Chapter 2
Switching Devices

Logic gates, as introduced in the previous chapter, represent the connectives used in logic formulas. Assume you have drawn a circuit diagram using logic gates, and you now want to build the circuit using switching devices of a given technology (pneumatic, electric, or electronic). To be able to do so you need to know how to translate each logic symbol into a symbol or a collection of symbols of the technology you intend to employ. This chapter is restricted to discussing symbols used for pneumatic valves, electric relays, and CMOS transistors, and to showing how they correspond to logic gates.

In general, any quantifiable physical entity (such as voltage, pressure, force, etc.) can be used as a signal. To be independent of the actual physical entity of any *binary* signal (and its unit of measurement, such as Volt, psi, Newton, etc.) it is convenient and customary to map its two values to the *integers* 0 and 1. If nothing is said to the contrary, the smaller numerical value of a signal is mapped to 0, while the larger numerical value is mapped to 1, and these two integers are not assigned any unit of measurement. In this book the lower case letters x and y (most often with indices) shall always be used as variables whose range is $\{0, 1\}$ (denoted as $x, y \in \{0, 1\}$, and meaning that the value of x and that of y is—in any given moment—either 0 or 1), and referred to as a binary **numeric variables**. These variables, x and y, are used to denote the input and output signals of actual switching devices and their symbols.

2.1 Pneumatic Valves

One of the most common pneumatic valves is the *spool-and-sleeve* design, the principle of which is shown in Fig. 2.1. A *zero* signal refers to atmospheric pressure, a *one* signal to a pressure of say 90 psi (pounds per square inch). If the pilot pressure is 0 the valve is said to be non-actuated and a spring presses the spool in the leftmost position. Actuating the valve by applying a 1-signal to the pilot chamber causes the

S.P. Vingron, *Logic Circuit Design*, DOI 10.1007/978-3-642-27657-6_2,

© Springer-Verlag Berlin Heidelberg 2012

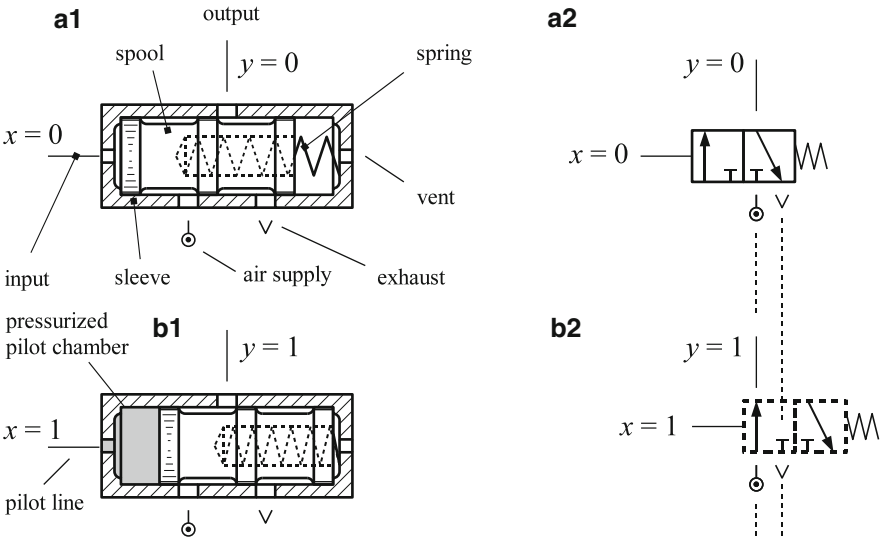

Fig. 2.1 A normally open valve

spool to be pressed to the right (against the force of the spring). In the non-actuated valve you will notice that the output port and the exhaust port are connected while the air supply is blocked. In the actuated valve the output is connected to the air supply and the exhaust is blocked. Let us refer to this valve as a **normally open valve** (or **NO-valve**, for short), this naming convention being borrowed from the realm of electric relays. *Normal* refers to the pilot pressure x being 0 (the valve is not actuated), whereas *open* refers to the path between the air supply and the output being *interrupted*. The standard symbol for the NO-valve is that of Fig. 2.1a2. The rectangle to which the spring is attached (in our case, the right rectangle) shows the port connections in the non-actuated (the *normal*) case, the left rectangle depicts the port connections in the actuated case.

I took the liberty of drawing the *controlled lines*—exhaust, air supply, and output—as being detached from the symbol's rectangles to better show how the symbol is to be understood in the actuated case: Think of the controlled lines as being static and the rectangles being pushed to the right by the pilot pressure, as indicated in Fig. 2.1b2. When the valve is not actuated, the spring pushes the rectangles back into the position shown in Fig. 2.1a2. In actual use, the controlled lines are always drawn attached to the rectangle (the one with the spring), and *the symbol is always drawn in the* non-actuated *mode*. It is *never* drawn in the actuated mode indicated in Fig. 2.1b2. By **convention**, all device symbols in a circuit are drawn in their **normal state**—i.e., their non-actuated mode—which means that they are drawn as if their input signals were zero.

An alternate behaviour to the NO-valve is obtained if the air supply and the exhaust are interchanged as shown in Fig. 2.2a. When the valve is *not* actuated the

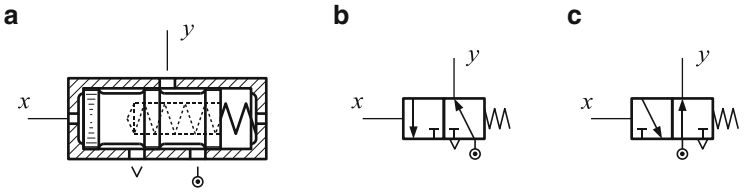

Fig. 2.2 A normally closed valve

output is connected to the air supply while actuating the valve connects the output to the exhaust. This behaviour is described by either of the symbols of Fig. 2.2b, c, and the valve is called a **normally closed valve** or **NC-valve**.

To analyse the behaviour of a spool-and-sleeve valve (Fig. 2.3) it suffices to work with the valve symbol. The input-output behaviour of such a valve is best documented in events tables as shown in Fig. 2.3a2, b2.

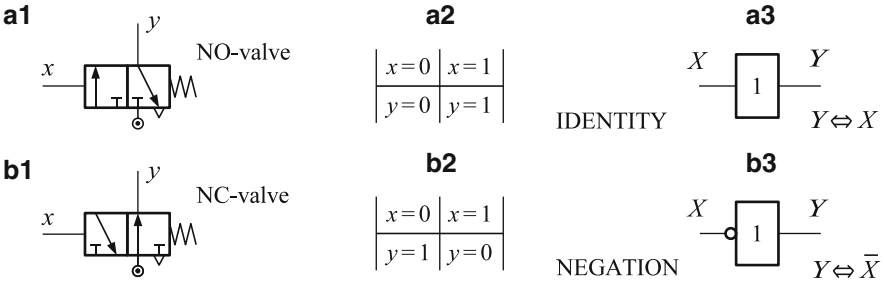

Fig. 2.3 Spool-and-sleeve valves as power amplifiers

According to the ATE of Fig. 2.3a2, 'the output y is 1 iff the input x is 1', or (in falling back on Sect. 1.4)

$$y = 1 \Leftrightarrow x = 1,$$

$$Y \Leftrightarrow X,$$

this function being called the IDENTITY, as the output Y is always identical to the input X. The symbol for the gate is shown in Fig. 2.3a3. You will probably not be alone in asking what good the IDENTITY function is, as its output is always the same as its input. Frankly, *the IDENTITY function serves no logical purpose*. But, it is of real technical importance as a *power amplifier*. In pneumatics, power is determined as the product of flow times pressure. A low-power pilot-signal can control a high-power air-supply to the output.

The behaviour of the NC-valve, on the other hand, is fully documented in the events table of Fig. 2.3b2, and from this follows

$$y = 1 \Leftrightarrow x = 0,$$

$$Y \Leftrightarrow \overline{X},$$

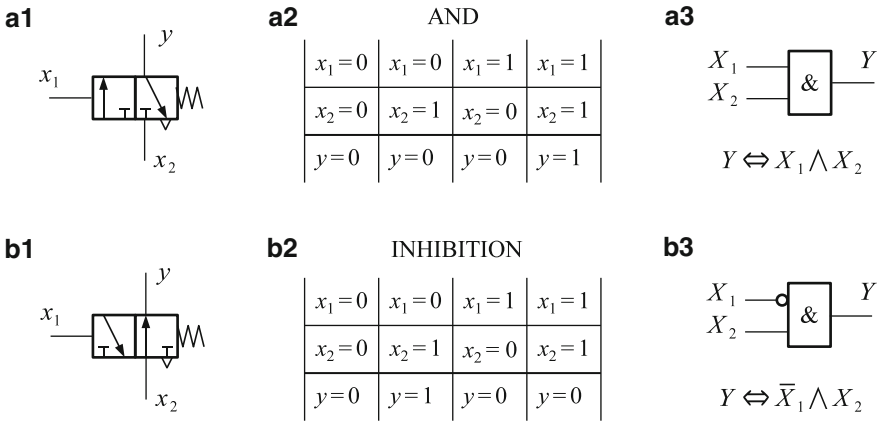

Fig. 2.4 Spool-and-sleeve valves as logic switches

The above function is called a NEGATION, the output always having the complementary value to the input.

The air supply of a spool-and-sleeve valve can be replaced by an input *signal*, as indicated in Fig. 2.4. The difference is that the air supply is *always* 1 while a signal can *vary*, meaning it can be either 0 or 1.

You will hopefully be able to recognise the valves' behaviour as being documented in the two event tables. The functions associated with these event tables are called AND and INHIBITION. The latter name stems from the fact that x_1 *inhibits* y, meaning that, when $x_1 = 1$, the output y is blocked. It is probably routine to you by now to express the events tables of Fig. 2.4a2, b2 in the following formal manner:

$$y = 1 \Leftrightarrow x_1 = 1 \text{ AND } x_2 = 1, \qquad y = 1 \Leftrightarrow x_1 = 0 \text{ AND } x_2 = 1,$$
$$Y \Leftrightarrow X_1 \wedge X_2, \qquad Y \Leftrightarrow \overline{X}_1 \wedge X_2.$$

In the above usages the spool-and-sleeve valve cannot realise the logical OR function, a function no design engineer would like to do without. A cheap and effective realisation of the OR function is the shuttle valve of Fig. 2.5a. The ball in the design shown will always seal the input of lower pressure, connecting the output with the input of higher pressure. This valve is called a **passive** valve as the air of the output signal is derived from an input line, not from an air supply. This is a drawback, as is the fact that the valve does not have an exhaust of its own; the output must be exhausted via an input line. The standard symbol for this valve (Fig. 2.5b) is obviously inspired by the design shown in Fig. 2.5a. The events table describing the valve's behaviour marks the valve as a realisation of the OR function.

To see that the events table of Fig. 2.5c does describe the OR function, read the table row-wise (and not column-wise, as we have been doing up till now). First note that 'if $x_1 = 1$ then $y = 1$' (this covering columns 2 and 3), and then that 'if $x_2 = 1$

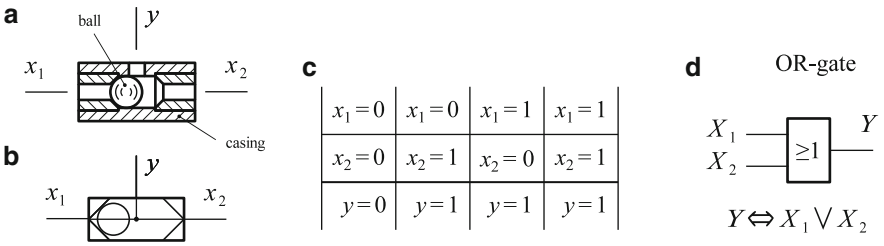

Fig. 2.5 A shuttle valve to realise the OR function

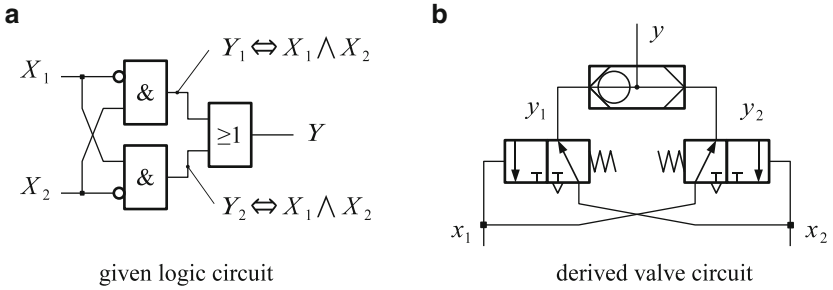

Fig. 2.6 A valve realisation of a logic circuit

then $y = 1$' (which covers columns 1 and 3). Thus we reason that '$y = 1$ iff $x_1 = 1$ OR $x_2 = 1$', and write this as

$$y = 1 \Leftrightarrow x_1 = 1 \text{ OR } x_2 = 1,$$
$$Y \Leftrightarrow X_1 \vee X_2.$$

We are now in a good position to transform a given logic circuit into one consisting of valve symbols. Such a transition is shown in Fig. 2.6 for the running example of Chap. 1. Of course this example is only used to demonstrate a technique, and has nothing to do with the original problem of switching lights on or off. To replace the INHIBITIONs of Fig. 2.6a by NC-valves, note the correspondence laid down in Fig. 2.4.

2.2 Electric Relays

In its simplest form, an electric relay (see Fig. 2.7) consists of a coil, a plunger, and a contact that is actuated by the plunger. When currant flows through the coil its magnetic field attracts the plunger, thereby opening a *normally closed contact* (Fig. 2.7a) and closing a *normally open contact* (Fig. 2.7d). Interrupting the currant

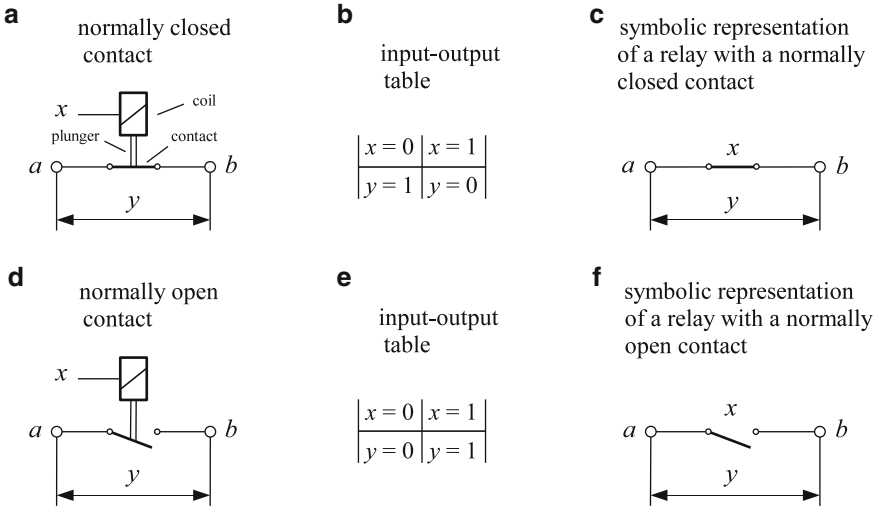

Fig. 2.7 The electric relay: principle operation, IO-table, symbol

to the coil allows a spring to return the plunger, and thus the contacts, to their initial positions.

The **input signal** to a relay is the voltage applied to the relay's coil, this voltage being represented by the binary variable $x \in \{0, 1\}$ of Fig. 2.7. The **output signal** $y \in \{0, 1\}$ stands for the **transmission** of the relay's contact, i.e., its ability to pass or transmit currant. The transmission y of a closed contact is taken to be 1, that of an open contact is 0.

Relays (as those of Fig. 2.7a, d) are *always* drawn under the assumption that their input signals are 0, i.e., that zero voltage is applied to their coils—this being referred to as the **normal state** of the relays. Thus, a *normally open contact*—a so-called NO-contact—is open ($y = 0$) when the relay's coil is not energised (when $x = 0$), and closed ($y = 1$) when $x = 1$. A *normally closed contact*—a NC-contact—is closed ($y = 1$) when the relay's coil is not energised ($x = 0$), and open ($y = 0$) when $x = 1$. This behaviour is expressed in the input-output tables (the events tables) of Fig. 2.7b, e. Stating when the outputs y of these tables are 1 is expressed formally in the logic formulas

$$y = 1 \Leftrightarrow x = 0, \qquad y = 1 \Leftrightarrow x = 1,$$
$$Y \Leftrightarrow \overline{X}, \qquad Y \Leftrightarrow X,$$

which show clearly that the relay with a normally closed contact is a NEGATION, while the relay with a normally open contact is an IDENTITY.

The **symbolic representation of a relay** (see Fig. 2.7c, f) consists of only two things: (a) a contact drawn in its *normal state*, its transmission y standing for the output of the relay, and (b) an input variable—in our example, x—which is the input

signal to the coil activating the contact. The symbolic representation of a relay is an anachronism in that it has neither an input lead, nor a clearly defined output lead (such as the *logic symbols* of Sect. 1.5, or the *pneumatic symbols* of Sect. 2.1 have).

As stated for pneumatic valves, by **convention**, all relays in a circuit are drawn in their **normal state** which means that they are drawn as if their input signals were zero.

The coil of a relay usually activates multiple contacts, some normally open, the others normally closed. A single contact, or two or more interconnected contacts that transmit current to a *single load* comprise a **relay network**; for example, see the serial network of Fig. 2.8b or the parallel network of Fig. 2.9b. The interconnected contacts of a relay network are always drawn *in their normal state*, assigning to each contact the input variable of the coil activating it (in our cases, the variables x, x_1, x_2, etc.). **The way the contacts are connected** unequivocally determines when the network's transmission y is 1. The network's transmission y is usually assigned to the network's load. A **load** is any device (such as a lamp, an electric motor, a relay coil, etc.) that receives and dissipates electric energy. The load is always needed to avoid the network from short circuiting. The only relay networks to be touched on here are the serial-parallel networks, well knowing that this does not do the possibilities of relay networks justice.

Figure 2.8a shows two relays in series, with a lamp as load. As explained in the above paragraph, the *relay network* is drawn as in Fig. 2.8b. The analysis of the input-output behaviour of Fig. 2.8a (or Fig. 2.8b, if you prefer) is undertaken in the events table of Fig. 2.8c, column 3 of this table stating when y is 1:

$$y = 1 \Leftrightarrow x_1 = 1 \text{ AND } x_2 = 0,$$

$$Y \Leftrightarrow X_1 \wedge \overline{X}_2.$$

This result lets us **conjecture** that the serial connection of two contacts is expressed by the logic AND ($\wedge$), and that, vice versa, the logic AND ($\wedge$) is realised by connecting two contacts in series. For our example, this conjecture allows us to

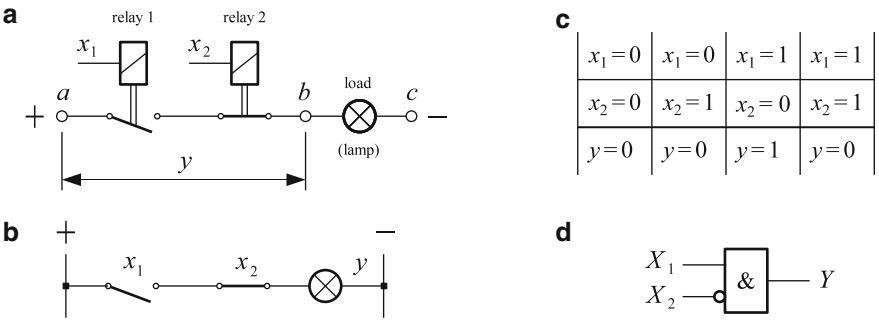

Fig. 2.8 Principle of a serial network

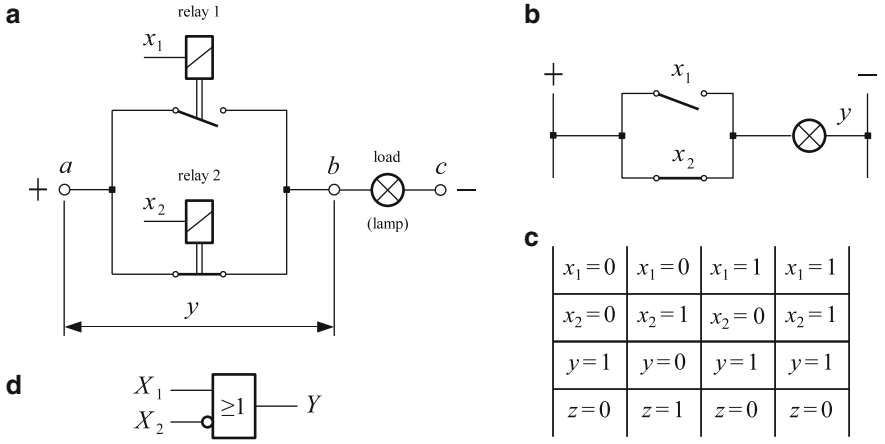

Fig. 2.9 Principle of a parallel network

state that the behaviour of the circuits of the Figs. 2.8a, b is equivalent to that of the gate of sub-figure (d).

Figure 2.9a depicts relays connected in parallel, again with a lamp as a load; the symbolic relay network is drawn in Fig. 2.9b. From either of these representations we deduce that the circuit's transmission y is 1 when the transmission of at least one of the contacts is 1; this is expressed in the events table of sub-figure (c) when concentrating only on the transmission y (i.e., for the moment, not considering the variable z).

As with the shuttle valve of Fig. 2.5, we read the events table of Fig. 2.9c row-wise arguing that $y = 1$ *iff* $x_1 = 1$ (hereby describing columns 3 and 4) OR $x_2 = 0$ (which covers column 1 and again column 3),

$$y = 1 \Leftrightarrow x_1 = 1 \text{ OR } x_2 = 0,$$

$$Y \Leftrightarrow X_1 \vee \overline{X}_2.$$

This result lets us **conjecture** that the parallel connection of two contacts is expressed by the logic OR ($\vee$), and that, vice versa, the logic OR ($\vee$) is realised by connecting two contacts in parallel. For the present example, this conjecture allows us to state that the behaviour of the circuits of the Fig. 2.9a, b is equivalent to that of the gate of sub-figure (d).

Inverting a Relay Circuit

There are two ways to invert a circuit. The simplest and most direct way is to invert the circuit's output. For instance, suppose we want to invert the circuit of

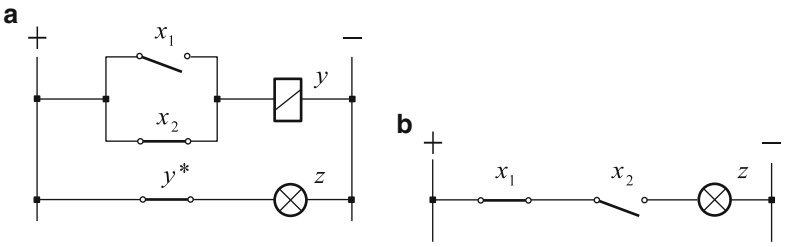

Fig. 2.10 Expressing the OR-function using double negation

Fig. 2.9b. This is done by replacing the lamp—which is the circuit's output load—by an inverter, that is, by a normally closed relay as shown in Fig. 2.10a. The relay's coil, y, is now the load for the original circuit, while the lamp is the load, z, for the inverter. The values of z are shown in the bottom row of Fig. 2.9c—they are the complement of those for y.

The second way to invert a circuit is to calculate the inverted circuit directly from the inverted output z of Fig. 2.9c, obtaining

$$Z \Leftrightarrow \overline{X}_1 \wedge X_2,$$

which leads to the circuit of Fig. 2.10b.

Realising Feedback

Feedback plays a dominating role in the design of so-called latches, these being elementary memory circuits. In fact they are so important that a whole Part of this book is devoted to them. In this early stage, we only want to see how to transform a logic feedback circuit, say, that of Fig. 2.11a, into a relay circuit. The output OR-gate of sub-figure (a) requires our contact network to consist of two parallel paths, one of which contains the NO-contact x_2. The AND-gate of sub-figure (a) leads to the serial circuit consisting of the NC-contact x_1 and the NO-contact named y^*. The load of the relay network is the coil y of a relay. To realise the *feedback* of the

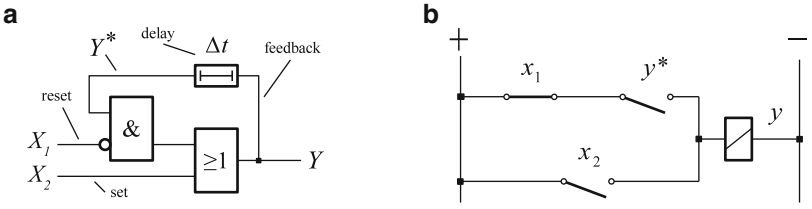

Fig. 2.11 Transforming the logic circuit of a latch into a relay network

latch pictured in sub-figure (a) we define the NO-contact y^* of sub-figure (b) to be actuated by the coil y.

The delay Δt in the logic circuit of sub-figure (a) expresses the fact that the NO-contact y^* follows the coil's state of activation with the time delay of Δt. The logic circuit is thus modelled after the relay circuit. But it must be noted that the *theory* of feedback latches does not require the existence of a delay in the feedback loop.

2.3 CMOS Transistors

The transistor was invented in 1948 by John Bardeen, Walter Brattain and William Shockley and soon replaced the vacuum tube as a cheap amplifier. We shall take a look at only one transistor device, of a large proliferation, namely the CMOS technology. The transistors used are **insulated-gate field-effect transistors (IGFET)**—Fig. 2.12—more commonly called **MOS** for the sequence of the layers of material used in their design—**metal-oxide-silicon**. There are two types of these transistors, according to the polarity (**n**egative or **p**ositive) of the electric charges used to create a conducting channel between the *drain* and the *source* regions: *n-channel MOS* (**NMOS**) and *p-channel MOS* (**PMOS**). Digital circuits that employ both these transistor types are called **complementary MOS (CMOS)**, and have the advantage of needing only a single level of supply voltage.

For current to be able to flow between the drain and the source region, you need to *create a conducting channel* in the silicon between these regions. This is achieved by applying the proper voltage between the gate and the substrate. In an n-channel MOS you obtain a conducting channel by applying a more positive potential to the gate than to the substrate (the voltage from gate to substrate is positive), thus attracting negative charges to the gate while repulsing positive charges. In a p-channel MOS the conducting channel is obtained by making the voltage from substrate to gate positive, thus attracting positive charges to the gate while repulsing negative charges from the gate. *In actual usage, the source is always connected*

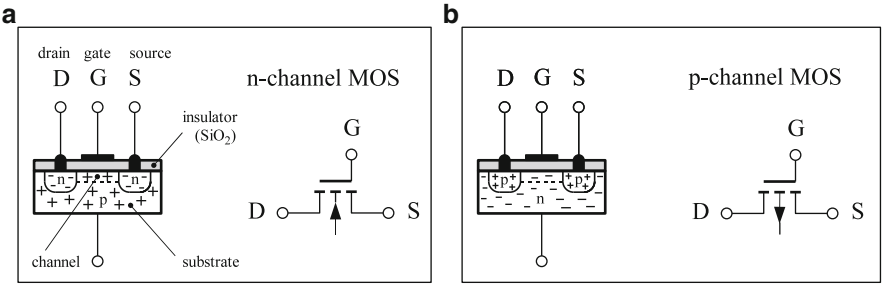

Fig. 2.12 Principle of the n-channel and p-channel MOS

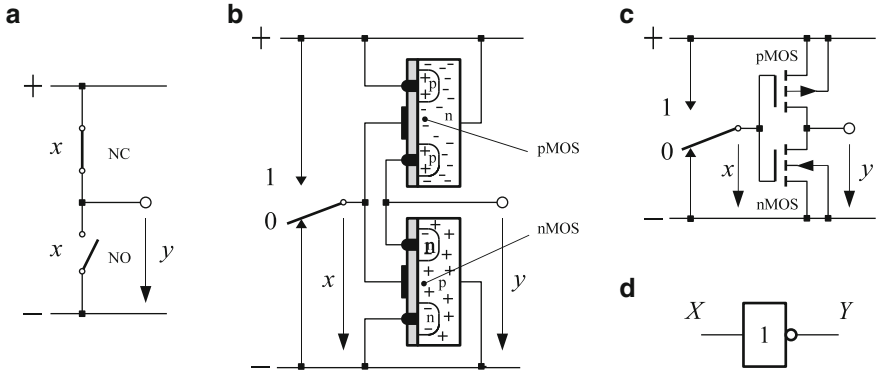

Fig. 2.13 The CMOS inverter

to the substrate. To obtain a channel, the density of source-type charges must be enhanced giving these transistors the name **enhancement type transistors**.

The simplest CMOS circuit is the **inverter** of Fig. 2.13. Its principle of operation is basic to CMOS technology. If you understand how the inverter works, you almost automatically understand the next two gates.The idea behind the inverter is shown in the relay circuit of Fig. 2.13a: The two serial contacts are always in a state of alternate transmission—when one is closed, the other is open. When the top switch, the NC-contact, is closed and the bottom switch, the NO-contact, open, the full supply voltage is fed through to the output y. *Vice versa*, the output y is zero when the top switch is open and the bottom one closed.

The relay circuit translates directly to the transistor circuit of Fig. 2.13b, the NC-contact being replaced by a PMOS, the NO-contact by an NMOS. In the circuit as drawn, the gates of both transistors are at ground potential dew to the way the switch governing the gate potential is set. As the substrate of the PMOS is connected to high potential a conducting p-channel is created. The NMOS transistor, on the other hand, is non-conducting as it has 0 voltage between gate and substrate, both being connected to ground. The voltage measurable at the output y is thus high. If the switch is brought to the alternate position, in which the potential at the gates of both transistors is high, the PMOS becomes non-conducting while the NMOS becomes conducting. The output voltage y is thus brought to zero. The actual circuit, of course, is always drawn using MOS symbols, giving you Fig. 2.13c. As the input and output of this circuit are always complementary, it is an inverter or a negation, the symbol for which is shown in Fig. 2.13d.

There are two successful ways, of four trivial ones, to generalise the principle inverter circuit of Fig. 2.13a: One of these is shown in Fig. 2.14a, the other in Fig. 2.15a, and both lead to circuits superbly adapted to CMOS technology.

The first generalisation of the inverter of Fig. 2.13a is achieved by substituting the single NC-contact by two serial NC-contacts, and substituting the single NO-contact by two parallel NO-contacts—this leading to Fig. 2.14a. The equivalent

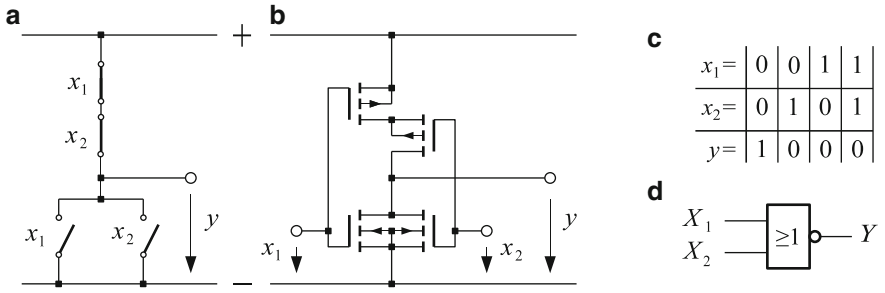

Fig. 2.14 The CMOS NOR-circuit

CMOS circuit is obtained by substituting PMOS transistors for NC-contacts, and NMOS transistors for NO-contacts. As with the inverter of Fig. 2.13c, note that in Fig. 2.14b the sources and substrates of the PMOS transistors are connected to high potential, or at least higher potential, while the sources and substrates of the NMOS transistors are connected to low (or lower) potential.

The simplest way to analyse the circuit of Fig. 2.14 is to develop the events table shown in Fig. 2.14c. You start out with the table of input events, leaving the bottom row, the one for the output y, empty. Then, for each input event, you deduce from the behaviour of the circuit the value for the output y, entering the outputs into the bottom row. Considering the result, you will notice that inverting each output value leads to the **or** function. In other words, the events table of Fig. 2.14c is that of a **not-or** function, usually referred to as the **nor** function. The logic symbol used for this function is shown in Fig. 2.14d; it is simply an OR gate with an inverted output.

The second generalisation of the inverter leads to Fig. 2.15a and that to Fig. 2.15b. The analysis needed to obtain the events table of Fig. 2.15c is easy so that I again leave the details to you. The result, as you will have noticed, is the inversion of the **and** function, i.e. is a **not-and** function, and is thus referred to as the **nand** function, the logic symbol for which is shown in Fig. 2.15d.

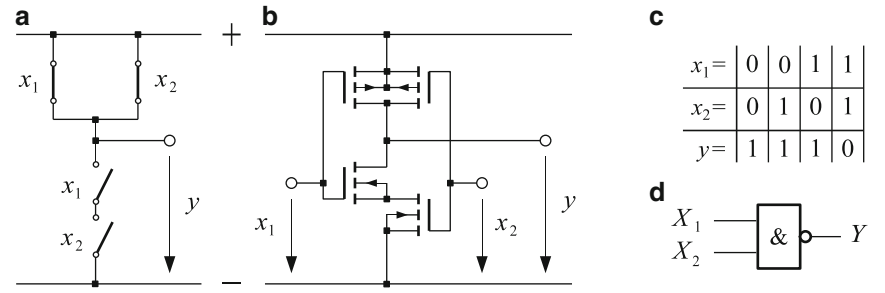

Fig. 2.15 The CMOS NAND-circuit

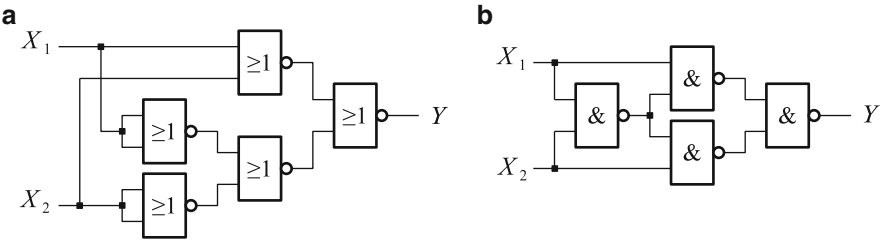

Fig. 2.16 NOR and NAND versions of our running example

Of course the CMOS circuits discussed here are only a small excerpt of the circuits used. But they are basic and absolutely essential to the understanding of the techniques employed. One of the most important omissions is a discussion of circuits with feedback, circuits that lead to the realisation of memory devices—so-called *latches* and *flip-flops*. These circuits are discussed, together with their theory, in later chapters.

The logic building blocks of CMOS technology are NAND-, NOR- and NEGATION-gates. These, therefore, one wishes to use when realising logic circuits in CMOS technology. To get an impression of pure NOR- and NAND-design, take a look at Fig. 2.16. These circuits are equivalent to that of Fig. 1.5 of our running example of Chap. 1. Transforming larger AND, OR and NOT based circuits into NOR or NAND circuits can be quite demanding and is discussed in a later chapter.

http://www.springer.com/978-3-642-27656-9

Logic Circuit Design
Selected Methods
Vingron, S.P.
2012, XIV, 258 p., Hardcover
ISBN: 978-3-642-27656-9