Chapter 2

RELATED WORK

2.1. Integrated circuits and programmable devices

2.1.1. Introduction

By the late 1940s the first transistor was created as a point-contact device formed from germanium. Such an invention was the basis for further digital circuits and programmable devices. In the next decade the development of transistors benefited in the first digital logic gates and circuits classified as TTL (transistor-transistor logic). The devices had up to 16 pins and each could perform a simple logic function (for example, the device 7400 contained four 2-input NAND gates, 7404 – six NOT inverters). Those small circuits were the first *application specific integrated circuits (ASICs)* where logic functions were fixed and unchangeable. It means that the ASIC contains dedicated logic values and cannot be reconfigured. Up to the early 1970s, ASICs were developed implementing more and more gates on one chip, although the main problem was fixed logic. Once manufactured device could not be changed, therefore there was no possibility to correct any errors or bugs. The designer could not verify her/his prototype using a real physical circuit.

This problem was solved in the 1970s, when the first programmable devices were introduced. Those circuits were named *programmable logic devices (PLDs)*. The PLD was built as a fixed array of AND (OR) functions driving a programmable array of OR (AND) functions (Maxfield, 2004). Such a circuit was initially used to implement simple combinational logic. Later, registered and tri-state outputs were added. In the early 1980s the first *complex PLD (CPLD)* was presented. Basically, the CPLD is an extended version of the PLD; it contains small PLD blocks linked to interconnect arrays. CPLDs were highly configurable but it was impossible to implement large and complex functions. Thus, in 1984, the first *field programmable gate array (FPGA)* was introduced. The device was made of a matrix of programmable logic blocks. Each logic block contained a 3-input *look-up table (LUT)* that could perform any combinational function. Additionally, there were a simple multiplexer and a flip-flop inside the logic block.

Nowadays, FPGAs are still the best solution for designers who want to verify their prototype of the device. However, FPGAs are too slow and too expensive to compete with ASICs in the case of mass production. On the other hand, features of FPGAs like partial reconfiguration offer new ideas and new markets for such devices.

2.1.2. Programmable logic devices

Programmable logic devices can be generally divided into two groups:

- **Simple programmable logic devices (SPLDs)**, which refer to relatively small programmable devices. Three types of the SPLD: PROM, the PLA and PAL are briefly described in this subsection.
- **Complex programmable logic devices (CPLDs)**, which consist of multiple smaller devices (like the SPLD). Because of their structure, CPLDs will be shown in the next subsection in greater detail.

2.1.2.1. Programmable read-only memory (PROM)

The first programmable logic device was made as *programmable read-only memory (PROM)*. Generally, PROM performs the function of a fixed AND-array which drives the programmed OR gates (array). The idea of PROM is shown in Fig. 2.1 (the device is in the unprogrammed state).

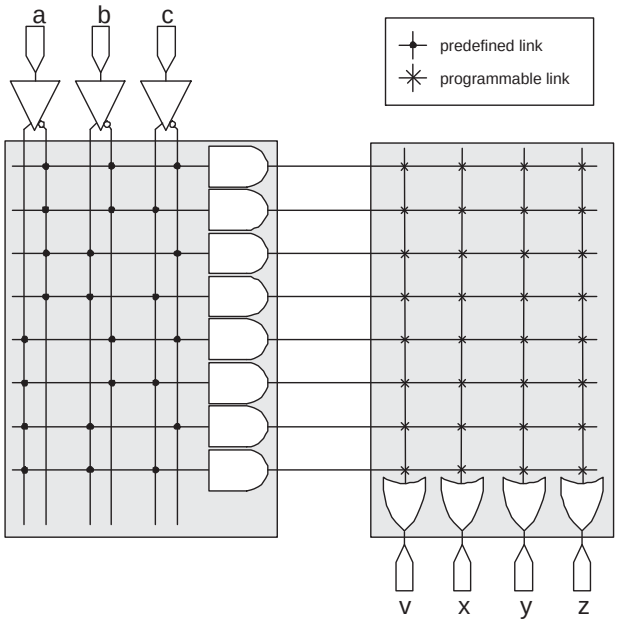

Fig. 2.1: Unprogrammed PROM device

Here the predefined (fixed) AND array has three inputs (variables a,b,c) and eight outputs that are created as all combinations of the inputs. Thus the outputs of the AND matrix hold all terms of the input function. The OR-matrix is programmable and its inputs refer to the outputs of the AND-matrix. Programmable

links of the OR array permit to create any sum of logic terms, therefore any logic function may be programmed using the AND-OR structure. An example of a programmed PROM device is shown in Fig. 2.2. The device performs simple combinational functions of three inputs and four outputs, where

$$\begin{aligned} v &= a \cdot b \cdot \overline{c} + a \cdot b \cdot c, \\ x &= \overline{a} \cdot b \cdot \overline{c} + \overline{a} \cdot b \cdot c, \\ y &= \overline{a} \cdot b \cdot c + a \cdot \overline{b} \cdot c + a \cdot b \cdot c, \\ z &= \overline{a} \cdot b \cdot \overline{c} + a \cdot \overline{b} \cdot c + a \cdot b \cdot c. \end{aligned} \tag{2.1}$$

The AND-array produces the combination of all possible logic terms. The logic functions v, x, y, z are in fact realised by the OR-array via programmable lines.

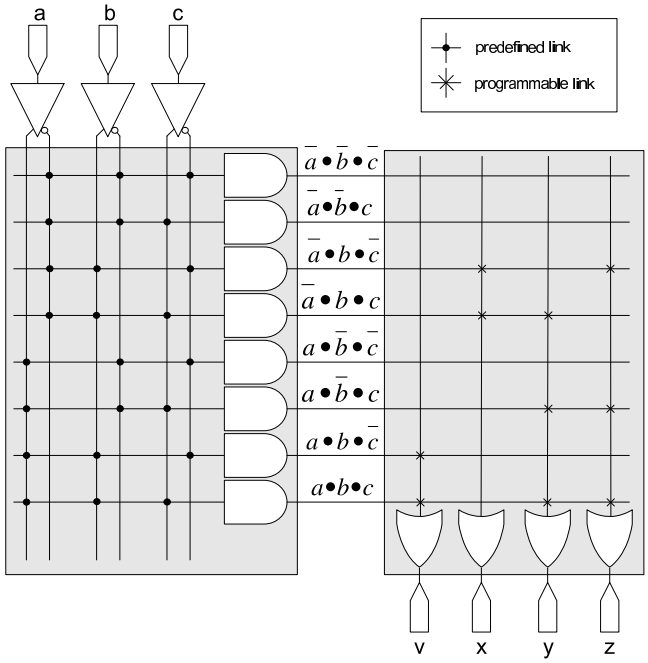

Fig. 2.2: Programmed PROM device

It is clear that every combination of inputs is always decoded creating the possibility of using all terms of the input function. Therefore, PROMs are useful for circuits that realise functions with a large amount of product terms. The main problem in PROM-based PLDs is the number of inputs because each additional input requires a memory volume that is twice as wide.

2.1.2.2. Programmable logic array (PLA)

In the mid-1970s, a new way of PLDs realisation was invented. In *programmable logic arrays (PLAs)*, both matrices, AND and OR, are configurable. What is most important, the number of inputs does not influence the number of terms performed by the AND-array. Therefore, the device based on PLA technology could perform functions with a relatively higher number of inputs in comparison to PROM. An example of an unprogrammed PLA device is illustrated in Fig. 2.3.

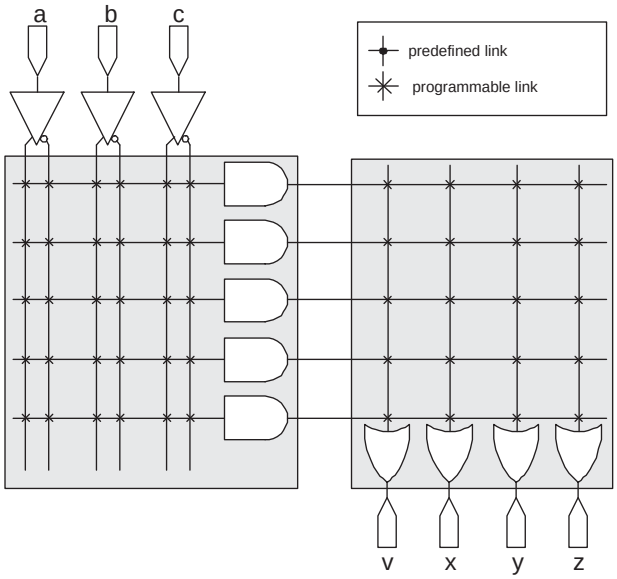

Fig. 2.3: Unprogrammed PLA device

The cost of the device can be reduced by minimizing the functions that ought to be realised (Dagless, 1983; Ciesielski and Yang, 1992; Yang and Ciesielski, 1989; Sasao, 1988). Because both matrices are programmable, the AND-matrix defines prime implicants while the OR-matrix sums all necessary implicants. The functions defined in the example (2.1) can be minimized and represented as

$$\begin{aligned} v &= a \cdot b, \\ x &= \overline{a} \cdot b, \\ y &= a \cdot c + b \cdot c, \\ z &= a \cdot c + \overline{a} \cdot b \cdot \overline{c}. \end{aligned} \tag{2.2}$$

There are only four primary implicants needed to represent all functions. Therefore, the AND-matrix realises four minterms unlike the PROM device, where all eight lines were used. Furthermore, the OR-matrix sums the products generated

by the AND-matrix using four programmable lines (Fig. 2.4). Additionally, the functions y and z use the sharing of the minterm ac, thus the size of both matrices is reduced.

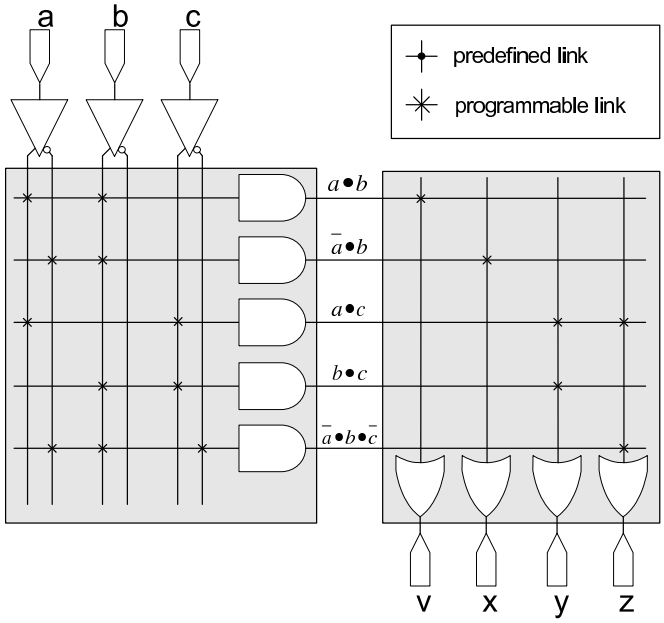

Fig. 2.4: Programmed PLA device

The PLA is useful especially for large projects where many common implicants are present in the design. However, because of its structure, the PLA was relatively expensive to manufacture. Furthermore, the device was slow due to propagation delays that appeared in programmable links. Therefore, in the late 1970s, a new type of PLD devices was proposed – programmable array logic.

2.1.2.3. Programmable array logic (PAL)

The structure of *programmable array logic (PAL)* is the opposite of the PROM structure. There is a programmable AND-matrix and a fixed OR-plane in the PAL device. An example of an unprogrammed PAL circuit is presented in Fig. 2.5.

In PAL, only the first level of the device is configurable. Thus, any variation of input values in the AND-matrix may be defined by the designer. Such a combination creates product terms at outputs of the AND-plane. The OR-matrix is fixed so it can connect a restricted number of product terms for the realisation of logic functions. Figure 2.6 shows example implementation of the functions 2.2. Similarly to the PLA, the AND-matrix is programmed to generate prime implicants on its outputs. Furthermore, the OR-matrix generates proper functions using fixed OR-lines.

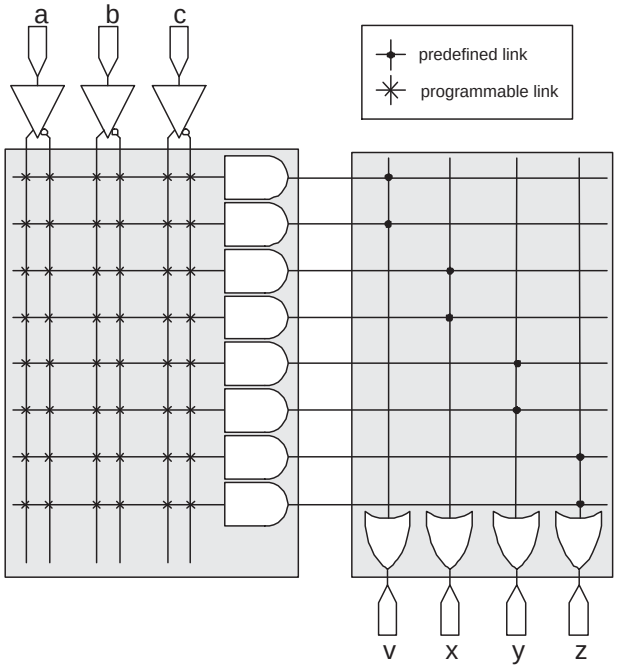

Fig. 2.5: Unprogrammed PAL device

PAL devices are not so flexible in configuration as PLAs. The structure of the device ought to be prepared, and very often designers use some approaches to omit the restricted number of product terms that should be OR-ed (Kania, 2004; Kania and Kulisz, 2007; Hrynkiewicz *et al.*, 1997). The main advantage of PAL devices in comparison to PLAs is their speed. There is only one programmable matrix in PAL, thus the circuit is much faster in comparison with the PLA device. Another important benefit is the price. Thanks to their low cost and high speed, PAL devices became very popular in the late 1970s. However, the development of prototyped systems effected the appearance of a new branch of programmable circuits: complex programmable logic devices.

2.1.3. Complex programmable logic devices

The complex programmable logic device appeared at the beginning of the 1980s. The main idea was to connect several small PLD devices to create a wider area for programmable logic. The first CPLDs had 100% connectivity to the inputs and outputs associated with each block (Maxfield, 2004). Therefore, the interconnection array was huge, which made the whole device slow and expensive in production. The solution to this problem was the programmable interconnect array (PIA). Nowadays, each vendor of CPLD devices has its own technology of

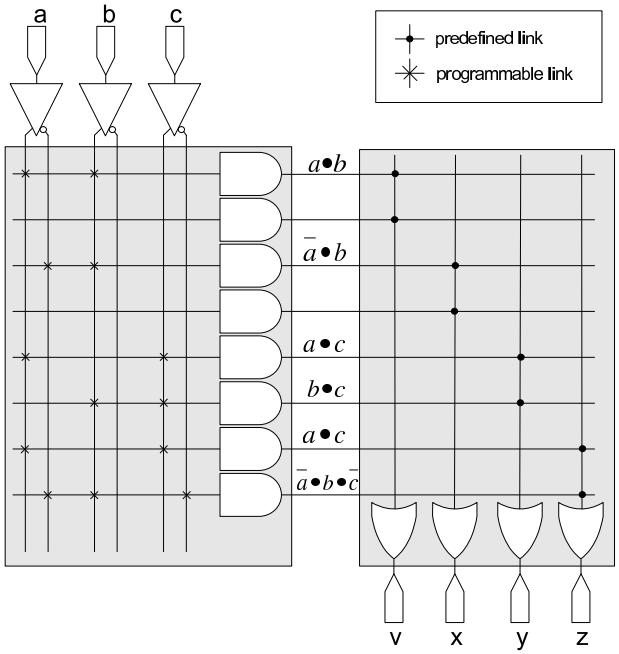

Fig. 2.6: Programmed PAL device

CPLD manufacturing, but the idea is similar: all SPLDs share a common PIA (Kania, 2004; Łuba, 2003; Maxfield, 2004). Figure 2.7 shows the structure of a typical CPLD device.

As has been mentioned, a typical CPLD consists of a programmable interconnect array surrounded by macrocells (Kania, 2004; Łuba, 2003; Maxfield, 2004). The macrocell is built using AND-OR matrices (usually small PLDs, like PAL), programmable flip-flops, and additional logic elements like multiplexers, OR and XOR gates. The most popular CPLDs that are currently available on the market are devices from Altera, Xilinx, Atmel, Lattice, Lucent and Cypres.

2.1.4. Field programmable gate arrays

The first field programmable gate array appeared in the mid-1980s. Its structure was different in comparison to the CPLD (Xilinx, 2001; Altera, 2008; Maxfield, 2004; Jenkins, 1994). The main idea of FPGAs was to use programmable logic elements for the implementation of simple logic functions. Such elements are called look-up tables (LUTs) and can perform any logic function with a specific number of inputs and one output. Early FPGAs contained logic elements that could perform any logic function up to three inputs and one output. Additionally, each LUT was connected with a multiplexer and a register, which created a simplified programmable logic block. The idea of the early programmable logic block is

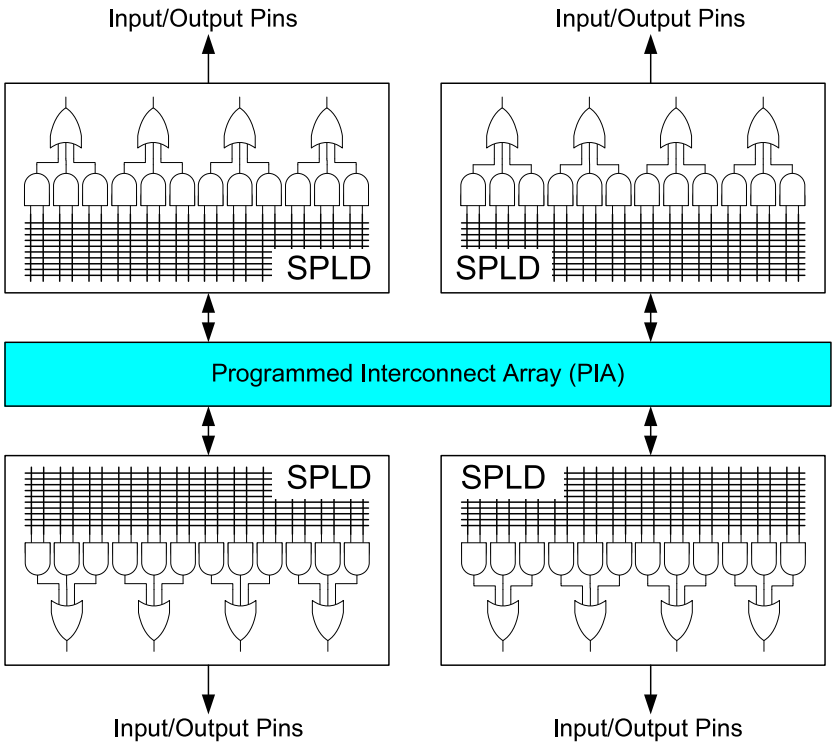

Fig. 2.7: CPLD structure

illustrated in Fig. 2.8. As has been mentioned, the LUT could be configured to perform any combinational function with three inputs and one output. Sequential functions could also be realised thanks to the register connected with the LUT.

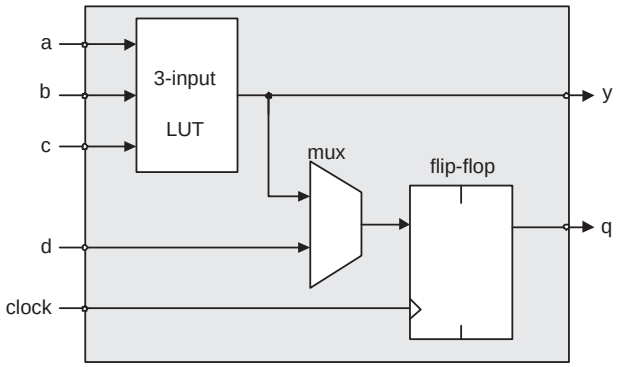

Fig. 2.8: Early CLB structure

To perform functions that require more than one output (or more than three inputs), more logic blocks were used. Thanks to the structure of the FPGA, logic blocks are connected via an interconnect matrix. Generally, the FPGA was created as a "... large number of logic blocks (islands), surrounded by a sea of programmable interconnections ..." (Maxfield, 2004).

Nowadays, there are many vendors of FPGA devices. The most popular are Xilinx, Altera, Lattice, Actel and Atmel. The structure of the FPGA depends on the vendor, although the idea is the same – the usage of the array of logic blocks based on the LUT and the register. There are different names for logic blocks, interconnections and other elements of the FPGA because vendors ascribe new ideas in the branch to their inventions. In this dissertation, the structure of the FPGA will be described based on Xilinx devices. Therefore, all references and information will concern Xilinx FPGAs.

Typical structure of the FPGA from Xilinx is shown in Fig. 2.9. The main elements of the device include: a matrix of configurable logic blocks (CLBs), configurable input/outputs blocks (IOBs) and dedicated memory blocks (BRAMs). All elements are connected via a programmable interconnect matrix.

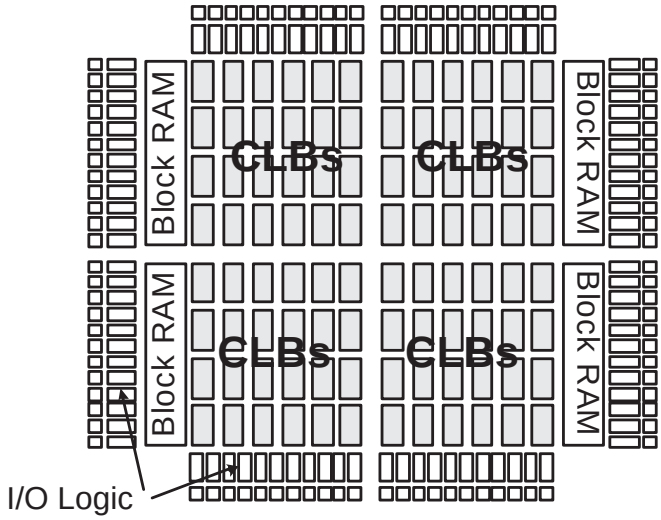

Fig. 2.9: Structure of a typical FPGA device

The *configurable logic block (CLB)* consists of four *logic cells (LCs)*. Two logic cells are organized into a *slice*. Thus each CLB has two similar slices. Figure 2.10 illustrates the idea of the CLB structure.

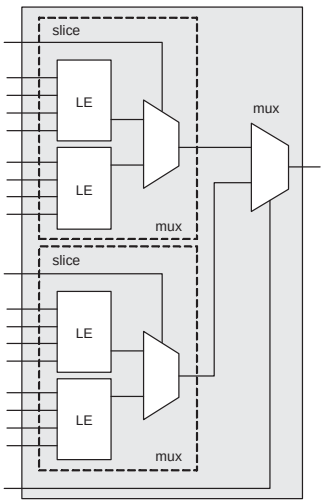

Fig. 2.10: Structure of the CLB block

The main elements of the logic cell include: a four-input and one-output LUT, a multiplexer and a flip-flop. Each family also has additional logic (like carry logic for arithmetic operations); however, it is not important for our further discussion. A simplified structure of the logic element (LE) is presented in Fig. 2.11. The register can be configured either as the flip-flop or a latch. What is important, the polarity of the clock may be rising-edge or falling-edge. Thus each LE could be configured as an active-low or active-high clock trigger.

The look-up table has four inputs and one output. Therefore it can realise any logic function that has maximum four input variables. Larger functions ought to be decomposed and more LUTs (or even slices) have to be used. Additionally, the look-up table can be configured as 16x1 RAM (random-access memory) or a 16-bit shift register.

The main role of the *configurable input/output block (IOB)* is to ensure the connectivity beetween the FPGA and other elements of prototyped systems. Thus a very essential fact is the wide variety of power supply standards (Maxfield, 2004). IOBs are organized into banks (the number of banks depends on the FPGA). Each bank may be configured independently so the designer can connect other devices to the FPGA, with different input/output standards.

Dedicated memory blocks are very important components of the FPGA. Nowadays, a lot of designs contain memory that can be implemented with dedicated memory blocks (Bursky, 1999). Each vendor has its own concept of such blocks. In the case of Xilinx devices, there are *block-RAMs (BRAMs)*. Such elements are synchronous, therefore the clock signal ought to be delivered. The number of BRAMs and the number of logic cells per each block depend on the particular device. BRAMs are organized into columns and each BRAM can be used separately.

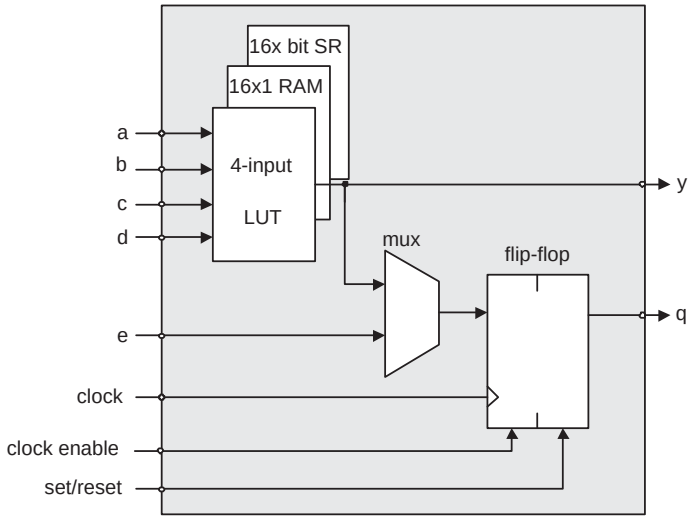

Fig. 2.11: Structure of the logic element (LE)

The main advantage of BRAMs is their size and reconfigurability. Depending on the device, there can be stored even up to three mega bits in the memory (the XC2V8000 device from the Virtex-II family). Additionally, BRAMs may be very easily reconfigured by the process of partial reconfiguration. The content of one (or more) BRAM is replaced while the rest of the device remains unchanged. Such a solution reduces the size of the target bit-stream used to configure the FPGA. Moreover, the design and configuration time is highly reduced as well. Partial reconfiguration of control units implemented on FPGAs is described in Chapter 6 in greater detail.

2.2. Control units

A digital system may be represented by the composition of the *control unit (CU)* and the *operational unit (OU)*, also known as a data-path (Gajski, 1996; De Micheli, 1994; Barkalov and Węgrzyn, 2006b). The idea of such a defined digital system is illustrated in Fig. 2.12.

Based on the set of input values (I) and the set of logic conditions (X), the CU sends proper microoperations (Y) to the OU. Additionally, a set of output values (O) is generated. The set of inputs (I) and the set of outputs (O) are used for communication with the environment of the digital system (Łuba, 2001; Molski, 1986).

The operational unit executes microoperations (Y) by processing proper input (*Data*) and generating results (*Results*). Additionally, the OU generates logic conditions (X) for the control unit.