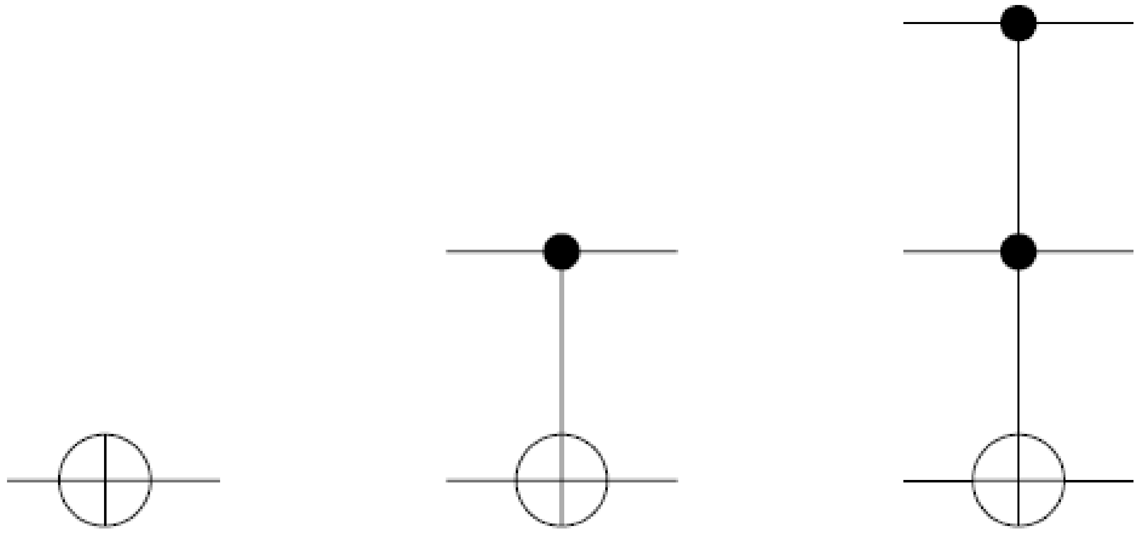
The effect of TOFFOLI gate on such an input is given by vector-matrix multiplication:

$$\mathrm{TOFFOLI}|110\rangle = \begin{pmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&1&0\end{pmatrix} \cdot \begin{pmatrix}0\\0\\0\\0\\0\\0\\1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\0\\0\\0\\0\\0\\1\end{pmatrix} = |111\rangle$$

Universal Reversible Basis

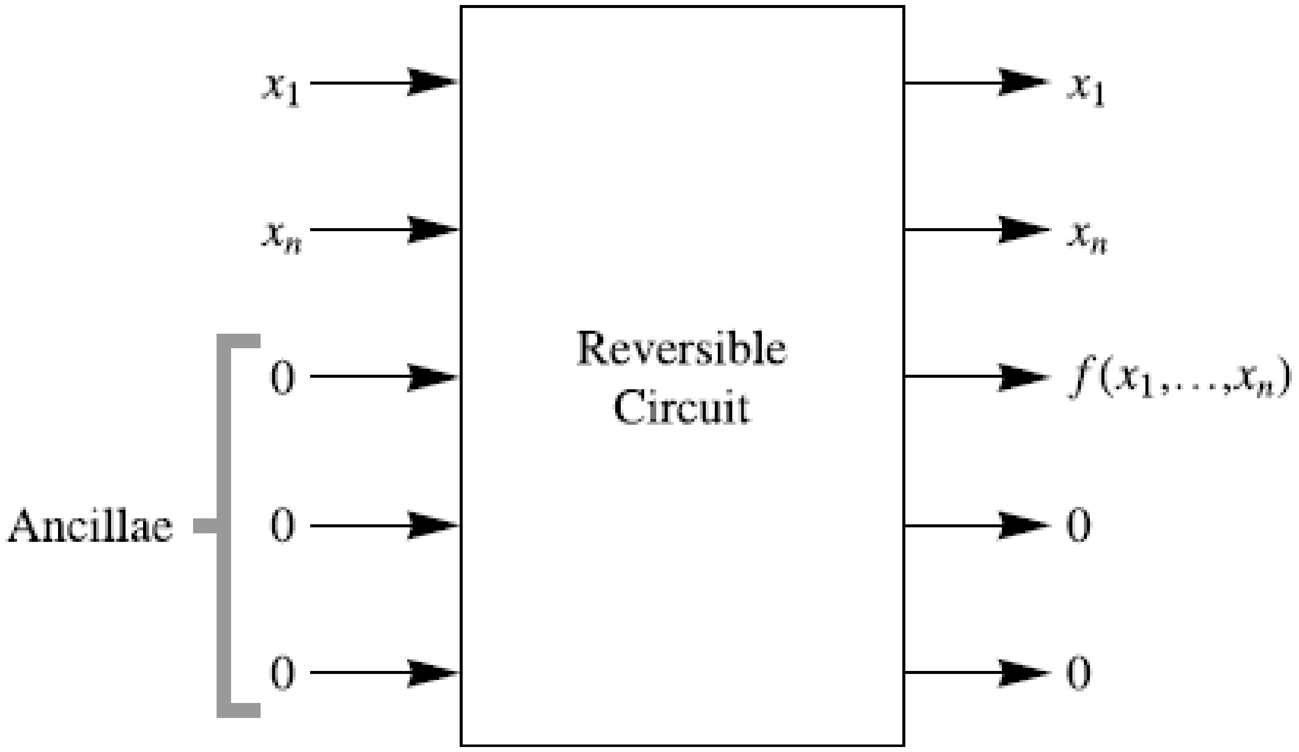
The reversible basis consisting of NOT, CNOT and Toffoli gates is universal for reversible computation.

NOT	CNOT	TOFFOLI
$a \mapsto 1 \oplus a$	$\begin{pmatrix} a \\ b \end{pmatrix} \mapsto \begin{pmatrix} a \\ a \oplus b \end{pmatrix}$	$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \mapsto \begin{pmatrix} a \\ b \\ c \oplus (a \cdot b) \end{pmatrix}$



Ancillae in Reversible Computing

- ✓ Every permutation on $\{0, 1\}^n$ can be realized by means of a reversible circuit over the NOT-CNOT-TOFFOLI basis using at most one ancilla bit.
- ✓ A reversible circuit computes a Boolean function $f : \{0, 1\}^n \longrightarrow \{0, 1\}$.



At the end of the computation all ancillae have their initial values, except one ancilla bit, designated as the “answer” bit, that carries the value of the function.

Balanced and unbalanced Boolean function in Reversible Computing

Every reversible circuit on *m* inputs, computing *f*, has exactly *m* outputs with one of them considered the value of *f*.

Balanced *f*

- If *m* = *n*, i.e., there is no ancilla bit, then every output function must be a balanced Boolean function.
- A balanced function on $\{0, 1\}^n$ returns a value “1” for 2*n*−1 of its inputs and a value “0” for the other 2*n*−1 inputs.
- A balanced function there is in NOT gate.

Not balanced *f*

- If the function is not balanced, we require m > n and there must be at least one ancilla bit. For example, the Toffoli gate can be used to construct NAND and FANOUT, using some ancilla bits 0 or 1 as input.
- We also have some *garbage* in the output which is not needed in the rest of computation.
- The ancilla and garbage bits are only important to make the computation reversible.
- We can represent this by: $(x, 0, 0) \mapsto (x, f(x), g(x))$, where we can assume all ancilla bits are 0 with the aid of the NOT gate.

Reversible Circuits

1. Assume all ancilla bits are 0 and the garbage bits be in a standard state, by adding a fourth register in an arbitrary state y so that the effect of the reversible circuit is: $(x,0,0,y) \mapsto (x,f(x),g(x),y)$
2. Next, use the CNOT gate to induce:
$$(x,f(x),g(x),y) \mapsto (x,f(x),g(x),y \oplus f(x))$$
3. Finally, apply the inverse of the circuit 1 to the first three registers to obtain:
$$(x,f(x),g(x),y \oplus f(x)) \mapsto (x,0,0,y \oplus f(x))$$

Deleting the ancilla bits, the overall evaluation is:

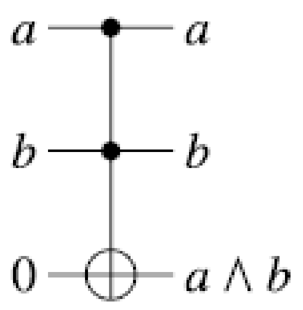
$$(x,y) \mapsto (x,y \oplus f(x))$$

which is the standard reversible circuit for evaluating *f*.

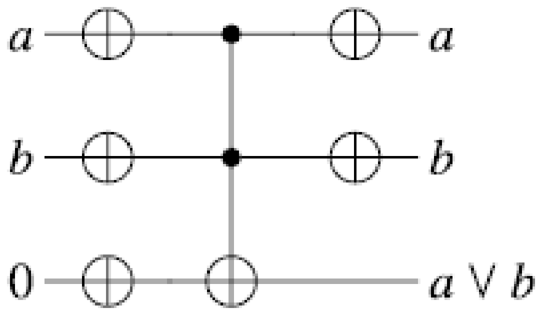
Can All Boolean Circuits Be Simulated Reversibly?

A simple (naïve) method for simulating any Boolean (irreversible) circuit is to replace each irreversible gate in the circuit with its reversible counterpart.

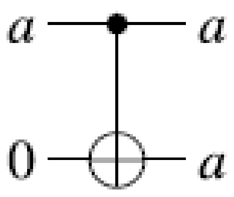
Reversible simulation of classical gates



AND gate



OR gate



FAN–OUT gate

Synchronous circuit:

- all paths from the inputs to any gate have the same length,
- have delay (identity) gates,
- gates at level *m* get inputs from gates at level *m* − 1.
- an irreversible circuit can be synchronous.

Terminology

For a Boolean circuit:

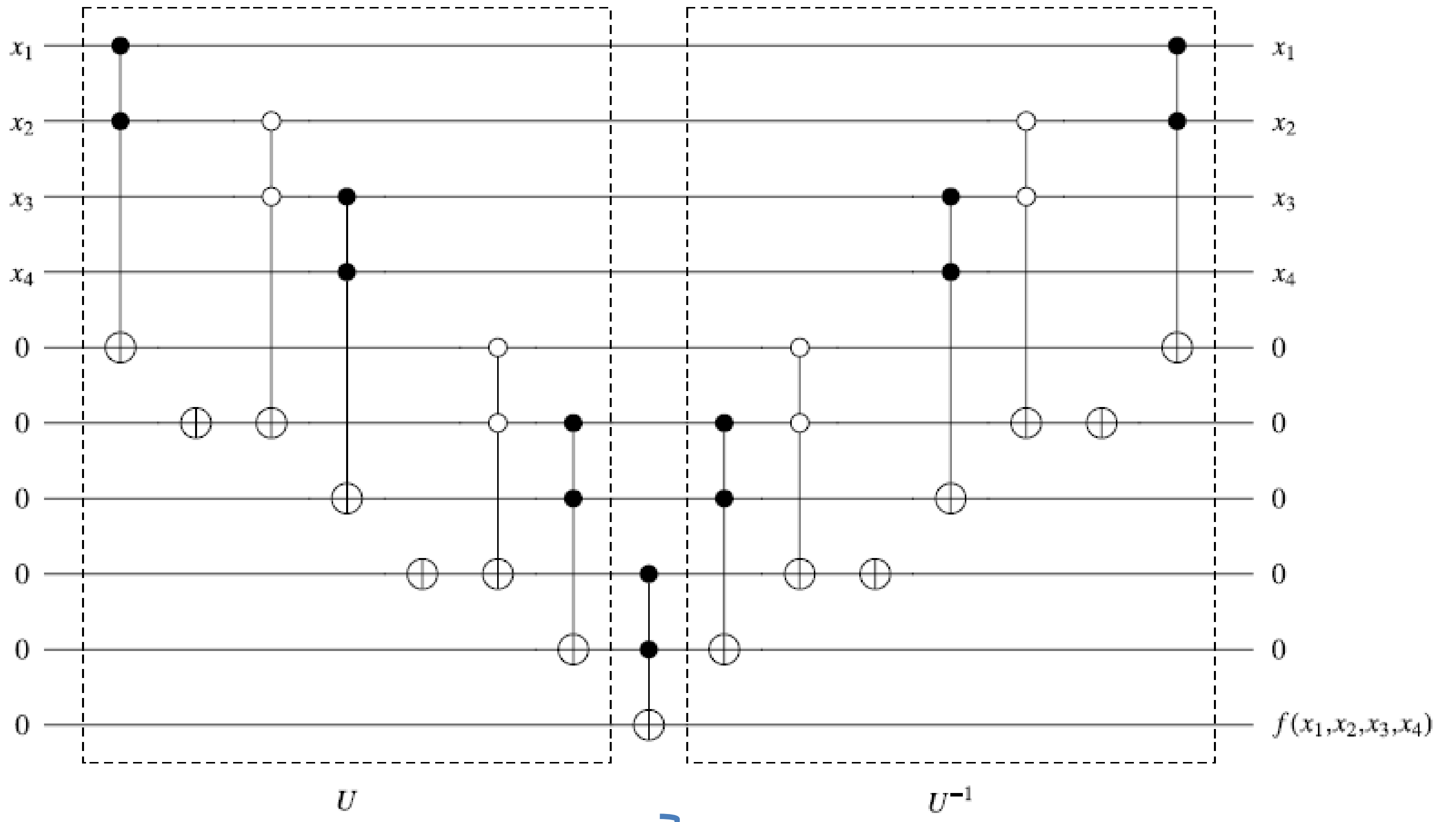
- Size-*t* is the total number of gates.
- Depth-*d* is the number of levels.
- Width-*w* is the maximum number of gates in any level.
- Space-*A* is the number of ancillae required to perform a computation.

Efficiency of Reversible Simulation

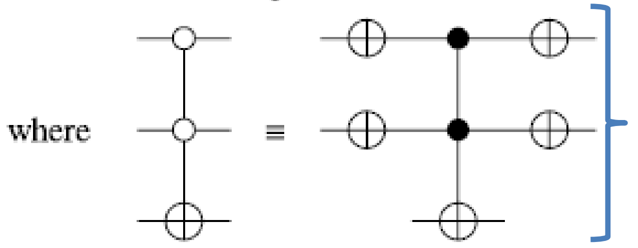
There is a procedure to create a reversible circuit that simulates an irreversible circuit while making substantial savings in the number of ancillae used.

1. Simulate the gates in the first-half levels.
2. Keep the results of the gates in level *d*/2 separately.
3. Clean up the ancillae bits.
4. Use them to simulate the gates in the second-half levels.
5. After computing the output, clean up the ancillae bits.
6. Clean up the result of level d/2.

$f(x_1,x_2,x_3,x_4) = ((x_1 \wedge x_2) \vee (x_2 \vee x_3)) \wedge ((x_2 \vee x_3) \wedge (x_3 \wedge x_4))$ computed reversibly



This method needs roughly half the number of ancillae used by the previous (naive) method.



The control gate is a hollow circle instead of the filled circle on the top line, and it flips the target bit **if the control bit is 0**.

Efficiency of Reversible Simulation

By applying the above procedure recursively, on a circuit of *size t*, *depth d*, and *width w* we obtain the following recursive relations for *S, the size*, and *A, the number of ancillae* needed:

$$S(t) \leq 6S(t/2) + O(1), \quad A(d) \leq A(d/2) + w + 1.$$

Solving these recursion relations leads to the following result.

Any irreversible computation (in the synchronous form) having t gates, depth d, and width w, can be simulated by a reversible circuit having $O(t^{2.58})$ gates, and at most $(w + 1) \log d + O(1)$ ancillae.

Conclusion

Since most of the irreversible computations going on inside a computer could, in principle, be implemented using reversible logic gates, then:

there is no need of net energy to run, apart from operations that require erasure of information, such as overwriting a memory register to make a copy of an answer!

Information

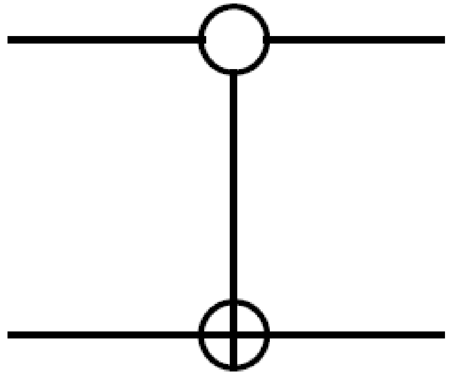
- Thus, the vast majority of the operations employed along the way can be done reversibly, and hence, don't generate any more information in their output than they had in their input.
- Computers just take the known information given as input and re-arrange it.

"Computers are useless—they only give answers!"

Pablo Picasso

Problems

Problem 1: There is nothing special about the control bit in the CNOT gate to take value 1. The control gate in figure flips the target bit if the control bit is 0. This is represented pictorially by a hollow circle instead of the filled circle on the top line. What is its matrix representation?



Problem 2: Show that the circuit on the left in Figure, swaps the two qubits. It is denoted by the figure on the right.