量子計算之數學基礎
MA5501*

Chapter 1. Logic Circuits（邏輯電路）

§1.1 Classical Logic Gates

§1.2 Universal Gates

§1.3 How A Classical Computer Adds Numbers

§1.1 Classical Logic Gates

Question: What is a classical computer or what does a classical computer do?

In a classical computer the processor essentially performs nothing more than a sequence of transformations of a **classical state** into another one. In mathematical terminology, a classical processor performs a ***sequence*** of evaluation of maps of the form

$$\begin{array}{rccc} f: & \{0,1\}^n & \to & \{0,1\}^m \\ & x & \mapsto & f(x) \end{array}$$

This is what we will refer to as the classical computational process, which is realized with a concatenation of **classical gates** and **circuits**.

§1.1 Classical Logic Gates

Question: What is a classical computer or what does a classical computer do?

In a classical computer the processor essentially performs nothing more than a sequence of transformations of a **classical state** into another one. In mathematical terminology, a classical processor performs a ***sequence*** of evaluation of maps of the form

$$\begin{array}{rccc} f: & \{0,1\}^n & \to & \{0,1\}^m \\ & x & \mapsto & f(x) \end{array}$$

This is what we will refer to as the classical computational process, which is realized with a concatenation of **classical gates** and **circuits**.

§1.1 Classical Logic Gates

Definition

A classical logical gate, also called a Boolean function, is a map

$$\begin{array}{rccc} g: & \{0,1\}^n & \to & \{0,1\} \\ & (x_1,\cdots,x_n) & \mapsto & g(x_1,\cdots,x_n) \end{array}.$$

We define an extended classical logical gate g as a map

$$\begin{array}{rccc} g: & \{0,1\}^n & \to & \{0,1\}^m \\ & (x_1,\cdots,x_n) & \mapsto & \big(g_1(x_1,\cdots,x_n),\cdots,g_m(x_1,\cdots,x_n)\big) \end{array},$$

where each g_j is a classical logic gate. A classical gate g is called reversible if it is a bijection and thus invertible.

Example

The addition $\oplus$ on $\mathbb{Z}_2$ can be treated as a classical logic gate from $\{0,1\}^2$ to $\{0,1\}$ given by $(a,b) \mapsto a \oplus b$.

§1.1 Classical Logic Gates

Definition

A classical logical gate, also called a Boolean function, is a map

$$\begin{array}{rccc} g: & \{0,1\}^n & \to & \{0,1\} \\ & (x_1,\cdots,x_n) & \mapsto & g(x_1,\cdots,x_n) \end{array}.$$

We define an extended classical logical gate g as a map

$$\begin{array}{rccc} g: & \{0,1\}^n & \to & \{0,1\}^m \\ & (x_1,\cdots,x_n) & \mapsto & \big(g_1(x_1,\cdots,x_n),\cdots,g_m(x_1,\cdots,x_n)\big) \end{array},$$

where each g_j is a classical logic gate. A classical gate g is called reversible if it is a bijection and thus invertible.

Example

The addition $\oplus$ on $\mathbb{Z}_2$ can be treated as a classical logic gate from $\{0,1\}^2$ to $\{0,1\}$ given by $(a,b) \mapsto a \oplus b$.

§1.1 Classical Logic Gates - **NOT** gate

The **NOT** gate, also called an inverter, is a logic gate which implements logical negation. It behaves according to the truth table below:

INPUT	OUTPUT
0	1
1	0

The analytical form of the **NOT** gate is given by $\mathbf{NOT}(a) = 1 - a$ for $a \in \{0, 1\}$. We note that the **NOT** gate is reversible, and the inverse of the $\mathbf{NOT}$ gate is itself.

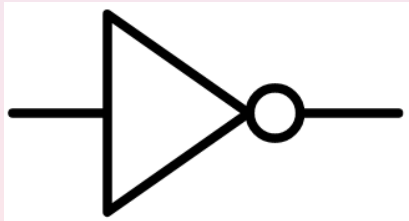


Figure 1: Traditional **NOT** Gate (Inverter) symbol

§1.1 Classical Logic Gates - **AND** and **OR** gates

The **AND** gate (及閘) is a basic digital logic gate that implements logical conjunction, and the **OR** gate (或閘) is a digital logic gate that implements logical disjunction. They behave according to the truth tables below:

INPUT		**OUTPUT**
A	B	A **AND** B
0	0	0
0	1	0
1	0	0
1	1	1

INPUT		**OUTPUT**
A	B	A **OR** B
0	0	0
0	1	1
1	0	1
1	1	1

Analytically, the function of **AND** finds the product of binary digits, while the function of **OR** finds the maximum between two binary digits; that is,

$$\mathbf{AND}(a,b) = a \cdot b \quad \text{and} \quad \mathbf{OR}(a,b) = \max\{a,b\} \quad \forall\, a,b \in \{0,1\}\,.$$

§1.1 Classical Logic Gates - **AND** and **OR** gates

The **AND** gate (及閘) is a basic digital logic gate that implements logical conjunction, and the **OR** gate (或閘) is a digital logic gate that implements logical disjunction. They behave according to the truth tables below:

INPUT		**OUTPUT**
A	B	A **AND** B
0	0	0
0	1	0
1	0	0
1	1	1

INPUT		**OUTPUT**
A	B	A **OR** B
0	0	0
0	1	1
1	0	1
1	1	1

Analytically, the function of **AND** finds the product of binary digits, while the function of **OR** finds the maximum between two binary digits; that is,

$$\mathbf{AND}(a,b) = a \cdot b \quad \text{and} \quad \mathbf{OR}(a,b) = \max\{a,b\} \quad \forall\, a,b \in \{0,1\}\,.$$

§1.1 Classical Logic Gates - **AND** and **OR** gates

The logic gate symbols for the **AND** and **OR** gates are

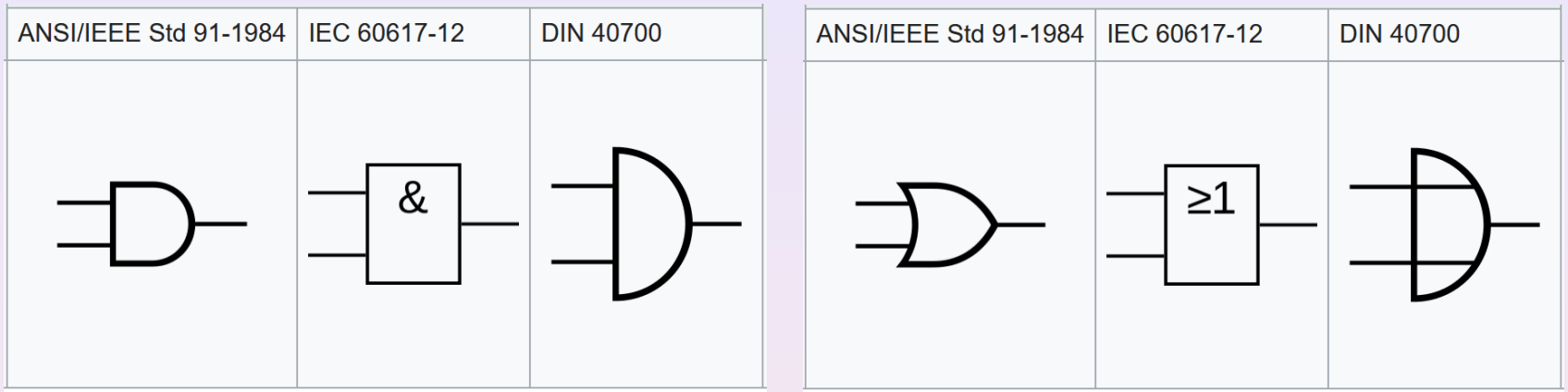


Figure 2: Logic gate symbols for **AND** (left) and **OR** (right) gates

We note that the **AND** gate and the **OR** gate are not reversible.

§1.1 Classical Logic Gates - **NAND** and **NOR** gates

The **NAND** gate (NOT-AND，反及閘) is a logic gate whose output is complement to that of an **AND** gate. In other words, the **NAND** gate produces an output which is false only if all its inputs are true.

On the other hand, the **NOR** gate (NOT-OR，反或閘) is a logic gate whose output is complement to that of an **OR** gate. They behave according to the truth tables below:

INPUT		OUTPUT
A	B	A **NAND** B
0	0	1
0	1	1
1	0	1
1	1	0

INPUT		OUTPUT
A	B	A **NOR** B
0	0	1
0	1	0
1	0	0
1	1	0

§1.1 Classical Logic Gates - **NAND** and **NOR** gates

The **NAND** gate (NOT-AND，反及閘) is a logic gate whose output is complement to that of an **AND** gate. In other words, the **NAND** gate produces an output which is false only if all its inputs are true. On the other hand, the **NOR** gate (NOT-OR，反或閘) is a logic gate whose output is complement to that of an **OR** gate. They behave according to the truth tables below:

INPUT		OUTPUT
A	B	A **NAND** B
0	0	1
0	1	1
1	0	1
1	1	0

INPUT		OUTPUT
A	B	A **NOR** B
0	0	1
0	1	0
1	0	0
1	1	0

§1.1 Classical Logic Gates - **NAND** and **NOR** gates

The **NAND** gate (NOT-AND，反及閘) is a logic gate whose output is complement to that of an **AND** gate. In other words, the **NAND** gate produces an output which is false only if all its inputs are true. On the other hand, the **NOR** gate (NOT-OR，反或閘) is a logic gate whose output is complement to that of an **OR** gate. They behave according to the truth tables below:

INPUT		**OUTPUT**
A	B	A **NAND** B
0	0	1
0	1	1
1	0	1
1	1	0

INPUT		**OUTPUT**
A	B	A **NOR** B
0	0	1
0	1	0
1	0	0
1	1	0

§1.1 Classical Logic Gates - **NAND** and **NOR** gates

The logic gate symbols for the **NAND** and **NOR** gates are

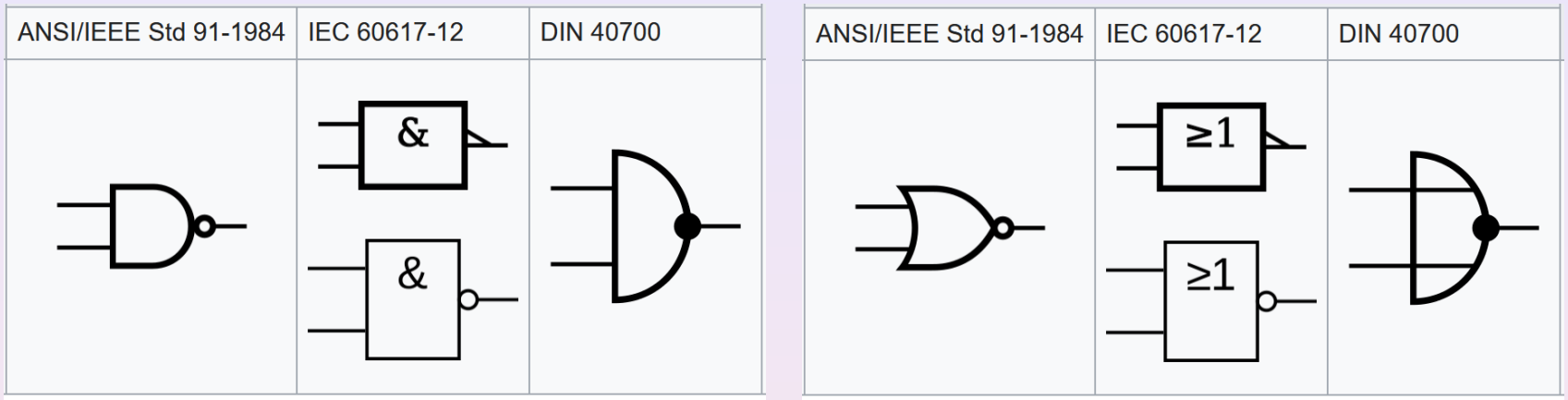


Figure 3: Logic gate symbols for **NAND** (left) and **NOR** (right) gates

We also note that the **NAND** gate and the **NOR** gates are not reversible.

§1.1 Classical Logic Gates - **XOR** and **XNOR** gates

The **XOR** gate (sometimes **EOR**, or **EXOR** and pronounced as Exclusive OR，互斥或閘) is a digital logic gate (from $\{0,1\}^2$ to $\{0,1\}$) that gives a true (1 or HIGH) output when the number of true inputs is odd. The **XNOR** gate (sometimes **ENOR**, **EXNOR** or **NXOR** and pronounced as Exclusive NOR，反互斥或閘) is a digital logic gate whose function is the logical complement of the Exclusive OR (**XOR**) gate. The **XOR** and **XNOR** gates behave according to the truth table below:

INPUT		OUTPUT
A	B	A **XOR** B
0	0	0
0	1	1
1	0	1
1	1	0

INPUT		OUTPUT
A	B	A **XNOR** B
0	0	1
0	1	0
1	0	0
1	1	1

§1.1 Classical Logic Gates - **XOR** and **XNOR** gates

The **XOR** gate (sometimes **EOR**, or **EXOR** and pronounced as Exclusive OR，互斥或閘) is a digital logic gate (from $\{0,1\}^2$ to $\{0,1\}$) that gives a true (1 or HIGH) output when the number of true inputs is odd. The **XNOR** gate (sometimes **ENOR**, **EXNOR** or **NXOR** and pronounced as Exclusive NOR，反互斥或閘) is a digital logic gate whose function is the logical complement of the Exclusive OR (**XOR**) gate. The **XOR** and **XNOR** gates behave according to the truth table below:

INPUT		OUTPUT
A	B	A **XOR** B
0	0	0
0	1	1
1	0	1
1	1	0

INPUT		OUTPUT
A	B	A **XNOR** B
0	0	1
0	1	0
1	0	0
1	1	1

§1.1 Classical Logic Gates - **XOR** and **XNOR** gates

The **XOR** gate (sometimes **EOR**, or **EXOR** and pronounced as Exclusive OR，互斥或閘) is a digital logic gate (from $\{0,1\}^2$ to $\{0,1\}$) that gives a true (1 or HIGH) output when the number of true inputs is odd. The **XNOR** gate (sometimes **ENOR**, **EXNOR** or **NXOR** and pronounced as Exclusive NOR，反互斥或閘) is a digital logic gate whose function is the logical complement of the Exclusive OR (**XOR**) gate. The **XOR** and **XNOR** gates behave according to the truth table below:

INPUT		**OUTPUT**
A	B	A **XOR** B
0	0	0
0	1	1
1	0	1
1	1	0

INPUT		**OUTPUT**
A	B	A **XNOR** B
0	0	1
0	1	0
1	0	0
1	1	1

§1.1 Classical Logic Gates - **XOR** and **XNOR** gates

The analytic form of the **XOR** and the **XNOR** gates are

$$\mathbf{XOR}(a,b) = a \oplus b = a + b - 2ab \qquad \forall\, a, b \in \{0,1\}\,,$$
$$\mathbf{XNOR}(a,b) = 1 \oplus a \oplus b = 1 + 2ab - a - b \qquad \forall\, a, b \in \{0,1\}\,.$$

The logic gate symbols for the **XOR** and **XNOR** gates are

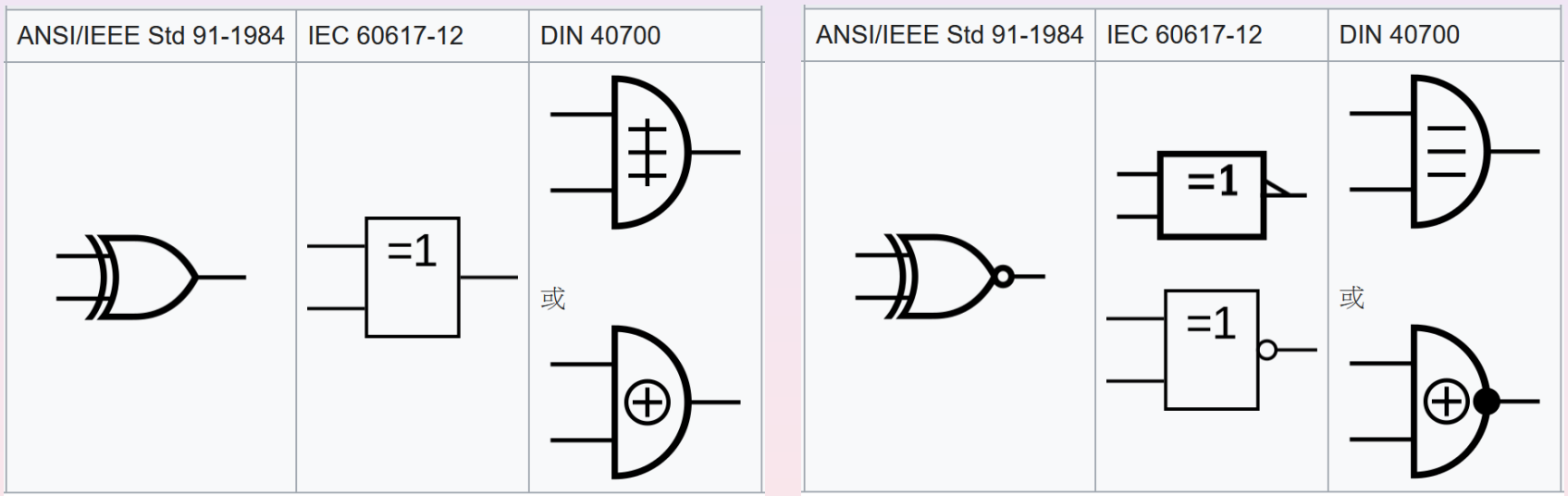


Figure 4: Logic gate symbols for **XOR** and **XNOR** gates

The **XOR** and **XNOR** gates are not reversible.

§1.1 Classical Logic Gates - the Toffoli gate

The Toffoli gate, also called **CCNOT** (pronounced controlled-controlled-not) gate, is a digital logic gate which behaves according to the truth table below:

INPUT			OUTPUT		
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

The Toffoli gate is a reversible logic gate.

§1.1 Classical Logic Gates - the Toffoli gate

The analytic form of the Toffoli gate is

$$\textbf{TOF}(a, b, c) = (a, b, ab \oplus c) \qquad \forall\, a, b, c \in \{0, 1\}$$

and the symbol for the Toffoli gate is

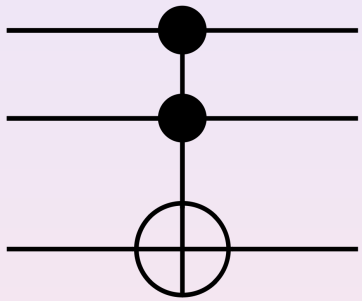


Figure 5: Circuit representation of Toffoli gate

The n-bit Toffoli gate is a generalization of Toffoli gate. It takes n bits $x_1, x_2, \cdots, x_n$ as inputs and outputs n bits: the first $n-1$ output bits are just $x_1, \cdots, x_{n-1}$, and the last output bit is $x_1 x_2 \cdots x_{n-1} \oplus x_n$.

§1.2 Universal Gates

Universal gates can be "combined" to perform "all" Boolean functions. Before talking about the precise definition of the universality of classical logic gates, we need to introduce two basic operations that can be easily performed by classical computers (via storing/swapping data in the memory, maybe?).

Definition (Restriction/Re-ordering)

Let $n, \ell \in \mathbb{N}$ and $\ell \leqslant n$. For a pairwise distinct set $\{j_1, \cdots, j_\ell\} \subseteq \{1, \cdots, n\}$, the **restriction and/or re-ordering** operation $r^{(n)}_{j_1 j_2 \cdots j_\ell}$ is a classical gate from $\{0,1\}^n$ to $\{0,1\}^\ell$ given by

$$r^{(n)}_{j_1 j_2 \cdots j_\ell}(x_1, \cdots, x_n) = (x_{j_1}, x_{j_2}, \cdots, x_{j_\ell}).$$

§1.2 Universal Gates

Definition (Padding - 填充)

Let $n, \ell \in \mathbb{N}$. For a given point $(y_1, y_2, \cdots, y_\ell) \in \{0,1\}^\ell$ and a pairwise distinct set $\{j_1, \cdots, j_\ell\} \subseteq \{1, 2, \cdots, n+\ell\}$, the **padding** operation $p^{(n)}_{y_1,\cdots,y_\ell;j_1,\cdots,j_\ell}$ is a classical gate from $\{0,1\}^n$ to $\{0,1\}^{n+\ell}$ given by

$$p^{(n)}_{y_1,\cdots,y_\ell;j_1,\cdots,j_\ell}(x_1, \cdots, x_n) = (z_1, \cdots, z_{n+\ell}),$$

where

$$z_k = \begin{cases} x_{k-\#\{r \in \{j_1,\cdots,j_\ell\} \mid r<k\}} & \text{if } k \notin \{j_1, j_2, \cdots, j_\ell\}, \\ y_{j_r} & \text{if } k \in \{j_1, j_2, \cdots, j_\ell\} \text{ and } k = j_r. \end{cases}$$

In other words, the padding operation $p^{(n)}_{y_1,\cdots,y_\ell;j_1,\cdots,j_\ell}$ inserts pre-determined bit values $y_1, \cdots, y_\ell \in \{0,1\}$ at pre-determined slots $j_1, \cdots, j_\ell \in \{1, \cdots, n+\ell\}$.

§1.2 Universal Gates

Definition

Let $\{g_1, g_2, \cdots, g_k\}$ be a collection of classical gates. The collection of all gates that can be constructed from g_1, g_2, $\cdots$, g_k, denoted by $\mathcal{F}[g_1, \cdots, g_k]$, is the set satisfying the following five construction rules:

1. For any pairwise distinct $j_1, \cdots, j_\ell \in \{1, \cdots, \ell\}$, where $\ell, n \in \mathbb{N}$ and $\ell \leqslant n$,
$$r^{(n)}_{j_1 j_2 \cdots j_\ell} \in \mathcal{F}[g_1, \cdots, g_k] .$$
2. For any $y_1, \cdots, y_\ell \in \{0, 1\}$ and pairwise distinct $j_1, \cdots, j_\ell \in \{1, \cdots, n + \ell\}$, where $\ell, n \in \mathbb{N}$,
$$p^{(n)}_{y_1, \cdots, y_\ell; j_1, \cdots, j_\ell} \in \mathcal{F}[g_1, \cdots, g_k] .$$
3. For any $1 \leqslant j \leqslant k$, $g_j \in \mathcal{F}[g_1, \cdots, g_k]$.

§1.2 Universal Gates

Definition

Let $\{g_1, g_2, \cdots, g_k\}$ be a collection of classical gates. The collection of all gates that can be constructed from g_1, g_2, $\cdots$, g_k, denoted by $\mathcal{F}[g_1, \cdots, g_k]$, is the set satisfying the following five construction rules:

1. For any pairwise distinct $j_1, \cdots, j_\ell \in \{1, \cdots, \ell\}$, where $\ell, n \in \mathbb{N}$ and $\ell \leqslant n$,
$$r^{(n)}_{j_1 j_2 \cdots j_\ell} \in \mathcal{F}[g_1, \cdots, g_k] .$$
2. For any $y_1, \cdots, y_\ell \in \{0, 1\}$ and pairwise distinct $j_1, \cdots, j_\ell \in \{1, \cdots, n + \ell\}$, where $\ell, n \in \mathbb{N}$,
$$p^{(n)}_{y_1, \cdots, y_\ell; j_1, \cdots, j_\ell} \in \mathcal{F}[g_1, \cdots, g_k] .$$
3. For any $1 \leqslant j \leqslant k$, $g_j \in \mathcal{F}[g_1, \cdots, g_k]$.

§1.2 Universal Gates

Definition

Let $\{g_1, g_2, \cdots, g_k\}$ be a collection of classical gates. The collection of all gates that can be constructed from g_1, g_2, $\cdots$, g_k, denoted by $\mathcal{F}[g_1, \cdots, g_k]$, is the set satisfying the following five construction rules:

1. For any pairwise distinct $j_1, \cdots, j_\ell \in \{1, \cdots, \ell\}$, where $\ell, n \in \mathbb{N}$ and $\ell \leqslant n$,
$$r^{(n)}_{j_1 j_2 \cdots j_\ell} \in \mathcal{F}[g_1, \cdots, g_k] .$$
2. For any $y_1, \cdots, y_\ell \in \{0, 1\}$ and pairwise distinct $j_1, \cdots, j_\ell \in \{1, \cdots, n + \ell\}$, where $\ell, n \in \mathbb{N}$,
$$p^{(n)}_{y_1, \cdots, y_\ell ; j_1, \cdots, j_\ell} \in \mathcal{F}[g_1, \cdots, g_k] .$$
3. For any $1 \leqslant j \leqslant k$, $g_j \in \mathcal{F}[g_1, \cdots, g_k]$.

§1.2 Universal Gates

Definition

Let $\{g_1, g_2, \cdots, g_k\}$ be a collection of classical gates. The collection of all gates that can be constructed from g_1, g_2, $\cdots$, g_k, denoted by $\mathcal{F}[g_1, \cdots, g_k]$, is the set satisfying the following five construction rules:

1. For any pairwise distinct $j_1, \cdots, j_\ell \in \{1, \cdots, \ell\}$, where $\ell, n \in \mathbb{N}$ and $\ell \leqslant n$,
$$r^{(n)}_{j_1 j_2 \cdots j_\ell} \in \mathcal{F}[g_1, \cdots, g_k] .$$
2. For any $y_1, \cdots, y_\ell \in \{0, 1\}$ and pairwise distinct $j_1, \cdots, j_\ell \in \{1, \cdots, n + \ell\}$, where $\ell, n \in \mathbb{N}$,
$$p^{(n)}_{y_1, \cdots, y_\ell ; j_1, \cdots, j_\ell} \in \mathcal{F}[g_1, \cdots, g_k] .$$
3. For any $1 \leqslant j \leqslant k$, $g_j \in \mathcal{F}[g_1, \cdots, g_k]$.

§1.2 Universal Gates

Definition (Cont.)

4. Compositions of elements of $\mathcal{F}[g_1, \cdots, g_k]$ belong to $\mathcal{F}[g_1, \cdots, g_k]$; that is, for any $h_1 : \{0,1\}^n \to \{0,1\}^m$ and $h_2 : \{0,1\}^\ell \to \{0,1\}^n$, we have

$$h_1, h_2 \in \mathcal{F}[g_1, ..., g_k] \quad \Rightarrow \quad h_1 \circ h_2 \in \mathcal{F}[g_1, ..., g_k] .$$

5. Cartesian products of elements of $\mathcal{F}[g_1, \cdots, g_k]$ belong to $\mathcal{F}[g_1, \cdots, g_k]$; that is, for any $h = (h_1, \cdots, h_m) : \{0,1\}^n \to \{0,1\}^m$ and $k = (k_1, \cdots, k_q) : \{0,1\}^p \to \{0,1\}^q$, we have

$$h, k \in \mathcal{F}[g_1, \cdots, g_k] \quad \Rightarrow \quad h \times k \in \mathcal{F}[g_1, \cdots, g_k] ,$$

where with $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ denoting vectors $(x_1, \cdots, x_n)$ and $(x_{n+1}, \cdots, x_{n+p})$, respectively, $h \times k : \{0,1\}^{n+p} \to \{0,1\}^{m+q}$ is the Cartesian product of h and k defined by

$$(h \times k)(\boldsymbol{x}_1, \boldsymbol{x}_2) = \big(h_1(\boldsymbol{x}_1), \cdots, h_m(\boldsymbol{x}_1), k_1(\boldsymbol{x}_2), \cdots, k_q(\boldsymbol{x}_2)\big) .$$

§1.2 Universal Gates

Definition (Cont.)

4. Compositions of elements of $\mathcal{F}[g_1, \cdots, g_k]$ belong to $\mathcal{F}[g_1, \cdots, g_k]$; that is, for any $h_1 : \{0,1\}^n \to \{0,1\}^m$ and $h_2 : \{0,1\}^\ell \to \{0,1\}^n$, we have

$$h_1, h_2 \in \mathcal{F}[g_1, ..., g_k] \quad \Rightarrow \quad h_1 \circ h_2 \in \mathcal{F}[g_1, ..., g_k] .$$

5. Cartesian products of elements of $\mathcal{F}[g_1, \cdots, g_k]$ belong to $\mathcal{F}[g_1, \cdots, g_k]$; that is, for any $h = (h_1, \cdots, h_m) : \{0,1\}^n \to \{0,1\}^m$ and $k = (k_1, \cdots, k_q) : \{0,1\}^p \to \{0,1\}^q$, we have

$$h, k \in \mathcal{F}[g_1, \cdots, g_k] \quad \Rightarrow \quad h \times k \in \mathcal{F}[g_1, \cdots, g_k] ,$$

where with $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ denoting vectors $(x_1, \cdots, x_n)$ and $(x_{n+1}, \cdots, x_{n+p})$, respectively, $h \times k : \{0,1\}^{n+p} \to \{0,1\}^{m+q}$ is the Cartesian product of h and k defined by

$$(h \times k)(\boldsymbol{x}_1, \boldsymbol{x}_2) = \big(h_1(\boldsymbol{x}_1), \cdots, h_m(\boldsymbol{x}_1), k_1(\boldsymbol{x}_2), \cdots, k_q(\boldsymbol{x}_2)\big) .$$

§1.2 Universal Gates

Example

Let $\mathbf{ID}$ be classical gates given by

$$\mathbf{ID}(a) = a \qquad \forall\, a \in \{0,1\}\,.$$

Then $\mathbf{ID}(a) = \mathbf{AND}(a,1) = (\mathbf{AND} \circ p^{(1)}_{1;2})(a)$ which implies that

$$\mathbf{ID} = \mathbf{AND} \circ p^{(1)}_{1;2}\,.$$

Therefore, $\mathbf{ID} \in \mathcal{F}[\mathbf{AND}]$.

Example

For $n \in \mathbb{N}$, let $\mathbf{COPY}^{(n)}$ be a classical gate given by $\mathbf{COPY}^{(n)}(\boldsymbol{a}) = (\boldsymbol{a}, \boldsymbol{a})$ for $\boldsymbol{a} \in \{0,1\}^n$. Using the identity

$$\mathbf{COPY}^{(n)} = r^{(2n)}_{1,3,\cdots,2n-1,2,4,\cdots,2n} \circ \underbrace{\mathbf{COPY}^{(1)} \times \cdots \times \mathbf{COPY}^{(1)}}_{n \text{ copies of } \mathbf{COPY}^{(1)}};$$

we find that $\mathbf{COPY}^{(n)} \in \mathcal{F}[\mathbf{COPY}^{(1)}]$ for all $n \in \mathbb{N}$.

§1.2 Universal Gates

Example

Let $\mathbf{ID}$ be classical gates given by

$$\mathbf{ID}(a) = a \qquad \forall\, a \in \{0,1\}\,.$$

Then $\mathbf{ID}(a) = \mathbf{AND}(a,1) = (\mathbf{AND} \circ p^{(1)}_{1;2})(a)$ which implies that

$$\mathbf{ID} = \mathbf{AND} \circ p^{(1)}_{1;2}\,.$$

Therefore, $\mathbf{ID} \in \mathcal{F}[\mathbf{AND}]$.

Example

For $n \in \mathbb{N}$, let $\mathbf{COPY}^{(n)}$ be a classical gate given by $\mathbf{COPY}^{(n)}(\boldsymbol{a}) = (\boldsymbol{a}, \boldsymbol{a})$ for $\boldsymbol{a} \in \{0,1\}^n$. Using the identity

$$\mathbf{COPY}^{(n)} = r^{(2n)}_{1,3,\cdots,2n-1,2,4,\cdots,2n} \circ \underbrace{\mathbf{COPY}^{(1)} \times \cdots \times \mathbf{COPY}^{(1)}}_{n \text{ copies of } \mathbf{COPY}^{(1)}};$$

we find that $\mathbf{COPY}^{(n)} \in \mathcal{F}[\mathbf{COPY}^{(1)}]$ for all $n \in \mathbb{N}$.

§1.2 Universal Gates

Example

For $a, b, c \in \{0, 1\}$,

$$\begin{aligned}
&\big((\mathbf{ID} \times \mathbf{ID} \times \mathbf{XOR}) \circ (\mathbf{ID} \times \mathbf{ID} \times \mathbf{AND} \times \mathbf{ID}) \circ r^{(5)}_{1,3,2,4,5} \cdots \\
&\qquad \cdots \circ (\mathbf{COPY}^{(1)} \times \mathbf{COPY}^{(1)} \times \mathbf{ID})\big)(a, b, c) \\
&= \big((\mathbf{ID} \times \mathbf{ID} \times \mathbf{XOR}) \circ (\mathbf{ID} \times \mathbf{ID} \times \mathbf{AND} \times \mathbf{ID}) \circ r^{(5)}_{1,3,2,4,5}\big)(a, a, b, b, c) \\
&= \big((\mathbf{ID} \times \mathbf{ID} \times \mathbf{XOR}) \circ (\mathbf{ID} \times \mathbf{ID} \times \mathbf{AND} \times \mathbf{ID})\big)(a, b, a, b, c) \\
&= (\mathbf{ID} \times \mathbf{ID} \times \mathbf{XOR})(a, b, \mathbf{AND}(a, b), c) = (a, b, ab \oplus c) \\
&= \mathbf{TOF}(a, b, c)\,.
\end{aligned}$$

Therefore, $\mathbf{TOF} \in \mathcal{F}\big[\mathbf{ID}, \mathbf{XOR}, \mathbf{AND}, \mathbf{COPY}^{(1)}\big]$, and the previous example further shows that $\mathbf{TOF} \in \mathcal{F}\big[\mathbf{XOR}, \mathbf{AND}, \mathbf{COPY}^{(1)}\big]$

§1.2 Universal Gates

Example

In this example we show that **AND**, **OR** and **NOT** can be constructed solely by **NAND** or **NOR**. The **AND** and **OR** gates can be implemented using **NAND** or **NOR** by the following logic circuit:

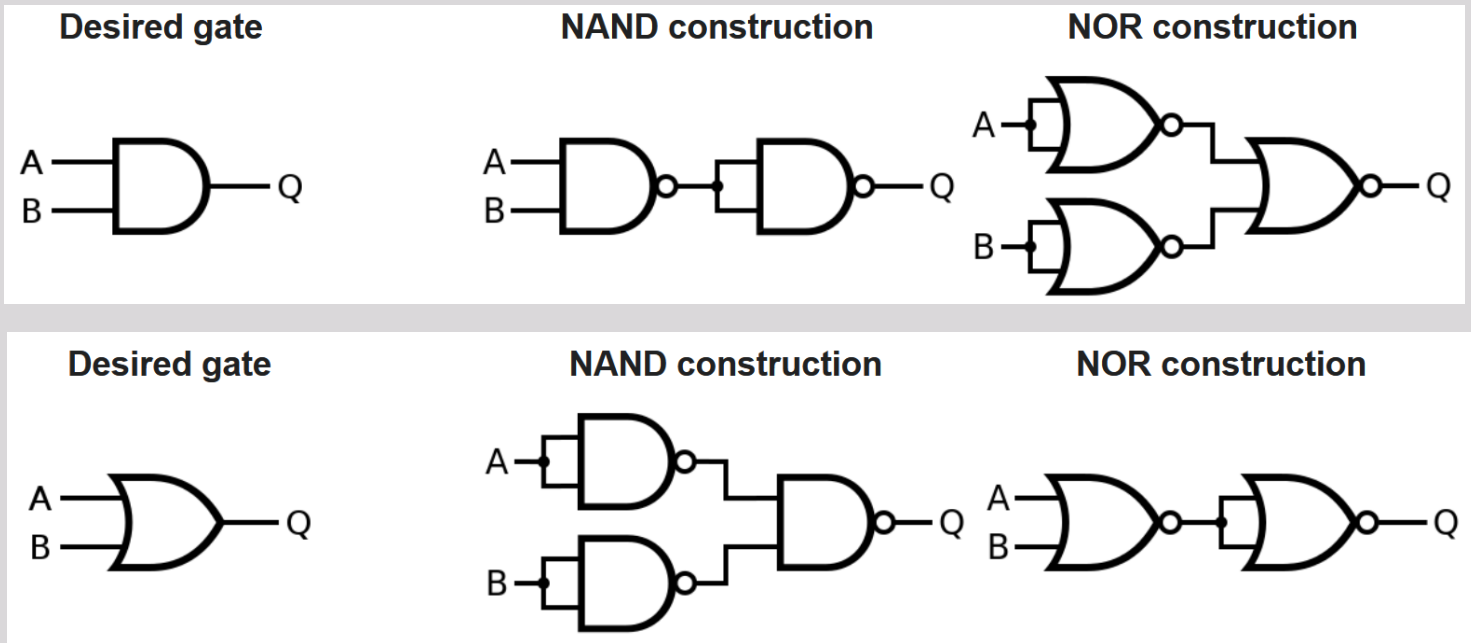


Figure 6: The construction of **AND** and **OR** from **NAND** or **NOR**

§1.2 Universal Gates

Example (cont.)

and the **NOT** gate can be constructed by **NAND** or **NOR** by the following logic circuit:

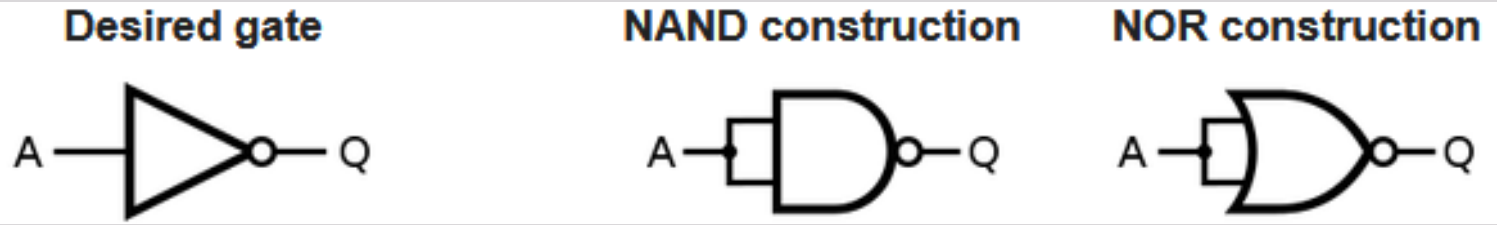


Figure 7: The construction of **NOT** from **NAND** or **NOR**

Therefore, $\mathbf{AND}, \mathbf{OR}, \mathbf{NOT} \in \mathcal{F}[\mathbf{NAND}] \cap \mathcal{F}[\mathbf{NOR}]$.

On the other hand, $\mathbf{NAND}, \mathbf{NOR} \in \mathcal{F}[\mathbf{AND}, \mathbf{OR}, \mathbf{NOT}]$ since

$$\mathbf{NAND} = \mathbf{NOT} \circ \mathbf{AND} \quad \text{and} \quad \mathbf{NOR} = \mathbf{NOT} \circ \mathbf{OR}.$$

As a consequence,

$$\mathcal{F}[\mathbf{AND}, \mathbf{OR}, \mathbf{NOT}] = \mathcal{F}[\mathbf{NAND}] = \mathcal{F}[\mathbf{NOR}].$$

§1.2 Universal Gates

Example (cont.)

and the **NOT** gate can be constructed by **NAND** or **NOR** by the following logic circuit:

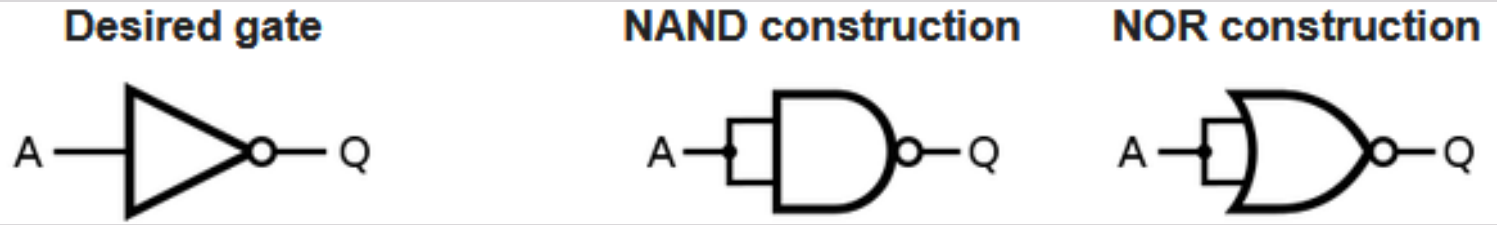


Figure 7: The construction of **NOT** from **NAND** or **NOR**

Therefore, $\mathbf{AND}, \mathbf{OR}, \mathbf{NOT} \in \mathcal{F}[\mathbf{NAND}] \cap \mathcal{F}[\mathbf{NOR}]$.

On the other hand, $\mathbf{NAND}, \mathbf{NOR} \in \mathcal{F}[\mathbf{AND}, \mathbf{OR}, \mathbf{NOT}]$ since

$$\mathbf{NAND} = \mathbf{NOT} \circ \mathbf{AND} \qquad \text{and} \qquad \mathbf{NOR} = \mathbf{NOT} \circ \mathbf{OR}\,.$$

As a consequence,

$$\mathcal{F}[\mathbf{AND}, \mathbf{OR}, \mathbf{NOT}] = \mathcal{F}[\mathbf{NAND}] = \mathcal{F}[\mathbf{NOR}]\,.$$

§1.2 Universal Gates

Remark:

1. In the construction of basic logic gate using **NAND** or **NOR**, the gate $\mathbf{COPY}^{(1)}$ is used implicitly. In other words, to be more precise we should write
$$\mathcal{F}[\mathbf{NAND}] \subseteq \mathcal{F}[\mathbf{AND}, \mathbf{OR}, \mathbf{NOT}] \subseteq \mathcal{F}[\mathbf{NAND}, \mathbf{COPY}^{(1)}],$$
$$\mathcal{F}[\mathbf{NOR}] \subseteq \mathcal{F}[\mathbf{AND}, \mathbf{OR}, \mathbf{NOT}] \subseteq \mathcal{F}[\mathbf{NOR}, \mathbf{COPY}^{(1)}].$$
2. A logic gate in $\mathcal{F}[\mathbf{AND}, \mathbf{OR}, \mathbf{NOT}]$ is called a Boolean circuit.
3. The **XOR** and **XNOR** gates can be constructed from **NAND** or **NOR** gates (with the help of $\mathbf{COPY}^{(1)}$). See the lecture note for the detail circuits.

Proposition

Let $\{g_1, \cdots, g_k\}$ be a collection of classical gates, and $h_1, \cdots, h_\ell \in \mathcal{F}[g_1, \cdots, g_k]$. Then $\mathcal{F}[h_1, \cdots, h_\ell] \subseteq \mathcal{F}[g_1, \cdots, g_k]$.

§1.2 Universal Gates

Remark:

1. In the construction of basic logic gate using **NAND** or **NOR**, the gate $\mathbf{COPY}^{(1)}$ is used implicitly. In other words, to be more precise we should write
$$\mathcal{F}[\mathbf{NAND}] \subseteq \mathcal{F}[\mathbf{AND}, \mathbf{OR}, \mathbf{NOT}] \subseteq \mathcal{F}[\mathbf{NAND}, \mathbf{COPY}^{(1)}],$$
$$\mathcal{F}[\mathbf{NOR}] \subseteq \mathcal{F}[\mathbf{AND}, \mathbf{OR}, \mathbf{NOT}] \subseteq \mathcal{F}[\mathbf{NOR}, \mathbf{COPY}^{(1)}].$$
2. A logic gate in $\mathcal{F}[\mathbf{AND}, \mathbf{OR}, \mathbf{NOT}]$ is called a Boolean circuit.
3. The **XOR** and **XNOR** gates can be constructed from **NAND** or **NOR** gates (with the help of $\mathbf{COPY}^{(1)}$). See the lecture note for the detail circuits.

Proposition

Let $\{g_1, \cdots, g_k\}$ be a collection of classical gates, and $h_1, \cdots, h_\ell \in \mathcal{F}[g_1, \cdots, g_k]$. Then $\mathcal{F}[h_1, \cdots, h_\ell] \subseteq \mathcal{F}[g_1, \cdots, g_k]$.

§1.2 Universal Gates

Definition

A collection $\{g_1, \cdots, g_k\}$ of classical gates is said to be **universal** if $g \in \mathcal{F}[g_1, \cdots, g_k]$ for every classical gate g.

Theorem

The classical TOFFOLI-gate is universal and reversible.

Proof of the universality of the Toffoli gate.

We have known that **TOF** is reversible, so it suffices to show the universality of **TOF**.

Since every gate $g : \{0,1\}^n \to \{0,1\}^m$ is a Cartesian product of m gates $g_1, g_2, \cdots, g_m : \{0,1\}^n \to \{0,1\}$, it suffices to show the universality only for a gate of the form $f : \{0,1\}^n \to \{0,1\}$, which we shall do by induction in n. □

§1.2 Universal Gates

Definition

A collection $\{g_1, \cdots, g_k\}$ of classical gates is said to be **universal** if $g \in \mathcal{F}[g_1, \cdots, g_k]$ for every classical gate g.

Theorem

The classical TOFFOLI-gate is universal and reversible.

Proof of the universality of the Toffoli gate.

We have known that **TOF** is reversible, so it suffices to show the universality of **TOF**.

Since every gate $g : \{0,1\}^n \to \{0,1\}^m$ is a Cartesian product of m gates $g_1, g_2, \cdots, g_m : \{0,1\}^n \to \{0,1\}$, it suffices to show the universality only for a gate of the form $f : \{0,1\}^n \to \{0,1\}$, which we shall do by induction in n. □

§1.2 Universal Gates

Definition

A collection $\{g_1, \cdots, g_k\}$ of classical gates is said to be **universal** if $g \in \mathcal{F}[g_1, \cdots, g_k]$ for every classical gate g.

Theorem

The classical TOFFOLI-gate is universal and reversible.

Proof of the universality of the Toffoli gate.

We have known that **TOF** is reversible, so it suffices to show the universality of **TOF**.

Since every gate $g : \{0,1\}^n \to \{0,1\}^m$ is a Cartesian product of m gates $g_1, g_2, \cdots, g_m : \{0,1\}^n \to \{0,1\}$, it suffices to show the universality only for a gate of the form $f : \{0,1\}^n \to \{0,1\}$, which we shall do by induction in n. □

§1.2 Universal Gates

Definition

A collection $\{g_1, \cdots, g_k\}$ of classical gates is said to be **universal** if $g \in \mathcal{F}[g_1, \cdots, g_k]$ for every classical gate g.

Theorem

The classical TOFFOLI-gate is universal and reversible.

Proof of the universality of the Toffoli gate.

We have known that **TOF** is reversible, so it suffices to show the universality of **TOF**.

Since every gate $g : \{0,1\}^n \to \{0,1\}^m$ is a Cartesian product of m gates $g_1, g_2, \cdots, g_m : \{0,1\}^n \to \{0,1\}$, it suffices to show the universality only for a gate of the form $f : \{0,1\}^n \to \{0,1\}$, which we shall do by induction in n. □

§1.2 Universal Gates

Proof of the universality of the Toffoli gate (cont.)

Before initiating the induction argument, let us first construct the **AND**, **XOR** and $\mathbf{COPY}^{(n)}$ gates using the Toffoli gate. Since

$$\begin{aligned} \textsc{Tof}(a,b,0) &= (a,b,ab)\,, \\ \textsc{Tof}(1,a,b) &= (1,a,a\oplus b)\,, \end{aligned} \qquad \forall\, a,b\in\{0,1\}\,,$$

we find that

$$\begin{aligned} \mathbf{AND}(a,b) &= ab = (r_3^{(3)}\circ \textsc{Tof})(a,b,0) = (r_3^{(3)}\circ \textsc{Tof}\circ p_{0;3}^{(2)})(a,b)\,, \\ \mathbf{XOR}(a,b) &= a\oplus b = (r_3^{(3)}\circ \textsc{Tof})(1,a,b) = (r_3^{(3)}\circ \textsc{Tof}\circ p_{1;1}^{(2)})(a,b)\,, \\ \mathbf{COPY}^{(1)}(a) &= (r_{1,3}^{(3)}\circ \textsc{Tof})(a,1,0) = (r_{1,3}^{(3)}\circ \textsc{Tof}\circ p_{1,0;2,3}^{(1)})(a)\,. \end{aligned}$$

Therefore, $\mathbf{AND}, \mathbf{XOR}, \mathbf{COPY}^{(1)} \in \mathcal{F}[\textsc{Tof}]$. Together with the fact that $\mathbf{COPY}^{(n)} \in \mathcal{F}[\mathbf{COPY}^{(1)}]$, we also conclude that $\mathbf{COPY}^{(n)} \in \mathcal{F}[\textsc{Tof}]$ for all $n\in\mathbb{N}$. □

§1.2 Universal Gates

Proof of the universality of the Toffoli gate (cont.)

Before initiating the induction argument, let us first construct the **AND**, **XOR** and $\mathbf{COPY}^{(n)}$ gates using the Toffoli gate. Since

$$\begin{aligned} \mathbf{TOF}(a,b,0) &= (a,b,ab)\,, \\ \mathbf{TOF}(1,a,b) &= (1,a,a\oplus b)\,, \end{aligned} \qquad \forall\, a,b\in\{0,1\}\,,$$

we find that

$$\begin{aligned} \mathbf{AND}(a,b) &= ab = (r_3^{(3)}\circ\mathbf{TOF})(a,b,0) = (r_3^{(3)}\circ\mathbf{TOF}\circ p_{0;3}^{(2)})(a,b)\,, \\ \mathbf{XOR}(a,b) &= a\oplus b = (r_3^{(3)}\circ\mathbf{TOF})(1,a,b) = (r_3^{(3)}\circ\mathbf{TOF}\circ p_{1;1}^{(2)})(a,b)\,, \\ \mathbf{COPY}^{(1)}(a) &= (r_{1,3}^{(3)}\circ\mathbf{TOF})(a,1,0) = (r_{1,3}^{(3)}\circ\mathbf{TOF}\circ p_{1,0;2,3}^{(1)})(a)\,. \end{aligned}$$

Therefore, $\mathbf{AND},\mathbf{XOR},\mathbf{COPY}^{(1)}\in\mathcal{F}[\mathbf{TOF}]$. Together with the fact that $\mathbf{COPY}^{(n)}\in\mathcal{F}[\mathbf{COPY}^{(1)}]$, we also conclude that $\mathbf{COPY}^{(n)}\in\mathcal{F}[\mathbf{TOF}]$ for all $n\in\mathbb{N}$. □

§1.2 Universal Gates

Proof of the universality of the Toffoli gate (cont.)

Before initiating the induction argument, let us first construct the **AND**, **XOR** and $\mathbf{COPY}^{(n)}$ gates using the Toffoli gate. Since

$$\begin{aligned} \textbf{TOF}(a,b,0) &= (a,b,ab)\,, \\ \textbf{TOF}(1,a,b) &= (1,a,a\oplus b)\,, \end{aligned} \qquad \forall\, a,b\in\{0,1\}\,,$$

we find that

$$\begin{aligned} \mathbf{AND}(a,b) &= ab = (r_3^{(3)}\circ\textbf{TOF})(a,b,0) = (r_3^{(3)}\circ\textbf{TOF}\circ p_{0;3}^{(2)})(a,b)\,, \\ \mathbf{XOR}(a,b) &= a\oplus b = (r_3^{(3)}\circ\textbf{TOF})(1,a,b) = (r_3^{(3)}\circ\textbf{TOF}\circ p_{1;1}^{(2)})(a,b)\,, \\ \mathbf{COPY}^{(1)}(a) &= (r_{1,3}^{(3)}\circ\textbf{TOF})(a,1,0) = (r_{1,3}^{(3)}\circ\textbf{TOF}\circ p_{1,0;2,3}^{(1)})(a)\,. \end{aligned}$$

Therefore, $\mathbf{AND},\mathbf{XOR},\mathbf{COPY}^{(1)}\in\mathcal{F}[\textbf{TOF}]$. Together with the fact that $\mathbf{COPY}^{(n)}\in\mathcal{F}[\mathbf{COPY}^{(1)}]$, we also conclude that $\mathbf{COPY}^{(n)}\in\mathcal{F}[\textbf{TOF}]$ for all $n\in\mathbb{N}$. □

§1.2 Universal Gates

Proof of the universality of the Toffoli gate (cont.)

Now we initiate the induction process. First we need to show that **TOF** is universal for gates of the form $f : \{0,1\} \to \{0,1\}$. There are four gates in this case: the identity gate **ID**, the **NOT** gate, the **TRUE** gate whose output is always 1, and the **FALSE** gate whose output is always 0. Note that for $a \in \{0,1\}$,

$$\mathbf{TOF}(1,0,a) = (1,0,a) \quad \text{and} \quad \mathbf{TOF}(1,1,a) = (1,1,1 \oplus a)\,.$$

Using the identity $p^{(1)}_{1,0;1,2}(a) = (1,0,a)$, we find that

$$\mathbf{ID}(a) = (r^{(3)}_3 \circ \mathbf{TOF})(1,0,a) = (r^{(3)}_3 \circ \mathbf{TOF} \circ p^{(1)}_{1,0;1,2})(a)\,,$$

$$\mathbf{TRUE}(a) = (r^{(3)}_1 \circ \mathbf{TOF})(1,0,a) = (r^{(3)}_1 \circ \mathbf{TOF} \circ p^{(1)}_{1,0;1,2})(a)\,,$$

$$\mathbf{FALSE}(a) = (r^{(3)}_2 \circ \mathbf{TOF})(1,0,a) = (r^{(3)}_2 \circ \mathbf{TOF} \circ p^{(1)}_{1,0;1,2})(a)\,,$$

$$\mathbf{NOT}(a) = (r^{(3)}_3 \circ \mathbf{TOF})(1,1,a) = (r^{(3)}_3 \circ \mathbf{TOF} \circ p^{(1)}_{1,1;1,2})(a)\,. \qquad \square$$

§1.2 Universal Gates

Proof of the universality of the Toffoli gate (cont.)

Now we initiate the induction process. First we need to show that **TOF** is universal for gates of the form $f : \{0,1\} \to \{0,1\}$. There are four gates in this case: the identity gate **ID**, the **NOT** gate, the **TRUE** gate whose output is always 1, and the **FALSE** gate whose output is always 0. Note that for $a \in \{0,1\}$,

$$\mathbf{TOF}(1,0,a) = (1,0,a) \quad \text{and} \quad \mathbf{TOF}(1,1,a) = (1,1,1 \oplus a)\,.$$

Using the identity $p^{(1)}_{1,0;1,2}(a) = (1,0,a)$, we find that

$$\mathbf{ID}(a) = (r^{(3)}_3 \circ \mathbf{TOF})(1,0,a) = (r^{(3)}_3 \circ \mathbf{TOF} \circ p^{(1)}_{1,0;1,2})(a)\,,$$
$$\mathbf{TRUE}(a) = (r^{(3)}_1 \circ \mathbf{TOF})(1,0,a) = (r^{(3)}_1 \circ \mathbf{TOF} \circ p^{(1)}_{1,0;1,2})(a)\,,$$
$$\mathbf{FALSE}(a) = (r^{(3)}_2 \circ \mathbf{TOF})(1,0,a) = (r^{(3)}_2 \circ \mathbf{TOF} \circ p^{(1)}_{1,0;1,2})(a)\,,$$
$$\mathbf{NOT}(a) = (r^{(3)}_3 \circ \mathbf{TOF})(1,1,a) = (r^{(3)}_3 \circ \mathbf{TOF} \circ p^{(1)}_{1,1;1,2})(a)\,. \qquad \square$$

§1.2 Universal Gates

Proof of the universality of the Toffoli gate (cont.)

Therefore, **TOF** is universal for gates of the form $f\colon \{0,1\} \to \{0,1\}$.

Suppose that **TOF** is universal for gates of the form $f\colon \{0,1\}^{n-1} \to \{0,1\}$. Let $f : \{0,1\}^n \to \{0,1\}$ be a classical gate. Define classical gates $g_0, g_1 : \{0,1\}^{n-1} \to \{0,1\}$ by

$$g_0(x_1, \cdots, x_{n-1}) = f(x_1, \cdots, x_{n-1}, 0)\,,$$
$$g_1(x_1, \cdots, x_{n-1}) = f(x_1, \cdots, x_{n-1}, 1)\,,$$

and $h : \{0,1\}^n \to \{0,1\}$ by

$$h(x_1, \cdots, x_n) = \mathbf{XOR}\big(\mathbf{AND}(g_0(x_1, \cdots, x_{n-1}), \mathbf{NOT}(x_n)), \mathbf{AND}(g_1(x_1, \cdots, x_{n-1}), x_n)\big)\,.$$

Next, we show that $h = f$ and use this fact to establish that $f \in \mathcal{F}[\mathbf{TOF}]$. The theorem is then concluded by induction. □

§1.2 Universal Gates

Proof of the universality of the Toffoli gate (cont.)

Therefore, **TOF** is universal for gates of the form $f\colon \{0,1\} \to \{0,1\}$. Suppose that **TOF** is universal for gates of the form $f\colon \{0,1\}^{n-1} \to \{0,1\}$. Let $f\colon \{0,1\}^n \to \{0,1\}$ be a classical gate. Define classical gates $g_0, g_1 : \{0,1\}^{n-1} \to \{0,1\}$ by

$$g_0(x_1, \cdots, x_{n-1}) = f(x_1, \cdots, x_{n-1}, 0),$$
$$g_1(x_1, \cdots, x_{n-1}) = f(x_1, \cdots, x_{n-1}, 1),$$

and $h : \{0,1\}^n \to \{0,1\}$ by

$$h(x_1, \cdots, x_n) = \mathbf{XOR}\big(\mathbf{AND}(g_0(x_1, \cdots, x_{n-1}), \mathbf{NOT}(x_n)), \mathbf{AND}(g_1(x_1, \cdots, x_{n-1}), x_n)\big).$$

Next, we show that $h = f$ and use this fact to establish that $f \in \mathcal{F}[\mathbf{TOF}]$. The theorem is then concluded by induction. □

§1.2 Universal Gates

Proof of the universality of the Toffoli gate (cont.)

Therefore, **TOF** is universal for gates of the form $f\colon \{0,1\} \to \{0,1\}$. Suppose that **TOF** is universal for gates of the form $f\colon \{0,1\}^{n-1} \to \{0,1\}$. Let $f\colon \{0,1\}^n \to \{0,1\}$ be a classical gate. Define classical gates $g_0, g_1 : \{0,1\}^{n-1} \to \{0,1\}$ by

$$g_0(x_1, \cdots, x_{n-1}) = f(x_1, \cdots, x_{n-1}, 0)\,,$$
$$g_1(x_1, \cdots, x_{n-1}) = f(x_1, \cdots, x_{n-1}, 1)\,,$$

and $h : \{0,1\}^n \to \{0,1\}$ by

$$h(x_1, \cdots, x_n) = \mathbf{XOR}\big(\mathbf{AND}(g_0(x_1, \cdots, x_{n-1}), \mathbf{NOT}(x_n)), \mathbf{AND}(g_1(x_1, \cdots, x_{n-1}), x_n)\big)\,.$$

Next, we show that $h = f$ and use this fact to establish that $f \in \mathcal{F}[\mathbf{TOF}]$. The theorem is then concluded by induction. □

§1.2 Universal Gates

Proof of the universality of the Toffoli gate (cont.)

Therefore, **TOF** is universal for gates of the form $f\colon \{0,1\} \to \{0,1\}$. Suppose that **TOF** is universal for gates of the form $f\colon \{0,1\}^{n-1} \to \{0,1\}$. Let $f\colon \{0,1\}^n \to \{0,1\}$ be a classical gate. Define classical gates $g_0, g_1 : \{0,1\}^{n-1} \to \{0,1\}$ by

$$g_0(x_1, \cdots, x_{n-1}) = f(x_1, \cdots, x_{n-1}, 0)\,,$$
$$g_1(x_1, \cdots, x_{n-1}) = f(x_1, \cdots, x_{n-1}, 1)\,,$$

and $h : \{0,1\}^n \to \{0,1\}$ by

$$h(x_1, \cdots, x_n) = \mathbf{XOR}\big(\mathbf{AND}(g_0(x_1, \cdots, x_{n-1}), \mathbf{NOT}(x_n)), \mathbf{AND}(g_1(x_1, \cdots, x_{n-1}), x_n)\big)\,.$$

Next, we show that $h = f$ and use this fact to establish that $f \in \mathcal{F}[\mathbf{TOF}]$. The theorem is then concluded by induction. □

§1.2 Universal Gates

Proof of the universality of the Toffoli gate (cont.)

For a fixed $\widehat{x}_n \equiv (x_1, \cdots, x_{n-1}) \in \{0,1\}^{n-1}$, there are four cases:

1. $g_0(\widehat{x}_n) = g_1(\widehat{x}_n) = 0$: in this case

$$\begin{aligned} h(x_1, \cdots, x_n) &= \mathbf{XOR}\big(\mathbf{AND}(0, \mathbf{NOT}(x_n)), \mathbf{AND}(0, x_n)\big) = 0 \\ &= f(x_1, \cdots, x_n)\,. \end{aligned}$$

2. $g_0(\widehat{x}_n) = g_1(\widehat{x}_n) = 1$: in this case

$$\begin{aligned} h(x_1, \cdots, x_n) &= \mathbf{XOR}\big(\mathbf{AND}(1, \mathbf{NOT}(x_n)), \mathbf{AND}(1, x_n)\big) = 1 \\ &= f(x_1, \cdots, x_n)\,. \end{aligned}$$

3. $g_0(\widehat{x}_n) = 0$ and $g_1(\widehat{x}_n) = 1$: in this case,

$$\begin{aligned} h(x_1, \cdots, x_n) &= \mathbf{XOR}\big(\mathbf{AND}(0, \mathbf{NOT}(x_n)), \mathbf{AND}(1, x_n)\big) \\ &= \mathbf{ID}(x_n) = f(x_1, \cdots, x_n)\,. \end{aligned}$$

□

§1.2 Universal Gates

Proof of the universality of the Toffoli gate (cont.)

4. $g_0(\widehat{x}_n) = 1$ and $g_1(\widehat{x}_n) = 0$: in this case,

$$\begin{aligned} h(x_1, \cdots, x_n) &= \mathbf{XOR}\big(\mathbf{AND}(1, \mathbf{NOT}(x_n)), \mathbf{AND}(0, x_n)\big) \\ &= \mathbf{NOT}(x_n) = f(x_1, \cdots, x_n)\,. \end{aligned}$$

Therefore, $h = f$. By the induction assumption, $g_0, g_1 \in \mathcal{F}[\textbf{TOF}]$; thus the identity

$$h = \mathbf{XOR} \circ (\mathbf{AND} \times \mathbf{AND}) \circ \big((g_0 \times \mathbf{NOT}) \times (g_1 \times \mathbf{ID})\big) \circ \mathbf{COPY}^{(n)}$$

and the fact that $\mathbf{XOR}, \mathbf{AND}, \mathbf{NOT}, \mathbf{ID}, \mathbf{COPY}^{(n)} \in \mathcal{F}[\textbf{TOF}]$ show that $f \in \mathcal{F}[\textbf{TOF}]$. □

Remark: Since **XOR** can be constructed using **NAND**, previous example shows that $\textbf{TOF} \in \mathcal{F}\big[\mathbf{NAND}, \mathbf{COPY}^{(1)}\big]$. Therefore,

$$\mathcal{F}[\textbf{TOF}] = \mathcal{F}\big[\mathbf{NAND}, \mathbf{COPY}^{(1)}\big] \supseteq \mathcal{F}[\mathbf{AND}, \mathbf{OR}, \mathbf{NOT}]\,.$$

§1.2 Universal Gates

Proof of the universality of the Toffoli gate (cont.)

4. $g_0(\widehat{x}_n) = 1$ and $g_1(\widehat{x}_n) = 0$: in this case,

$$\begin{aligned} h(x_1, \cdots, x_n) &= \mathbf{XOR}\big(\mathbf{AND}(1, \mathbf{NOT}(x_n)), \mathbf{AND}(0, x_n)\big) \\ &= \mathbf{NOT}(x_n) = f(x_1, \cdots, x_n)\,. \end{aligned}$$

Therefore, $h = f$. By the induction assumption, $g_0, g_1 \in \mathcal{F}[\mathbf{TOF}]$; thus the identity

$$h = \mathbf{XOR} \circ (\mathbf{AND} \times \mathbf{AND}) \circ \big((g_0 \times \mathbf{NOT}) \times (g_1 \times \mathbf{ID})\big) \circ \mathbf{COPY}^{(n)}$$

and the fact that $\mathbf{XOR}, \mathbf{AND}, \mathbf{NOT}, \mathbf{ID}, \mathbf{COPY}^{(n)} \in \mathcal{F}[\mathbf{TOF}]$ show that $f \in \mathcal{F}[\mathbf{TOF}]$. □

Remark: Since **XOR** can be constructed using **NAND**, previous example shows that $\mathbf{TOF} \in \mathcal{F}\big[\mathbf{NAND}, \mathbf{COPY}^{(1)}\big]$. Therefore,

$$\mathcal{F}[\mathbf{TOF}] = \mathcal{F}\big[\mathbf{NAND}, \mathbf{COPY}^{(1)}\big] \supseteq \mathcal{F}[\mathbf{AND}, \mathbf{OR}, \mathbf{NOT}]\,.$$

§1.3 How A Classical Computer Adds Numbers

In a (classical) computer, each number is stored as a ***binary number*** which is a number expressed in the base-2 numeral system. In an N-bit system, the first bit is always used to store the sign of the number, and the rest $(N-1)$ bits are used to express the number (we will not go further into the fixed point or floating point system).

Every **non-negative** binary number takes the form $0i_n i_{n-1} i_{n-2} \cdots i_1$ (or more precisely, $(0i_n i_{n-1} \cdots i_1)_2$), where $i_k \in \{0, 1\}$ for each k, and is the same as the number

$$2^{n-1} i_n + 2^{n-2} i_{n-2} + \cdots + 2^1 i_2 + i_1 = \sum_{k=1}^{n} 2^{k-1} i_k$$

in the usual base-10 numeral system. For example, the number 13 in the base-10 numeral system is expressed as $0 \cdots 01101$ in the base-2 numeral system.

§1.3 How A Classical Computer Adds Numbers

In a (classical) computer, each number is stored as a ***binary number*** which is a number expressed in the base-2 numeral system. In an N-bit system, the first bit is always used to store the sign of the number, and the rest $(N-1)$ bits are used to express the number (we will not go further into the fixed point or floating point system).

Every **non-negative** binary number takes the form $0i_n i_{n-1} i_{n-2} \cdots i_1$ (or more precisely, $(0i_n i_{n-1} \cdots i_1)_2$), where $i_k \in \{0, 1\}$ for each k, and is the same as the number

$$2^{n-1} i_n + 2^{n-2} i_{n-2} + \cdots + 2^1 i_2 + i_1 = \sum_{k=1}^{n} 2^{k-1} i_k$$

in the usual base-10 numeral system. For example, the number 13 in the base-10 numeral system is expressed as $0 \cdots 01101$ in the base-2 numeral system.

§1.3 How A Classical Computer Adds Numbers

Every **negative** binary number takes the form $1i_ni_{n-1}i_{n-2}\cdots i_1$ (or more precisely, $(1i_ni_{n-1}\cdots i_1)_2$), where $i_k \in \{0,1\}$ for each k, and is the same as the number

$$-2^{n-1}(1-i_n)-\cdots-2^1(1-i_2)-2^0(1-i_1)-1=-1-\sum_{k=1}^{n}2^{k-1}(1-i_k)$$

in the usual base-10 numeral system (here the two's-complement number system - 二補數系統 - is used). For example, the number -13 is $1\cdots10011$ (which is obtained by exchanging 0 and 1 in the binary expression of 13 and the outcome plus 1 is the binary expression of -13). We also note that the number $1i_ni_{n-1}i_{n-2}\cdots i_1$ is the same as $-2^n+0i_ni_{n-1}\cdots i_1$, where $0i_ni_{n-1}\cdots i_1$ denotes the non-negative number given previously.

§1.3 How A Classical Computer Adds Numbers

Every **negative** binary number takes the form $1i_ni_{n-1}i_{n-2}\cdots i_1$ (or more precisely, $(1i_ni_{n-1}\cdots i_1)_2$), where $i_k \in \{0,1\}$ for each k, and is the same as the number

$$-2^{n-1}(1-i_n)-\cdots-2^1(1-i_2)-2^0(1-i_1)-1=-1-\sum_{k=1}^{n}2^{k-1}(1-i_k)$$

in the usual base-10 numeral system (here the two's-complement number system - 二補數系統 - is used). For example, the number -13 is $1\cdots10011$ (which is obtained by exchanging 0 and 1 in the binary expression of 13 and the outcome plus 1 is the binary expression of -13). We also note that the number $1i_ni_{n-1}i_{n-2}\cdots i_1$ is the same as $-2^n+0i_ni_{n-1}\cdots i_1$, where $0i_ni_{n-1}\cdots i_1$ denotes the non-negative number given previously.

§1.3 How A Classical Computer Adds Numbers

Example

Since 7 in the base-10 numeral system is the same as $0\cdots0111$ is the base-2 numeral system, the classical computers compute $7+13$ and $7-13$ (which is the same as $7+(-13)$) as follows:

$$\begin{aligned} 7+13 &= (0\cdots00111)_2 + (0\cdots01101)_2 \\ &= (0\cdots010100)_2 = 2^4+2^2 = 20\,, \\ 7+(-13) &= (0\cdots00111)_2 + (1\cdots10011)_2 \\ &= (1\cdots111010)_2 = -2^2-2^0-1 = -6\,. \end{aligned}$$

§1.3 How A Classical Computer Adds Numbers

Remark: For a non-negative integer $k = (k_{n-1}k_{n-2}\cdots k_0)_2$, in matlab® k_j is the $(j+1)$-th component of the vector $\boldsymbol{x}$ given by

$$\boldsymbol{x} = \mathbf{de2bi}(k, n)\,.$$

In other words, $\boldsymbol{x}$ given above lists the lowest bit to the highest bit of k from left to right. To obtain the bit expression in exactly the same order, we use the **flip** function so that

$$(k_{n-1}, k_{n-2}, \cdots, k_0) = \mathbf{flip}(\mathbf{de2bi}(k, n))\,.$$

We also remark that in matlab® the input of **de2bi** has to be non-negative integers (so it will not output the bit expression of negative integer in the two's complement number system).

§1.3 How A Classical Computer Adds Numbers

An adder (加法器) is a digital circuit that performs addition of numbers.

• **Half adders (半加法器)**: The half adder adds two single binary digits A and B. It has two outputs, sum (S) and carry $(\mathrm{C}$, 進位$)$. The carry signal represents an overflow into the next digit of a multi-digit addition. The sum of A and B is $2\mathrm{C} + \mathrm{S}$. The truth table for the half adder is:

INPUT		OUTPUT	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

§1.3 How A Classical Computer Adds Numbers

The simplest half-adder design, pictured below,

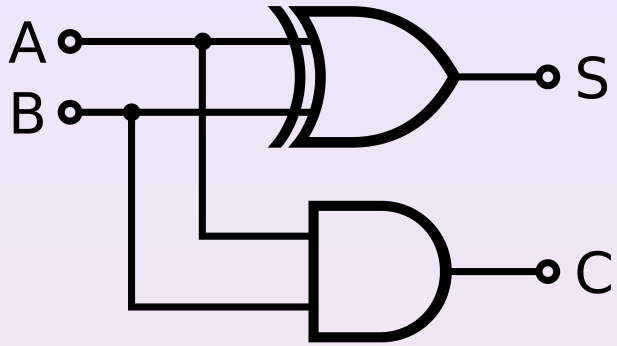


Figure 8: The logic diagram of the half adder

incorporates an **XOR** gate (that gives a true output when the number of true inputs is odd) for S and an **AND** gate for C. The Boolean logic for the sum (in this case S) will be $\mathrm{A}'\mathrm{B}+\mathrm{A}\mathrm{B}'$ (which is $(1-\mathrm{A})\mathrm{B}+\mathrm{A}(1-\mathrm{B})$) whereas for the carry (C) will be AB. The half adder adds two input bits and generates a carry and sum, which are the two outputs of a half adder.

§1.3 How A Classical Computer Adds Numbers

• Full adder (全加法器): A one-bit full-adder adds **three** one-bit numbers, often written as A, B, and $\mathrm{C_{in}}$; A and B are the operands, and $\mathrm{C_{in}}$ is a bit carried in from the previous stage. The circuit produces a two-bit output. Output carry and sum typically represented by the signals $\mathrm{C_{out}}$ and S, where the sum of A and B equals $2\mathrm{C_{out}} + \mathrm{S}$. The truth table for the full adder is:

INPUT			OUTPUT	
A	B	$\mathrm{C_{in}}$	$\mathrm{C_{out}}$	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

§1.3 How A Classical Computer Adds Numbers

A circuit design for the full adder is given below:

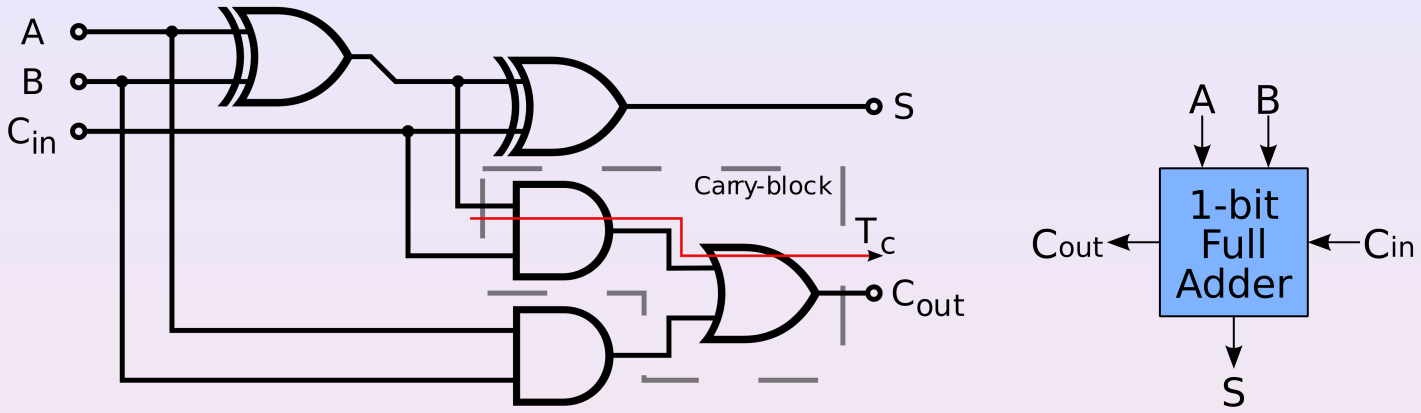
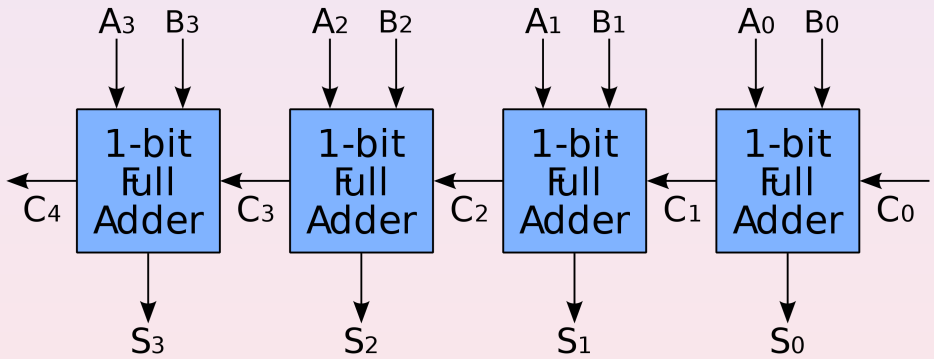


Figure 9: The logic diagram of the full adder (left) and a schematic symbol for a 1-bit full adder (right), here C_{in} and $\mathrm{C}_{\mathrm{out}}$ drawn on sides of block to emphasize their use in a multi-bit adder

§1.3 How A Classical Computer Adds Numbers

We can create a logical circuit using multiple full adders to add N-bit numbers. Each full adder inputs a C_{in}, which is the $\mathrm{C}_{\mathrm{out}}$ of the previous adder. This kind of adder is called a **ripple-carry adder** (RCA), since each carry bit "ripples" to the next full adder. Note that the first (and only the first) full adder may be replaced by a half adder (under the assumption that $\mathrm{C}_{\mathrm{in}} = 0$). The following figure provides a symbol for a 4-bit full adder:



here two input 4-bit numbers is $\mathrm{A} = (\mathrm{A}_3\mathrm{A}_2\mathrm{A}_1\mathrm{A}_0)_2$, $\mathrm{B} = (\mathrm{B}_3\mathrm{B}_2\mathrm{B}_1\mathrm{B}_0)_2$ and the sum of A and B is a 5-bit number $S = (\mathrm{C}_4\mathrm{S}_3\mathrm{S}_2\mathrm{S}_1\mathrm{S}_0)_2$.