4 Review of Field Programmable Gate Arrays (FPGAs)

4.1 Programmable Logic Devices

Programmable logic devices (PLDs) are a general denomination for integrated circuits whose *hardware* is programmable. They are different from microprocessors, for example, for which the tasks are programmable but for which the hardware is fixed; the hardware itself is programmable in a PLD, so that with the same device many different circuits (including microprocessors) can be implemented.

A major motivation for PLDs was the fact that at that time (1970s) most digital circuit boards were constructed with several (or many) devices from the 74xx series, as illustrated in figure 4.1, where the smaller devices are 74xx chips, which could potentially be replaced with a single PLD.

To achieve that purpose, programmable AND-OR arrays were employed as illustrated in figure 4.2a, where the little circles represent programmable connections, so different boolean functions can be implemented with the same hardware. This kind of implementation is called *sum of products* (SOP) because it consists of a product layer (AND gates) followed by a sum layer (OR gate). An example is shown in figure 4.2b, which implements the SOP $y = a \bullet b + a' \bullet b' \bullet c \bullet d$ (black circles indicate that the wires are interconnected; of course, a logic 1 must be applied to the unused inputs of the first AND gate, while a logic 0 must be applied to at least one input of the third AND gate because it is not used in this example).

The first PLDs were introduced in the 1970s, as indicated in table 4.1, which summarizes their evolution. The first two approaches were called programmable logic array (PLA), introduced by Signetics in mid-1970s, and programmable array logic (PAL), introduced subsequently by Monolithic Memories. These devices, however, had major limitations—the main of which was the absence of flip-flops—so they were adequate for implementing only combinational circuits.

Such limitations were removed with the introduction of the generic array logic (GAL) approach, by Lattice, in early 1980s, which contains a "macrocell" at every output, with

---

I am grateful to Stephen Trimberger, and his 30-year experience with Xilinx, for kindly reviewing and helping improve this chapter. The reading of his paper "Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology" (*Proceedings of the IEEE*, vol. 3, March 2015) is highly recommended.

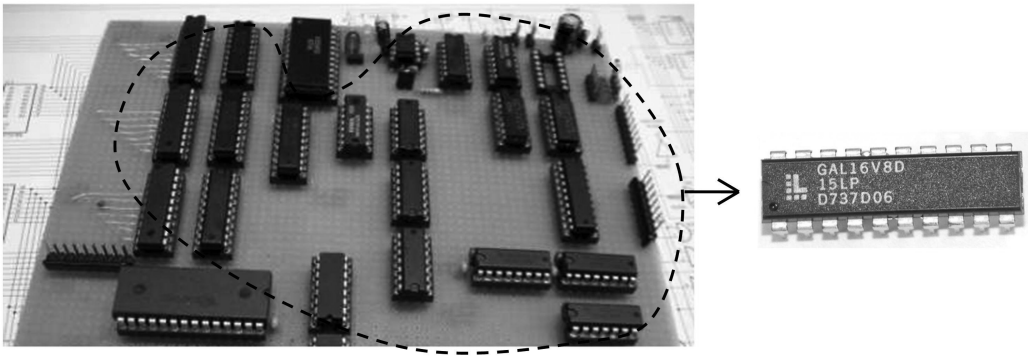

Figure 4.1
Replacement of several 74xx devices with a single PLD.

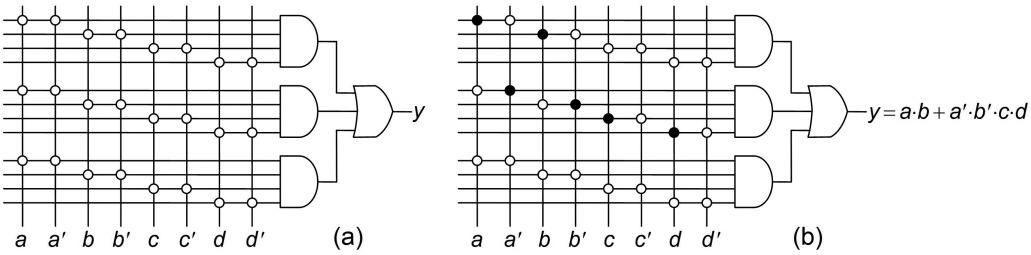

Figure 4.2
Illustration of programmable AND-OR array of first PLDs.

a flip-flop and several routing options (details will be shown in section 4.4). Some of such devices, like the GAL16V8 chip (notice that this is the PLD in figure 4.1), gained substantial popularity, finally bringing attention to the PLDs field.

Two companies, which eventually consolidated and today dominate the PLDs market, were founded around that time. The first (Altera, 1983, now part of Intel) introduced the complex PLD (CPLD) approach, which consists of relatively small arrays of moderate-size GAL-like blocks, still with nonvolatile configuration memory but with other features like more I/O options and more sophisticated clock and logic routing schemes. The second (Xilinx, 1984) introduced the field programmable gate array (FPGA—this term was popularized later, by Actel) approach, which consists essentially of a large matrix of small GAL-like macrocell clusters (hence a 2D array instead of the 1D array of previous PLDs), with the additional important differences that the programmable AND-OR array was replaced with a lookup table (LUT) and the nonvolatile configuration memory was replaced with a volatile version (SRAM). The first devices of these two companies, both delivered in 1984, are pictured in figure 4.3.

All cases listed in table 4.1 are described individually next. However, before we start, the main strategies for holding the configuration data in PLDs are described.

Table 4.1
Historic view and some construction details of programmable logic devices

Type	Architecture	Logic functions implementation	Main config. technologies	Introduced by
SPLD	PLA	AND-OR array, both programmable	EEPROM	Signetics, mid 1970s
	PAL	AND-OR array, only AND programmable	EEPROM	Monolitic Memories, mid/late 1970s
	GAL	AND-OR array, only AND programmable	EEPROM	Lattice, early 1980s
CPLD	Small array of GAL-like blocks	AND-OR array, only AND programmable	EPROM, then EEPROM, then Flash	Altera, 1984
	Simplified FPGA, with nonvolatile config. memory (*)	LUT		Altera, early/mid 2000s (MAX II, then V and 10)
FPGA	Large array of small GAL-like clusters, with LUT in place of AND-OR array and volatile config. memory	LUT	SRAM	Xilinx, 1984
			Antifuse	Actel, late 1980s
			Flash	Several companies

(*) Not true CPLD, now correctly referred to as FPGA, just with nonvolatile configuration memory.

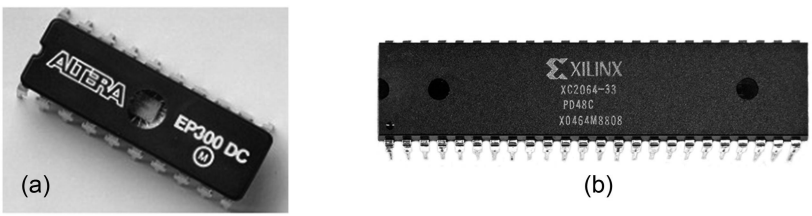

Figure 4.3
(a) First Altera CPLD (EP300) and (b) first Xilinx FPGA (XC2064), both from 1984.

4.2 PLD Configuration Memories

A PLD is able to produce different circuits because the routing among its cells is programmable (as in the example of figure 4.2b). The memory that holds such routing in place is called *configuration memory* (see the EEPROM inset in figure 4.4), which (as already mentioned) is nonvolatile in CPLDs and typically volatile in FPGAs. The technologies employed to build such memories are listed in table 4.1; in summary, SPLDs and CPLDs employed mostly EEPROM in the beginning and then Flash memory, while FPGAs employ mostly SRAM, with

nonvolatile memory (Flash or antifuse) used in just a few device families of some companies. Brief comments on each technology are presented next.

The SRAM alternative, used in nearly all FPGAs, has the (minor) disadvantage of losing the configuration when the power is turned off. To solve that problem, FPGA companies sell low-cost nonvolatile memory to store the configuration data, which is retrieved and loaded automatically by the FPGA itself every time the power is turned on.

The EEPROM approach, used in early PLDs, was replaced with Flash EEPROM (or Flash, for short), which has the advantage of requiring just one transistor per cell, against two of EEPROM (Flash does not need the cell-addressing transistor).

A less popular approach is antifuse technology, introduced by Actel (now Microsemi), in the late 1980s. Contrary to regular fuses, which melt and so disconnect two wires when traversed by a large current, an antifuse connects them because the material, after melting, becomes conductive. Another antifuse technology was introduced later by QuickLogic (which refers to it as ViaLink), which connects two metal layers when melted instead of silicon-polysilicon layers of Actel's original antifuse.

Because SRAM, contrary to Flash and antifuse, can be manufactured in conventional CMOS processes, it has lower cost and can take full advantage of process advancements. Consequently, like Xilinx's and Intel's, nearly all FPGAs are now SRAM-based, including the general-purpose ones from Microsemi and QuickLogic.

4.3 PAL and PLA Devices

Figure 4.4 illustrates the general architectures employed in PAL and PLA devices. In (a), a programmable AND array is connected to a nonprogrammable OR gate, while in (b) both are programmable. As already seen in figure 4.2, the little circles indicate programmable connections for which EEPROM was the typical solution.

Because the inputs to the OR gates are also programmable in PLAs, a larger variety of boolean functions could be implemented than in arrays of same size in PALs. On the other hand, the parasitic resistance and capacitance of the associated long lines caused PLAs to be slower (besides consuming more silicon space). Nevertheless, due to the limitations mentioned in section 4.1, neither approach was truly successful.

4.4 GAL Devices

GAL caused a major advancement in the acceptance of PLDs. It employed the PAL architecture plus a "macrocell" at each output. As already mentioned, a successful device in this approach was the GAL16V8 PLD, depicted in figure 4.5a (and also at right in figure 4.1), with sixteen inputs and eight outputs in a twenty-pin package (eight pins are bidirectional).

Details of the macrocell are shown in figure 4.5b. Note the following (very important) improvements over PAL alone: an option for registered or unregistered output (due to the

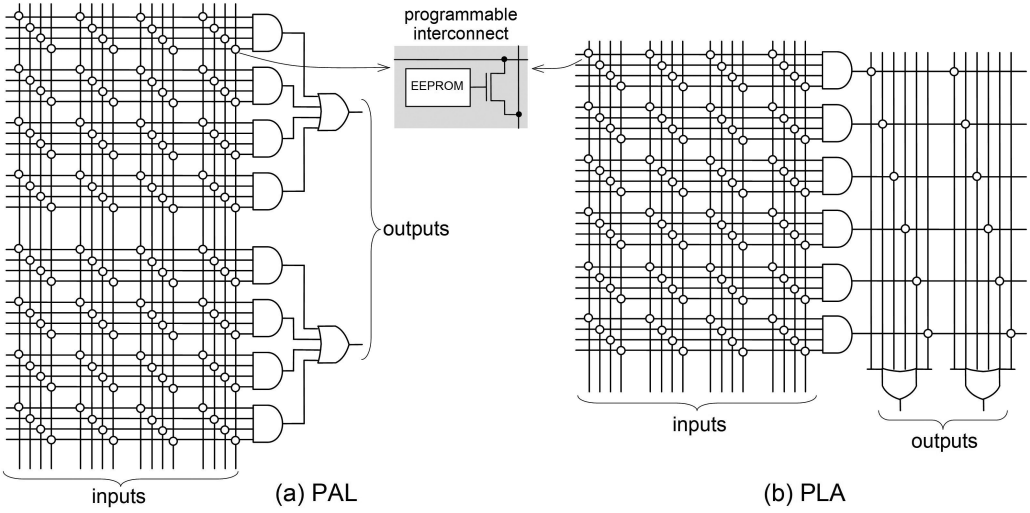

Figure 4.4
General construction principle of (a) PAL and (b) PLA devices.

flip-flop); choice of inverted or non-inverted output (due to the XOR gate); tri-state output; output sent back to the programmable array (so signals do not need to go out of the chip and return when complex functions must be implemented, which lowers the speed and consumes user pads); and direct connection between adjacent cells (with similar benefits).

Even though GAL too is now only of historical interest, understanding it is crucial because, as will be seen, CPLDs and FPGAs are both related to GAL.

4.5 CPLD Devices

As seen in table 4.1, the CPLD approach was introduced by Altera, with its first device (EP300, figure 4.3a) delivered in 1984. A simple way of describing a CPLD is as a relatively small array of moderate-size GAL-like units (figure 4.6), still with nonvolatile configuration memory but with some additional features compared to GAL, like superior internal interconnects, more elaborate clock network, and more I/O options.

An example of true CPLD family (hence employing the general architecture of figure 4.6) is the old MAX3000 series from Altera, constructed with $n = 2$, 4, 8, 16, or 32 GAL-like structures, each with sixteen macrocells (Altera called such structures logic array block, or LAB). Consequently, the total number of macrocells (and therefore of flip-flops) in this family ranged from 32 to 512. Another example of true CPLD family, this time from Xilinx, is the old XC9500 series, constructed with $n = 2$, 4, 6, 8, 12, or 16 GAL-like structures, each with eighteen macrocells.

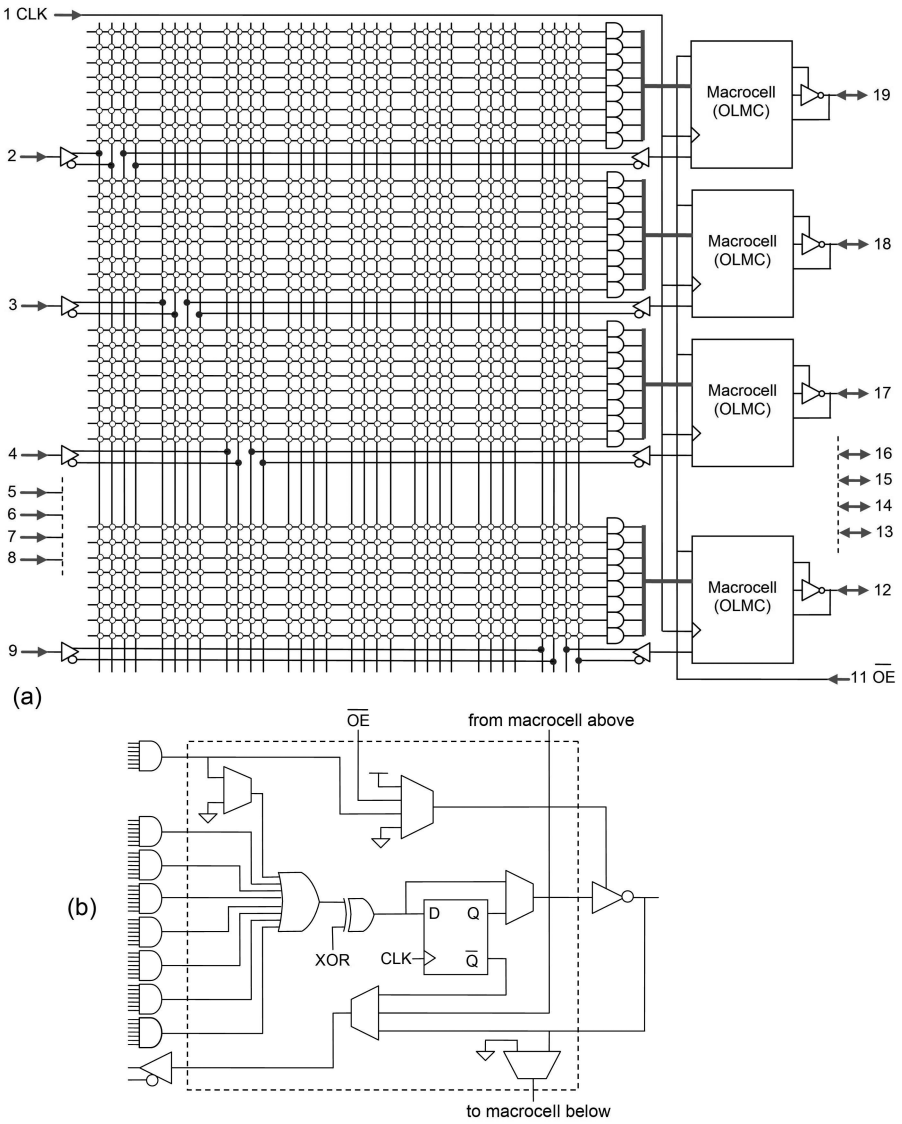

Figure 4.5
The Lattice GAL16V8 device: (a) General structure; (b) Macrocell.

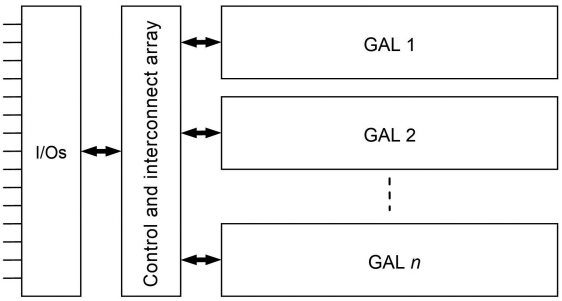

Figure 4.6
Illustration of general CPLD architecture.

A major weakness of PAL-like devices (hence all PLDs seen so far) is that they do not scale well with technology. Note in figure 4.4a that the number of programmable points quadruples when the number of inputs (and the size of the AND layer) doubles; consequently, the device's growth with the transistors' shrink factor is just linear instead of quadratic (for instance, the resulting long, heavily loaded lines impact the speed and power consumption). As a result, the original CPLD approach is now practically gone. Companies either do not manufacture them anymore or use "simplified" FPGAs in their place, with nonvolatile memory employed for the configuration switches (SRAM is replaced with Flash memory); so from a user's perspective everything looks the same. Examples of such "CPLDs" are the Altera MAX II, MAX V, and MAX 10 families, which are now (appropriately) referred to by Intel as FPGAs instead of CPLDs. In summary, these are just simpler, lower-cost FPGAs but with nonvolatile configuration memory (and the programmable AND/OR array replaced with a LUT, of course).

4.6 FPGA Devices

Contrary to CPLDs, the matrix-like (2D) architecture of FPGAs, with the large AND-OR arrays (and their long, slow lines) replaced with compact LUTs, plus the use of SRAM in place of Flash for configuration, allow FPGAs to scale well with technology, fully benefiting from the huge progress in device fabrication processes. (Recall also that an *N*-input LUT can implement any *N*-variable Boolean function, which is not true in general for *N*-input AND-OR arrays.) Indeed, still in the 1990s, FPGAs started replacing (small) ASICs. Eventually, with the ample adoption of FPGAs in the communications infrastructure worldwide (for instance, Cisco is among the largest FPGA consumers), FPGA became definitely a major player, being now present in all sorts of applications, like machine learning, real-time video processing, automotive, data centers, high-performance computing, etc. As a result, the PLDs market now consists essentially only of FPGAs, described in this section.

What was and what is an FPGA Figure 4.7a illustrates how FPGAs looked like in the beginning. Its core, the programmable logic array, was a relatively modest matrix of small GAL-like macrocell clusters, but with LUTs in place of AND-OR arrays for computing the logic functions and with SRAM in place of Flash for configuration. The additional features consisted essentially of some user SRAM blocks, hardware multipliers (for DSP applications), the indispensable PLLs (for clock manipulation, section 2.8), and a selection of basic I/O standards (3.3V LVCMOS, LVDS, etc.).

The current situation of FPGAs is illustrated in figure 4.7b. The programmable logic array is much larger, and so are the original features (listed under the logic array block). But even more importantly, a number of very sophisticated units are now also available (listed at right in figure 4.7b), including high-bandwidth memory (HBM), with the largest throughput in the industry, the fastest transceivers in the market (for internet infrastructure, for example), hard IPs for several communications standards (PCI Express, 10G XAUI, 100G Ethernet, etc.), hard ARM processor cores (for general-purpose computing), and so on. In summary, an FPGA is now a complex "ecosystem," fabricated using the latest technology and targeting all sorts of applications, as mentioned previously.

Below are some major features or advantages of FPGAs:

- They allow the construction of solutions that would be technically inferior or economically unviable with other approaches, notably when units from the list at right in figure 4.7b are also involved.

- Solutions can be tailored exactly as needed in the target application. This helps system optimization, like higher speed through parallelism and lower power consumption by proper circuit architecture and proper clock management. They are also great for doing math!

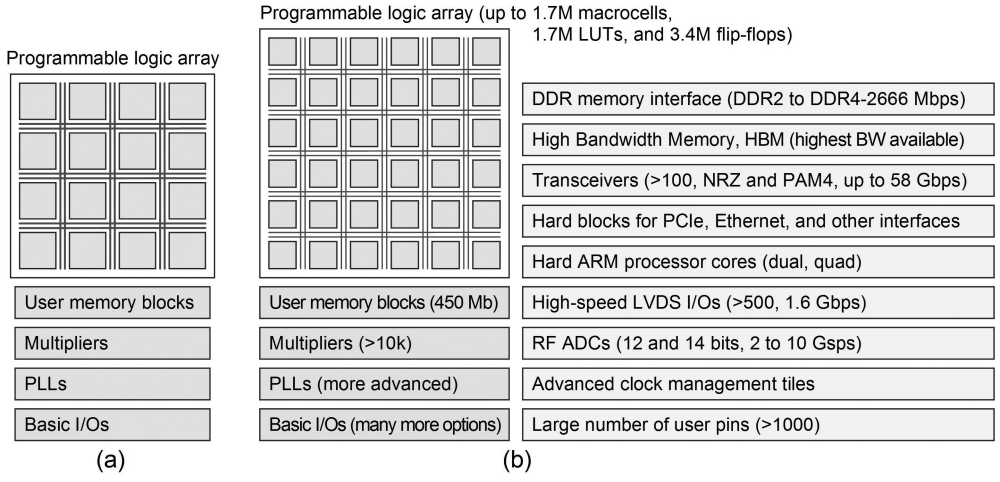

Figure 4.7
(a) FPGAs in the beginning and (b) FPGAs today.

- Designs are developed much faster and do not have the risk and non-recurring engineering (NRE) cost of ASIC-based solutions.

- Contrary to ASICs, designs can usually be easily modified. This might be needed to comply with new standards or different application parameters.

- Contrary to CPLDs, they can easily benefit from progress in device fabrication technology.

- And, surprisingly, they are easily programmed! VHDL and Verilog languages, plus current compilers and simulators, make this task very accessible. Simulation is also much simpler than in ASICs because the hardware has already been validated.

The programmable logic array The programmable logic array block seen in figures 4.7a and b is described next. Figure 4.8 summarizes the current Intel architectural approach—used, for example, in its top FPGA family, Stratix 10. It consists of a large matrix of LAB units, shown in (a). The contents of each LAB are shown in (b), consisting of 10 adaptive logic module (ALM) units. Finally, the contents of each ALM are depicted in (c), where the LUT (with eight inputs, separable into two or more smaller LUTs) can be observed, followed by muxes and flip-flops, which allow the outputs to be registered or unregistered and also allow direct connection between adjacent units—helpful, for example, in the construction of fast arithmetic circuits. Notice the overall similarity of purposes between this circuit and the original macrocell (figure 4.5b).

Figure 4.9 summarizes the current Xilinx architectural approach—used, for example, in its top FPGA family, Virtex UltraScale+. Note in (a) that the general architecture is the same as that in figure 4.8a, just with a different name for the logic block, here called configurable logic block (CLB). The contents of each CLB are depicted in (b), consisting of two Slice units. Finally, the contents of each Slice are shown in (c), where a LUT, muxes, and flip-flops can again be observed in each of the eight circuits that comprise one Slice. For simplicity, these circuits too will be individually (and informally) referred to as "macrocells" in table 4.4, shown later.

In summary, each LAB (figure 4.8) contains ten LUTs and forty DFFs, while each CLB (figure 4.9) contains sixteen LUTs and thirty-two DFFs. These are standard cells, so what changes from one FPGA to another of the same family regarding the FPGA logic array is just the number of LABs or CLBs in the (big) matrix.

The construction of LUTs is illustrated in figure 4.10 (for $N = 3$ inputs), consisting simply of an SRAM memory (of 1-bit words) plus a multiplexer. Because the memory contains 2^N addresses, any N-bit function can be implemented by it. For instance, note that the SOP implemented in figure 4.10 for the given SRAM contents is $y = a \bullet b + a' \bullet b' \bullet c$.

FPGA families We list next details regarding the FPGA families from the two main vendors. Table 4.2 shows all current Intel families, which are (from lowest to highest end) MAX, Cyclone, Arria, and Stratix. It shows also the last two generations (V and 10) for each family and the corresponding technologies. Note that the MAX devices employ old technologies (180nm and 55nm), while Cyclone and Arria employ relatively new technologies (28nm and

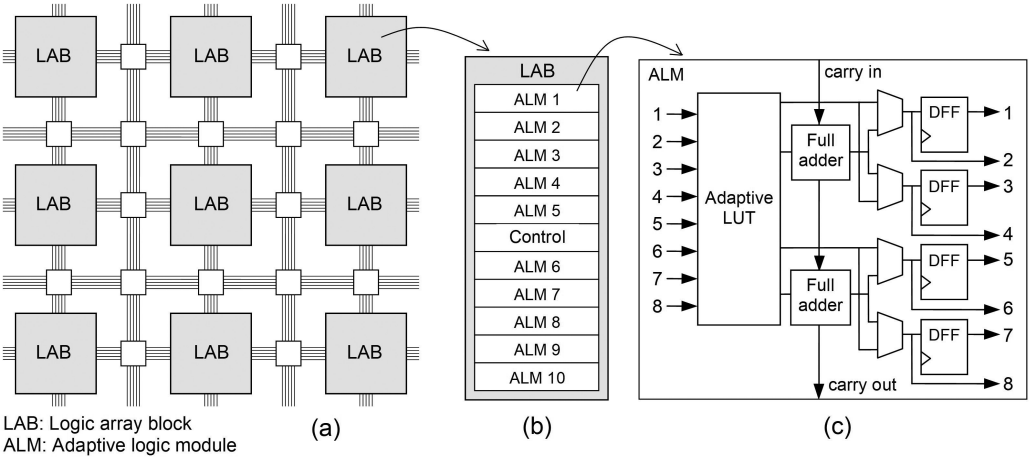

Figure 4.8
Intel architecture for the logic part of Stratix 10 and other FPGAs: (a) FPGA grid (array of LAB blocks); (b) LAB contents; (c) ALM contents.

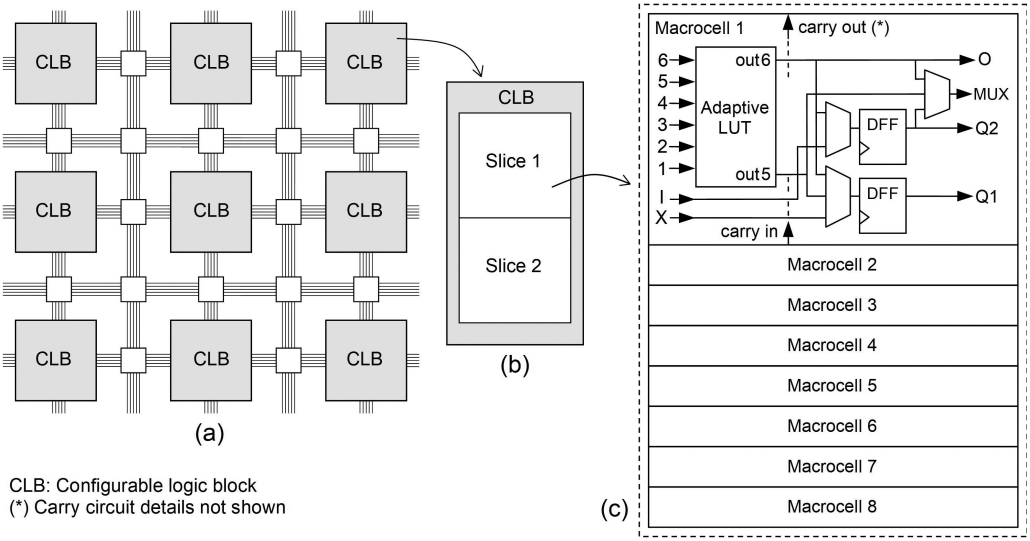

Figure 4.9
Xilinx architecture for the logic part of UltraScale+ and other FPGAs: (a) FPGA grid (array of CLB blocks); (b) CLB contents; (c) Slice contents.

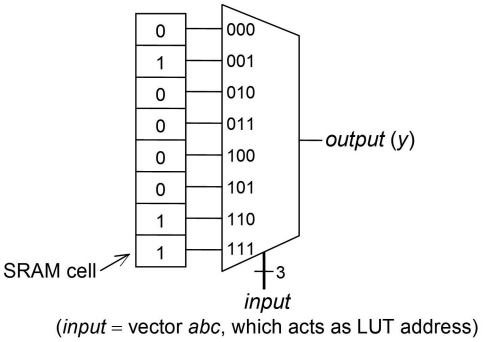

Figure 4.10
LUT construction principle (for 3 inputs, here computing $y = a \bullet b + a' \bullet b' \bullet c$).

Table 4.2
Intel FPGA families

FPGAs / Features	MAX (1)		Cyclone						Arria				Stratix				
	V	10	V				10		V		10		V		10		
	—	—	E	GX GT	SE (2)	SX ST (2)	LP	GX	GX GT GZ	SX ST (2)	GX GT	SX (2)	E	GX GS GT	GX	TX MX (2)	SX (2)
Transceivers	✘	✘	✘	✓	✘	✓	✘	✓	✓	✘	✓	✘	✘	✓	✓	✓	✓
ARM processor	✘	✘	✘	✘	✓	✓	✘	✘	✘	✓	✘	✓	✘	✘	✘	(3)	✓
A/D converter	✘	✓	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘
Technology (nm)	180	55	28				20		28		20		28		14		

(1) With nonvolatile (Flash) configuration memory. (2) Referred to as "SoC FPGAs" (notice ARM processor). (3) Some.

20nm), and Stratix 10 employs an even newer node (14nm). Observe also that the division in series depends mainly on the additional features, of which the presence of transceivers and/or hard microprocessors are the main dividing factors; for example, when a microprocessor (always one or more ARM cores) is built into the FPGA fabric, that FPGA series is referred to as an SoC (system-on-chip) FPGA, because of the easiness with which complete complex systems can be built in that device.

Table 4.3 shows all current Xilinx families, which are (from lowest to highest end) Spartan, Artix, Kintex, and Virtex. It shows also the Zynq family devices, which are referred to

Table 4.3
Xilinx FPGA families

FPGAs / Features	Spartan	Artix	Kintex		Virtex		Zynq (1)				
	7	7	7	UltraSc UltraSc+	7	UltraSc UltraSc+	7000		UltraScale+		
							—	S	CG	EG,EV	RF
Transceivers	✘	✓	✓	✓	✓	✓	(2)	(2)	✘	✘	✓
ARM processor	✘	✘	✘	✘	✘	✘	✓	✓	✓	✓	✓
ARM GPU	✘	✘	✘	✘	✘	✘	✘	✘	✘	✓	✘
A/D converter	(2)	✓	✓	✘	✓	✘	✘	✘	✘	✘	✘
RF A/D converter	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✓
Technology (nm)	28	28	28	20, 16	28	20, 16	28		16		

(1) Referred to as "SoC FPGAs" (notice ARM processor). (2) Some.

Table 4.4
Examples of features in top FPGAs from Intel and Xilinx

Feature	Stratix 10 GX (fig. 4.8)	Virtex UltraScale+ (fig. 4.9)
Name of main logic block (figs. 4.8, 4.9)	LAB (logic array block)	CLB (configurable logic block)
Name of logic sub-block (figs. 4.8, 4.9)	ALM (adaptive logic module)	Slice
Number of LABs or CLBs	12.8k to 186.7k LABs	24.6k to 108k CLBs
Number of ALMs or Slices	10 ALMs per LAB	2 Slices per CLB
Number of "macrocells"	1 per ALM	8 per Slice
Number of LUTs per "macrocell"	1 (total: 128k to 1867k)	1 (total: 394k to 1728k)
Number of flip-flops per "macrocell"	4 (total: 512k to 7470k)	2 (total: 788k to 3456k)
Number of equivalent logic elements	378k to 5510k	862k to 3780k
Logic elements per "macrocell" ratio	2.95	2.19
Block memory (Mb)	30 to 229	115 to 454
Number of transceivers	24 to 96 (17 and 30 Gbps)	32 to 128 (32 and 58 Gbps)
Number of PCI Express ports	1 to 4	1 to 6
Number of user I/O pins	392 to 1640	208 to 832
Core operating voltage	0.8V to 0.94V	0.72V to 0.9V
Maximum reference clock speed (MHz)	800	800
CMOS technology	14nm	16nm

as SoC FPGAs because of the presence of hard microprocessors (again, ARM cores). The last three Xilinx generations are 7, UltraScale, and UltraScale+, which employ technology nodes 28nm, 20nm, and 16nm, respectively.

Table 4.4 shows examples of important features in two top devices of similar categories, the first (Stratix 10 GX) being from Intel and the second (Virtex UltraScale+) from Xilinx. (The reader is invited to check the construction of the first seven lines against figures 4.8 and 4.9.) This table is not a comparison because there are many differences that cannot be listed in a simple table, but it is interesting to see that things are not much different from one to the other. For example, notice that the total numbers of LUTs, flip-flops, and equivalent logic elements overlap nicely. Observe also the similarity in the other features, like the core operating voltage and the maximum reference clock frequency. Finally, observe the variety and abundance of resources (and this table shows just some of them) in this kind of device.

Price of FPGAs We close this section with comments on FPGA prices, which can go from about a dollar up to tens of thousands of dollars. A simple, informal illustration for price ranges is presented in figure 4.11, where some numbers, covering the interval from $1 to

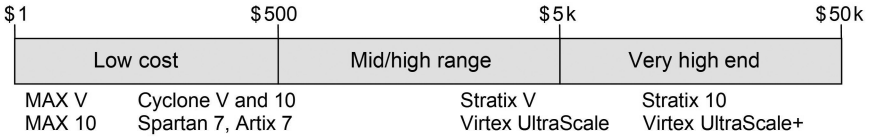

Figure 4.11
An illustrative, informal representation for FPGAs' price ranges.

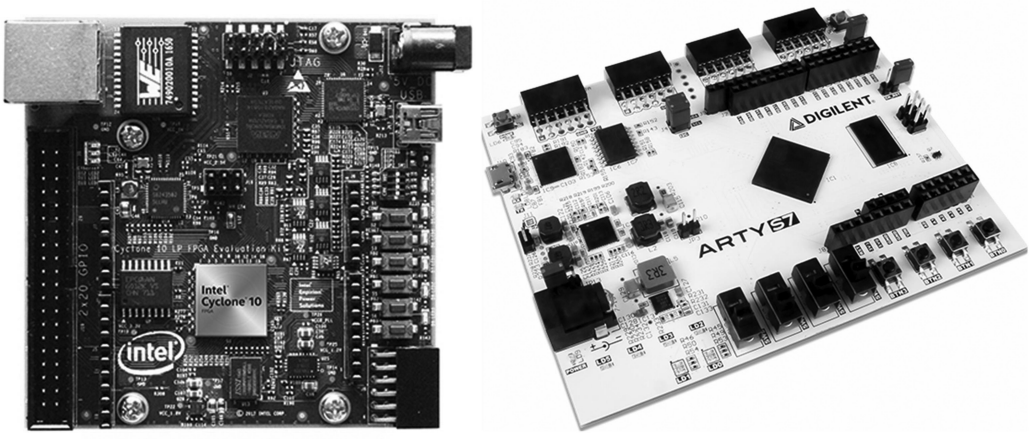

Figure 4.12
Examples of low-cost FPGA development boards (with Cyclone 10 on the left and Spartan 7 on the right).

$50,000, are employed to help visualize the big picture. For example, current Cyclone, Spartan, and Artix devices fall typically in the first range, while current Stratix and Virtex devices usually fall in the upper part of the second range and in the third range. It is important, however, to emphasize that devices in the low-cost range can be very powerful; for example, the initial Cyclone 10 GX FPGA, which contain 31k ALMs (85k equivalent logic elements), six high-speed (12.5 Gbps) transceivers, 6 Mb of user RAM, plus many other features, can be purchased for just over $100. As additional examples, the initial Spartan 7 FPGA, with 6k equivalent logic cells, can be bought by under $12, and MAX 10 FPGAs can cost less than $5.

The FPGA industry also offers a large selection of development boards, with some having special prices for academic use. Two low-cost examples (under $100) are depicted in figure 4.12, for a Cyclone 10 FPGA on the left and for a Spartan 7 FPGA on the right.