

Chapter 1

Basic Knowledge to Understand FPGAs



Toshinori Sueyoshi

Abstract An FPGA is a wonderful digital device which can implement most of the practically required digital circuits with much easier effort than other solutions. For understanding FPGAs, fundamental digital design techniques such as logic algebra, combinational circuits design, sequential circuits design, and static timing analysis are required. This chapter briefly introduces them first. Then, the position of FPGA among various digital devices is discussed. The latter part of this chapter is for 40-year history of programmable devices. Through the history, you can see why SRAM style FPGAs have become dominant in various types of programmable devices, and how Xilinx and Altera (Intel) have grown up major FPGA vendors. Various small vendors and their attractive trials that are not existing now are also introduced.

Keywords Digital circuits' design · Static timing analysis · Programmable logic devices · Field-programmable gate array

1.1 Logic Circuits

Field-programmable gate array (FPGA) is a logic device that can implement user-desired logics by programming logic functions. To understand the structure and design of FPGAs, the basis of logic circuits is briefly introduced in [1, 2].

1.1.1 Logic Algebra

In logic algebra, also called Boolean algebra, all variables can take either the value 0 or 1. Logic algebra is an algebraic system defined by the operators AND, OR, and NOT applied to such logic values (0,1). AND, OR, and NOT are binary or unary operators defined in Table 1.1. Here, we use the symbols “ \cdot ”, “ $+$ ”, and “ $-$ ” for these

T. Sueyoshi (✉)
Kumamoto University, Kumamoto, Japan
e-mail: sueyoshi@cs.kumamoto-u.ac.jp

Table 1.1 Axioms of logic algebra

AND (\cdot)	OR ($+$)	NOT ($\bar{}$)
$0 \cdot 0 = 0$	$0 + 0 = 0$	$\bar{0} = 1$
$0 \cdot 1 = 0$	$0 + 1 = 1$	
$1 \cdot 0 = 0$	$1 + 0 = 1$	$\bar{1} = 0$
$1 \cdot 1 = 1$	$1 + 1 = 1$	

Table 1.2 Theorems of logic algebra

Zero element	$x \cdot 0 = 0, x + 1 = 1$
Neutral element	$x \cdot 1 = x, x + 0 = x$
Idempotent law	$x \cdot x = x, x + x = x$
Complement law	$x \cdot \bar{x} = 0, x + \bar{x} = 1$
Involution law	$\bar{\bar{x}} = x$
Commutative law	$x \cdot y = y \cdot x, x + y = y + x$
Associative law	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$ $(x + y) + z = x + (y + z)$
Distribution law	$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$ $x + (y \cdot z) = (x + y) \cdot (x + z)$
Absorption	$x + (x \cdot y) = x$ $x \cdot (x + y) = x$
De Morgan's laws	$\overline{x + y} = \bar{x} \cdot \bar{y}$ $\overline{x \cdot y} = \bar{x} + \bar{y}$

three logic operators, respectively. AND ($x \cdot y$) is an operation whose result is 1 when both x and y are 1. OR ($x + y$) is an operation whose result is 1 when either x or y is 1. NOT (\bar{x}) is a unary operation giving the inverse of x ; that is, when x is 0 its result is 1, otherwise its result is 0. In logic algebra, the theorems shown in Table 1.2 are satisfied. Here the symbol “=” shows that both sides are always equal or equivalent. By exchanging logic value 0 into 1, and operation AND into OR, the equivalent logic system is formed. This is called a dual system. In logic algebra, if a theorem is true, its dual is also true.

1.1.2 Logic Equation

A logic equation consists of an arbitrary number of logic operations, logic variables, and binary constants, separated by parentheses if needed to represent the order of computation. When a logic equation is formed with n logic variables $x_1, x_2, x_3, \dots, x_n$, its result is either 0 or 1 according to the procedure represented with an equation by substituting 0 or 1 in the variables (2^n in total), following

an arbitrary combination. That is, a logic equation represents a logic function $F(x_1, x_2, x_3, \dots, x_n)$. If the priority is not defined by parentheses, AND is given a higher priority than OR. The AND operator “.” is often omitted. Arbitrary logic functions can be represented by logic equations, but there are a lot of logic equations for representing the same logic function. Thus, by giving some restrictions, a logic function can be 1 to 1 corresponding to a logical equation. It is called a standard logic form. A single variable or its inverse is called a literal. Logical AND of literals which does not allow duplication of itself is called a product term. The sum-of-products form is a logic equation only with logical OR of products. A product term formed from all literals is called a minterm. A sum-of-products only containing minterms is called sum-of-products canonical form. A product-of-sums is a dual of a sum-of-products. A maxterm is formed with OR of literals for all inputs without duplication. A product-of-sums canonical form is formed only with maxterms.

1.1.3 Truth Table

Truth tables and logic gates (shown later) are representations of logic functions other than logic equations. The table which enumerates all combinations of inputs and corresponding outputs is called the truth table. In the case of combinational circuits, a truth table can represent all combinations of inputs, and so it is a complete representation of the circuit. The specification of a combinational circuit is defined in the form of a truth table. For n inputs, the number of entries of the truth table is 2^n . The corresponding output is also added to the entry.

A truth table is a unique representation of a logic function. Although a logic equation also represents a unique logic function, a logic function can be represented with various equivalent logic equations. A straightforward implementation of a truth table is called lookup table(LUT), which is used in major FPGAs.

From a truth table, two canonical forms such as sum-of-products or product-of-sums can be induced. The sum-of-products canonical form is derived by making minterms of input variables when the corresponding output is 1, and then applying the OR operator. On the other hand, the product-of-sums canonical form is derived by making maxterms of inverted input variables when the corresponding output is 0, and then applying the AND operator. An example of making a logical equation from a truth table is shown in Fig. 1.1.

1.1.4 Combinational Circuits

A logic circuit can be classified into combinational or sequential whether it includes memory elements or not. In combinational circuits, which do not include memory elements, the output is defined only with current input values. Combinational circuits have a given number of inputs and outputs and consist of logic gates computing basic

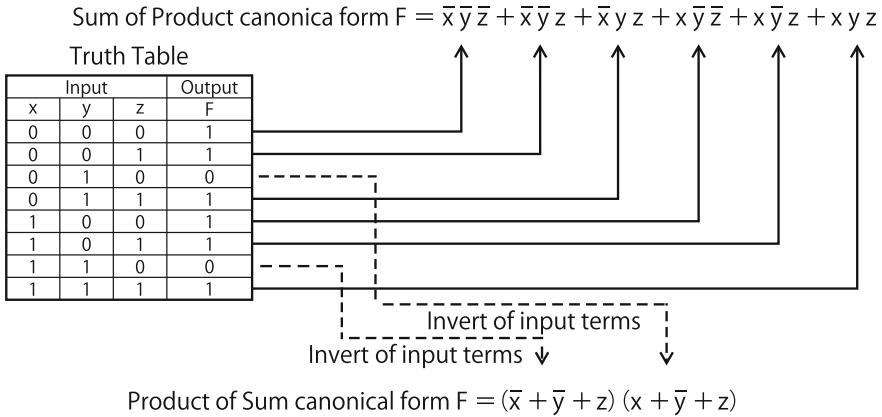


Fig. 1.1 An example of making a logical equation

logical functions such as AND, OR, and NOT connected with wires. These logic gates correspond to three basic operations: Logical and, logical or, and logical not are called AND gate, OR gate, and NOT gate, respectively. Additionally, there are gates for well-known binary operations: NAND gate, NOR gate, and EXOR gate. NAND gate, NOR gate, and EXOR gate compute inverted AND, inverted OR, and exclusive OR, respectively. Figure 1.2 shows their symbols (MIL symbols), truth tables, and logical equations. \oplus is used for the symbol for logic operation of exclusive OR. The table shows two inputs gates for binary operations, while gates with more than three inputs are also used. CMOS used in most of the current major semiconductor LSIs often includes compound gates like OR-AND-NOT or AND-OR-NOT.

Any logic circuits can be represented with the sum-of-products canonical form. Thus, any combinational circuits can represent any arbitrary logic function by a NOT-AND-OR form. This is called AND-OR two-stage logic circuits or AND-OR array. AND-OR two-stage logic circuits are implemented by a programmable logic array (PLA).

1.1.5 Sequential Circuits

Logic circuits including memory elements are called sequential circuits. While combinational circuits decide their outputs only with the current inputs, outputs of sequential circuits are not fixed with only current inputs. That is, the prior inputs influence the current output.

Sequential circuits are classified into synchronous and asynchronous. In synchronous sequential circuits, outputs and internal states are changed synchronously following a clock signal, while asynchronous sequential circuits do not have a clock signal. Here, only synchronous circuits used in most FPGA design are introduced.







Operation	Symbol	Function	Equation															
AND		<table border="1" data-bbox="519 208 652 296"> <thead> <tr> <th>x</th> <th>y</th> <th>z</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	x	y	z	0	0	0	0	1	0	1	0	0	1	1	1	$z = x \cdot y$
x	y	z																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		<table border="1" data-bbox="519 322 652 411"> <thead> <tr> <th>x</th> <th>y</th> <th>z</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	x	y	z	0	0	0	0	1	1	1	0	1	1	1	1	$z = x + y$
x	y	z																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
NOT		<table border="1" data-bbox="519 437 652 490"> <thead> <tr> <th>x</th> <th>z</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> </tr> </tbody> </table>	x	z	0	1	1	0	$z = \bar{x}$									
x	z																	
0	1																	
1	0																	
NAND		<table border="1" data-bbox="519 516 652 605"> <thead> <tr> <th>x</th> <th>y</th> <th>z</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	x	y	z	0	0	1	0	1	1	1	0	1	1	1	0	$z = \overline{x \cdot y}$
x	y	z																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		<table border="1" data-bbox="519 622 652 710"> <thead> <tr> <th>x</th> <th>y</th> <th>z</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	x	y	z	0	0	1	0	1	0	1	0	0	1	1	0	$z = \overline{x + y}$
x	y	z																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
EXOR		<table border="1" data-bbox="519 728 652 816"> <thead> <tr> <th>x</th> <th>y</th> <th>z</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	x	y	z	0	0	0	0	1	1	1	0	1	1	1	0	$z = x \cdot \bar{y} + \bar{x} \cdot y = x \oplus y$
x	y	z																
0	0	0																
0	1	1																
1	0	1																
1	1	0																

Fig. 1.2 Basic logic gates

Outputs of synchronous circuits are determined both by the inputs and the memorized values. That is, states depending on past inputs value influence the current outputs in sequential circuits. They are represented with a model of finite-state automaton as shown in Fig. 1.3. Figure 1.3a shows Mealy finite-state machine, while Fig. 1.3b illustrates Moore finite-state machine. Outputs are determined by the internal states and inputs in Mealy machine, while in Moore machine, they are only depending on their internal states. Compared with Mealy machine, Moore machine can decrease the size of the circuits, since a smaller number of states are required for the target function. However, outputs are directly influenced by the change of input signals and so the signal can glitch because of the difference of gate or wiring delay which may lead to unpredicted hazards. On the other hand, Moore machine can directly use states to generate outputs; thus, high-speed operation without hazard can be achieved. The circuits' size can become large because of the increasing number of states.

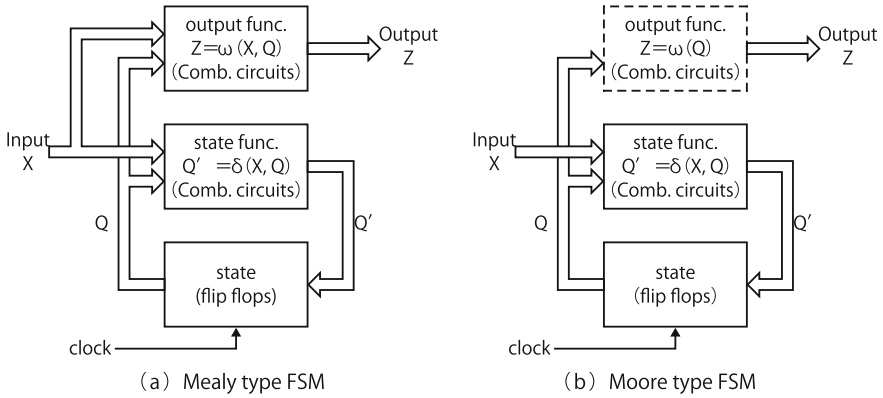


Fig. 1.3 Mealy machine (a) and Moore machine (b)

1.2 Synchronous Logic Design

In synchronous logic design, all states of the system are idealized to change synchronously with a clock so as to make the design simple. It is a fundamental design policy used in FPGAs.

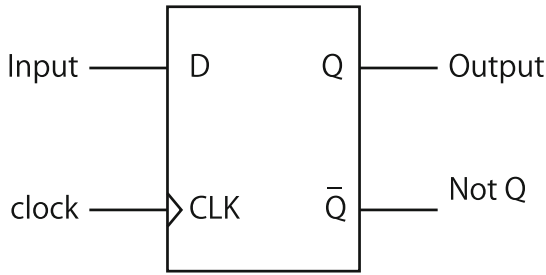
1.2.1 Flip-Flop

A one-bit memory element called flip-flop (FF) is used as a memory element in sequential circuits. D-flip-flops (D-FFs), embedded in basic blocks of an FPGA, change their outputs at the rising edge (or falling edge) of the clock. That is, they are edge-trigger type. The symbol and truth table of a D-FF are shown in Fig. 1.4. Here, it stores the value at D input at the rising edge of the clock and outputs it at Q-output.

1.2.2 Setup Time and Hold Time

A CMOS D-FF has a master-slave structure consisting of two latch (loop) circuits, each of which uses a couple of transfer gates and inverters (NOT gates), as shown in Fig. 1.5. A transfer gate takes the role of a switch, and it changes to on/off according to CLK. The front-end latch stores the input with the inverse of the clock in order to avoid the hazard appearing just after the change of the clock. The operation of a D-FF is shown in Fig. 1.6.

When $CLK = 0$ (master operating), the D input is stored into the front-end latch, and the back-end latch holds the data of the previous cycle. Since the transfer gate



D	CLK	Q	\bar{Q}
X	L	Qn	$\bar{Q}n$
L		L	H
H		H	L
X	H	Qn	$\bar{Q}n$
X		Qn	$\bar{Q}n$

X : Don't care
(Both L and H are OK)

: L → H
(Up edge)

: H → L
(Down edge)

Fig. 1.4 D-Flip-flop

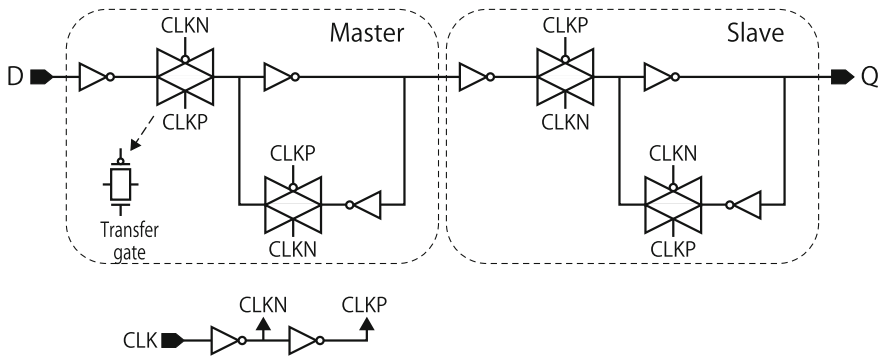


Fig. 1.5 Master-slave D-Flip-flop

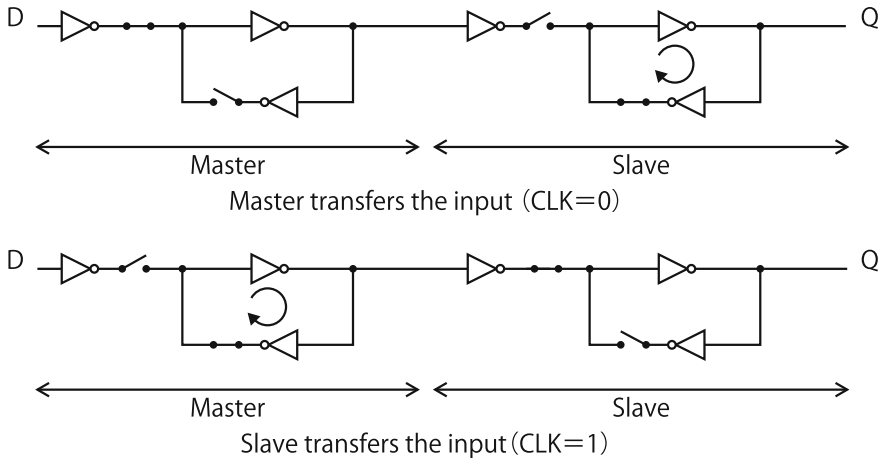


Fig. 1.6 Operation of master–slave D-Flip-flop

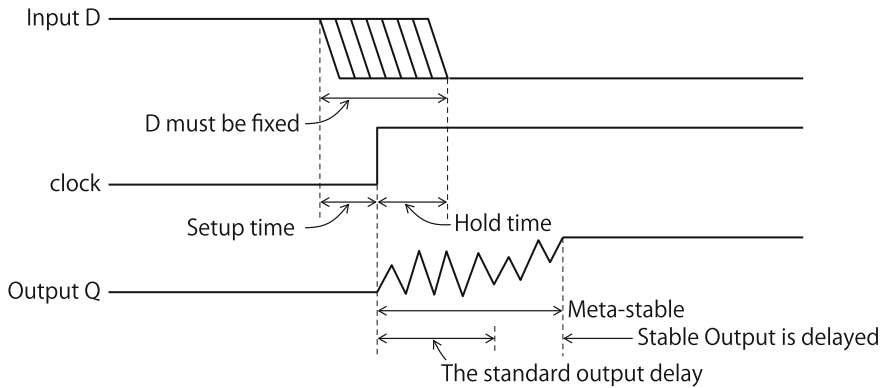


Fig. 1.7 Setup time and hold time

connecting the front-end and back-end is cut off, the signal is not propagated. When CLK=1 (slave operating), the data stored in the front-end is transferred to the back-end. At that time, the signal from D input is isolated. If the data is not well propagated between both inverters of the front-end loop when CLK becomes 1, the signal may become unstable, taking an intermediate level called meta-stable, as shown in Fig. 1.7. Since the meta-stable continues longer than the delay time of a gate, the data might be stored incorrectly. To prevent this, the restriction of setup time must be satisfied.

Also if the D input is changed just after CLK=1 and the gate at the D input is cut off, illegal data can be stored or unstable state can occur. In order to avoid it, the restriction of hold time must also be satisfied.

For all the FFs in an FPGA, a timing limitation such as setup time and hold time should be defined for correct operation.

1.2.3 Timing Analysis

Translating register-transfer level (RTL) description in hardware description language (HDL) into a netlist (wiring information between gates) is called logic synthesis. The design step for fitting circuits of the netlist into an FPGA implementation is called “place & route.” In an FPGA, an array of predefined circuits and interconnections between them are provided on a chip. The FPGA design stages fix where the circuits translated by the synthesis are located and how to connect them.

In order to verify the correct operation of the designed circuits, not only the function (logic) must be ensured, but also the timing constraints have to be satisfied. In the design of FPGAs, the circuits must be evaluated through the logic synthesis and the place & route. The correctness of the logic is verified by RTL simulations. Since dynamic timing analysis by post place & route simulations with delay requires a large amount of computation time, static timing analysis (STA) is used instead. STA can be executed only with a netlist, and comprehensive verification can be done. Moreover, since it basically traces the circuits only once, the execution speed of the STA is high. It is commonly used in other EDA tools besides FPGAs, to certify whether the design works at a required speed to cope with recent increasing size of target circuits.

Timing analysis includes setup and hold time analysis for timing verification. It verifies whether the delay of the design implemented on FPGA satisfies the timing restrictions. Wiring delay depends on the mapping and routing of the design to the resource of the FPGA, that is, the compilation result of the place & route tool. The design is relatively easy if the performance and number of gates of the target FPGA are large enough, but if the size of the design uses almost all of its resources, the place & route can require a considerable amount of time. The delay of the elements and interconnections of all paths must be checked including the timing margin so as to certify whether the setup time and hold time are satisfied.

1.2.4 Single-Clock Synchronous Circuits

Since FPGAs have a large flexibility in place & route, synchronous circuits are widely used; thus, the target of STA is focused on synchronous circuits. Although the STA is fast, the target circuits can have a certain limitation. That is, the start point and the end point of the delay analysis must be a FF with the same clock input, and the delay between them is accumulated. The transient time of the signal is different since the wiring delay is not the same. Thus, an FPGA design receives all input data at FFs and outputs all data through FFs, as shown in Fig. 1.8. In other words, the system’s circuits work with the same edge of the same clock. Inverse clock or reverse edge is basically not allowed, and such a single-clock system is recommended.

The precondition of the synchronous design is to deliver the clock to all FFs at the same timing. The wiring length of real clock signals is often long, and so the

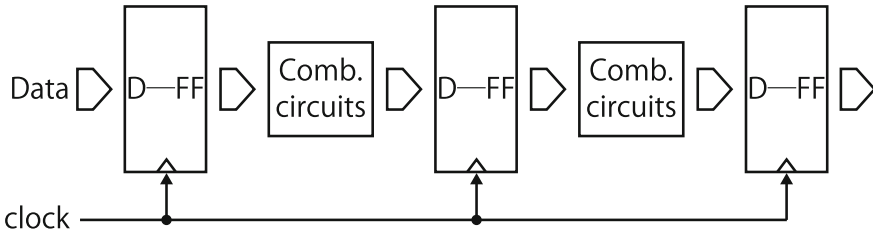


Fig. 1.8 Single-clock system

wiring delay becomes large. Also, the fan-out influences the delay time. Because of their influence, clock timing is slightly different for each FF. This effect is called the clock skew. Jitter is a fluctuation of the clock edge by the variance of the oscillator or distortion of the wave. In order to deliver the clock at the same time, such skew or jitter must be managed under a certain bound.

The clock skew influences the cycle time as well as the delay of logic gates. That is, the most important step in integrated circuits is the clock tree design. In the case of FPGAs, the hierarchical clock tree is already embedded with global buffers providing a high drive capability in the chip to distribute a clock to all FFs, and thus, a low skew clock distribution can easily be achieved. Compared with ASIC designs, in FPGAs, the design step for clock distribution is easier.

1.3 Position and History of FPGAs

Here, the position of FPGA in the logic devices is introduced, and then about 30 years of history of development are reviewed [3, 4].

1.3.1 The Position of FPGA

Logic devices are classified into standard logic devices and custom ICs, as shown in Fig. 1.9. In general, the performance (operational speed), density of integration (the number of gates), and flexibility of given design are advantageous for devices close to custom ICs. On the other hand, non-recurring engineering (NRE) cost for IC designs becomes high and the turnaround time (TAT) from an order to its delivery becomes longer.

Custom ICs are classified into full-custom and semi-custom ICs. The former uses cells designed from scratch, and the latter uses standard cells. Semi-custom ICs are further classified into various types depending on how the NRE cost and TAT are reduced. A cell-based ASIC uses a standard cell library. On the other hand, a gate array uses a master-slice consisting of an array of standard cells, and only steps for

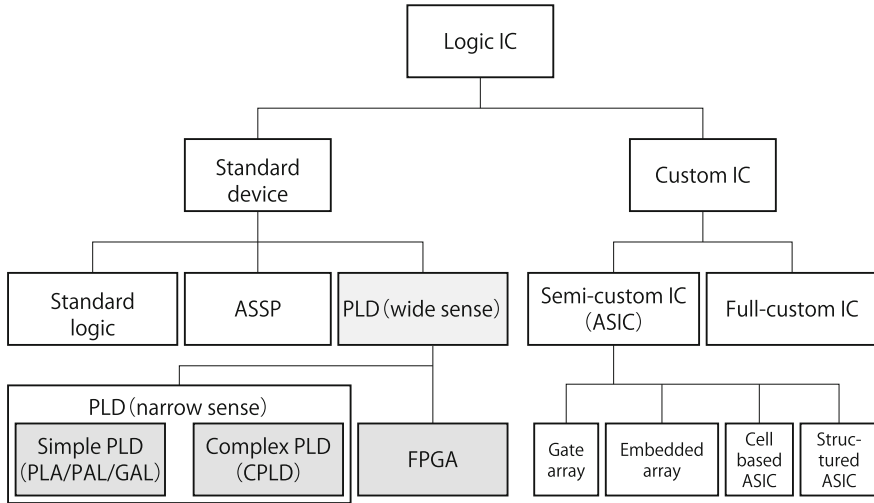


Fig. 1.9 FPGA position in semiconductor devices

wiring follow. An embedded array is a compromise method of cell-based and gate array. The structured ASIC includes standard functional blocks such as SRAM and PLL with a gate array part so as to minimize the design cost. They focus on reducing the NRE cost and shortening the TAT.

Unlike application-specific standard parts (ASSPs), a programmable logic device (PLD) can realize various logic circuits depending on a user program. PLDs have been widely developed by introducing the properties of field programming and freedom of reconfiguring. An FPGA is a PLD which combines multiple logic blocks in the device for a high degree of programming. Since it has a gate array like a structure, it is called field-programmable gate array. FPGA can be mass-produced with a blank (initial) state. So, it can be treated as a standard device from semiconductor vendors, but it can also be considered as an easy-made ASIC with a small NRE cost, and without any mask fee.

More than 40 companies have tried to join the FPGA/PLD industry so far. Here, the history is introduced for each of the era shown in Tables 1.3 and 1.4.

1970s (The Era of FPLA and PAL)

The PLDs started from a programmable AND-OR array with a similar structure to a programmable read only memory (PROM).

The circuit information was stored in memory elements. In 1975, Signetics Co. (became later Philips, and now it is now known as NXP Semiconductors) sold a fuse-based programmable field-programmable logic array (FPLA). Then NMI Co. announced programmable array logic (PAL) which used a simpler structure but

Table 1.3 History of FPGA (1)

Age	Max. num of gates	Represented devices	Features	Companies
1970s	10s–100	Field-programmable logic array (FPLA)	User-programmable, fuse-type, one-time	Signetics (join to Philips, now NXP Semiconductors)
		Programmable array logic (PAL)	Fixed OR-array, high-speed, bipolar, one-time	NMI (join to Vantis, now Lattice Semiconductors)
1980s	100s	Genetic array logic (GAL)	Low-power CMOS electric erasable EEPROM	Lattice
	100s–1000s	FPLA (field-programmable logic array)	Array of CLB interconnect I/O cells are programmable	Xilinx
		Complex programmable gate array (CPLD)	Multiple AND-OR Arrays, high density, high capacity and high-speed	Altera, AMD Lattice
		Anti-fuse FPGA	High-speed, non-volatile but one-time	Actel, Quick Logic
1990s	1000s–Million’s	SRAM-based FPGA	New products to glowing SRAM-based FPGA (Flex, ORCA, VF1, AT40K families)	Altera, AT & T (Lucent), AMD (Vantis, Lattice) Atmel
		Flash-based FPGA	Non-volatile electrically re-programmable	GateField
		BiCMOS FPGA	High-speed ECL using BiCMOS FPGA(DL5000 family)	DynaChip

achieved high-speed operation using bipolar circuits. PAL was widely used by taking a fixed OR array and bipolar PMOS. On the other hand, it consumed a large amount of power, and the erase and re-program were not allowed.

1980s

(1) The appearance of GAL, EPLD, and FPGA:

In 1989, low-power and erase/re-programmable CMOS EPROM-/EEPROM-

Table 1.4 History of FPGA (2)

Age	Max. num of gates	Represented devices	Features	Companies
2000s	1 Million–15 millions	Million-gate FPGA, SoPD (System on Prog. Device)	Processor-core HardIP, SoftIP, Multi-input LB, Hi-speed I/F, Multi-platform	Altera, Xilinx
		Startup vendors’ FPGA Ultra-low-power FPGA, High-Speed ASYNC FPGA, Dynamic Reconf. FPGA, A large-scale FPGA, Monolithic 3D FPGA	Low leak process or power gating, Data tokens transfers, Virtually 3D DRP tech., Scalable wire structure, Amorphas Si TFT techniques	Silicon Blue, Achronix, Tabula, Abound Logic, Tier Logic
2010s	20 Millions (28 nm)–50 millions (20 nm)	28 nm gen. FPGA 20 nm gen. FPGA 16/14 nm gen. FPGA, New gen. SoPD (SoC FPGA), Dynamic PR FPGA, 3D (2.5D), FPGA for Automobile Optical FPGA	TSMC 28 nm, 20 nm, 16 nm FIN FET Intel’s 14 nm FIN FET, ARM embedded Zynq, Cyclone V SoC Standard support of PR TSV, SiP AEC-Q100 standard ISO-26262 standard Vivado HLS OpenCL	Altera, Xilinx
		Oligopoly	Withdraw of Quicklogic and Atmel. Termination of new FPGA vendors. Frequent M & A	Big 4 vendors Xilinx, Altera Lattice, Actel
		Industry consolidation	Data center, IoT Big data analysis, machine learning, network virtualization, high-performance computing	Microsemi acquired Actel Lattice acquired Silicon Blue Intel acquired Altera

based PLDs were pushed into the market from various vendors. In this era, Japanese semiconductor companies grew rapidly using DRAM technologies, while US traditional big vendors were relatively in depression. Thus, the leading companies were mostly newly developed US venture companies. Various PLD architectures including Lattice’s (established in 1983) generic array logic (GAL) and Altera’s erasable PLD (EPLD) were developed, and especially GAL was popularly used. It was upper compatible of PAL with the fixed OR array, and a CMOS-based EEPROM was adopted as a programmable element. PLDs with

a single AND-OR array such as GAL, FPLA, and PAL, described before, are called simple PLD (SPLD). Their number of gates is about 10s–100s. Advances in semiconductor technologies allowed to implement more gates than for GALs, since increasing the size of a single AND-OR array was not efficient. So, as a flexible large PLD, FPGA and CPLD were introduced.

Xilinx (established in 1984), the first to design FPGAs, was a venture company established by Ross H. Freeman and Bernard V. Vonderschmitt who had spun out from Zilog. Freeman adopted a basic logic cell with a combination of 4-input 1-output LUT and FF and commercialized a practical FPGA (XC2064 series) based on CMOS SRAM technologies. William S. Carter, who joined a little later, invented a more efficient interconnection method to connect logic cells. Their innovations are known as famous patents in FPGA: Freeman's patent and Carter's patent. Ross H. Freeman was included to the US National Inventors Hall of Fame in 2009 for his innovation with FPGAs. Xilinx's FPGA (the product name was then LCA) was highly flexible where erase/re-programming can be done by using CMOS SRAM technology, and its power consumption was low. Based on the advanced research of Petri-net at the Massachusetts Institute of Technology (MIT), Concurrent Logic (now Atmel) commercialized an FPGA with a partial reconfigurable capability. Also, based on the research on virtual computer at Edinburgh University, Algotronix (now part of Xilinx) announced a flexible partial reconfigurable FPGA. The former was Atmel's AT6000, and the latter was Xilinx's XC6200. They are the origins of the dynamically reconfigurable FPGAs.

(2) The second half of the 80s (Appearance of anti-fuse FPGAs and CPLD):

In the latter half of the 1980s, in order to accelerate the implementation density and operational speed, anti-fuse FPGAs, which do not allow erase/re-program, appeared. On the other hand, since the early FPGAs could not achieve the desired performance, other structures of large-scale PLDs were investigated. Altera, AMD, and Lattice, which had produced AND-OR array PLD, developed a large-scale PLD by combining multiple blocks of AND-OR PLDs. They were called complex PLD (CPLD) later. While their flexibility and degree of integration could not compete with FPGAs, CPLDs had the advantage of high-speed design, and re-writable non-volatile memory devices could be easily introduced. Thus, CPLD was a representative of large-scale PLDs comparable to FPGAs until the early 1990s. However, from the late 1990s, since the degree of integrity and speed of SRAM-based FPGAs were improved rapidly, CPLDs started to be considered as economical small devices.

(3) Venture companies until the 80s:

FPGA industry has been mainly driven by various venture companies. Xilinx, which first commercialized FPGAs, was established in 1984. Altera and Lattice were established almost the same year SPLDs were commercialized, and then entered the FPGA industry. Actel is also a venture company established slightly later. They had grown as the big-four vendors in the FPGA industry. QuickLogic appeared later, and these five vendors lead the industry. From major semiconductor companies, only AT&T (former Lucent and Agere, whose FPGA project

was sold to Lattice), and Motorola (Freescale) entered the industry. AT&T was the second source of Xilinx, and Motorola developed products with a license from Pilkington. Consequently, there were no major semiconductor companies which developed FPGAs from scratch. TI and Matsushita (now Panasonic) tried to enter the FPGA industry in cooperation with Actel. Infineon and Rhom started FPGA business with Zycad (Gatefield, later); however, all of them have withdrawn from this initiative.

(4) Japanese semiconductor vendors and major semiconductor vendors:

Venture companies established in the 80s such as Lattice, Altera, Xilinx, and Actel are all fabless maker, meaning that they have no facility for producing semiconductors. Therefore, they relied their fabrication on Japanese semiconductor vendors; for example, Xilinx and Lattice relied on Seiko Epson, and Altera relied on Sharp. Actel had a comprehensive contract with TI and Matsushita including fabrication. In 1990s, GateField, which developed flash FPGAs, had a comprehensive contract with Rohm. However, recently, most FPGAs are produced by Taiwan semiconductor companies such as TSMC and UMC which provide an advanced CMOS technology. Since Japanese major semiconductor vendors focused on DRAM as a standard product and on gate arrays as custom products, they had no intention to enter the PLD industry.

US major semiconductor companies such as TI and National Semiconductor, which focused on logic LSI and memory ICs, had already been part of the market of bipolar AND-OR array. They also tried to produce CMOS EPROM- or EEPROM-based PLDs. However, they could not compete against the aggressive venture companies which developed new architectures, and most of them ceased their activities in the PLD industry. Although AMD purchased MMI in 1987 and aggressively developed new CPLD architectures, it split the activity to Vantis and sold it to Lattice in 1999 in order to concentrate on CPU business.

1990s

(1) Increasing the size of FPGAs:

In the 1990s, both Xilinx and Altera increased the size (gate number) of their FPGAs by improving and extending their XC4000 and FLEX architectures. The size was increased from 1000s to 10,000 in the early 1990s and reached to a hundred thousand in the late 1990s. A large rapid prototyping platform using large-scale FPGAs, as shown in Fig. 1.10, was then developed. The FPGA industry grew up rapidly, and AT&T, Motorola, and Vantis entered SRAM-based FPGAs in these years. In Japan, Kawasaki Steel, NTT, and Toshiba tried to produce their own devices, but eventually products were never released.

It is said that some vendors gave up the production because of the risk of conflict with Xilinx's basic patents (Freeman's patent and Carter's patent). Regarding Altera's PLD products (FLEX family), there has been a long dispute whether they infringe Xilinx's patents. The case was settled in 2001, and after that, Altera



Fig. 1.10 A rapid prototype using 12 FPGAs

was able to start using the word “FPGA,” too. Some novel FPGAs appeared in the late 90s. For example, GateField (currently acquired by Actel then Microsemi) announced FPGAs with non-volatile yet erase/re-writable flash memory, and DynaChip commercialized high-performance FPGAs with ECL logic using Bi-CMOS process. After the late 1990s, the degree of integration and operational speed of FPGAs rapidly increased, and the difference with CPLDs widened. From that era, FPGAs became a representative device of PLD. On the other hand, since the performance gap between semi-custom LSIs such as gate array or cell-based ICs has been drastically reduced, FPGAs expanded into the semi-custom (especially gate array) market.

Through the 1990s, general-purpose FPGAs pursued their growth, and the mixed integration of MPUs and DSPs was an inevitable result. In 1995, Altera’s FLEX10K integrated memory blocks to expand its application, and phase-locked loop (PLL) to manage high-speed clock signals was also provided. From this era, FPGAs were mass-produced and widely spread. In 1997, the logic size reached 250,000 gates and the operational speed increased from 50 to 100 MHz. In 1999, Xilinx announced an FPGA with a new architecture called Virtex-E, and Altera announced the APEX20K for the coming million-gate era.

(2) New companies in the 1990s:

In the early 90s, a few companies including Crosspoint, DynaChip (Dyna Logic), and Zycad (Gatefield) entered the industry. Zycad had had a certain experience

as an EDA vendor based on logic emulators, but sold this project later. In this era, four major leading companies such as Xilinx, Altera, Actel, and Quicklogic grew steadily. Crosspoint and DynaChip canceled their projects. Crosspoint was established in 1991, and it was the last established vendor of anti-fuse FPGAs. In 1991, it applied the basic patents and announced its products, but closed in 1996. Crosspoint FPGA used amorphous silicon anti-fuse for through-holes between aluminum layers to form user-programmable gate array. The finest logic cells with a pair of transistors were used to realize similar density of integrity as gate arrays. This type of programmable devices never appeared again. In the late 1990s, Xilinx and Altera became so strong that there were almost no new FPGA vendors. Instead, there were a lot of venture companies for dynamically reconfigurable coarse-grained reconfigurable devices. However, most of them have vanished, and none has achieved a big success.

2000s

- (1) Million-gate era, and becoming a system LSI:
In the 2000s, FPGA became a system LSI. Altera’s soft-core processor Nios is a processor IP supported by the vendor. Altera also announced “Excalibur,” the first FPGA with hard-core processor (Fig. 1.11). Excalibur integrated an ARM

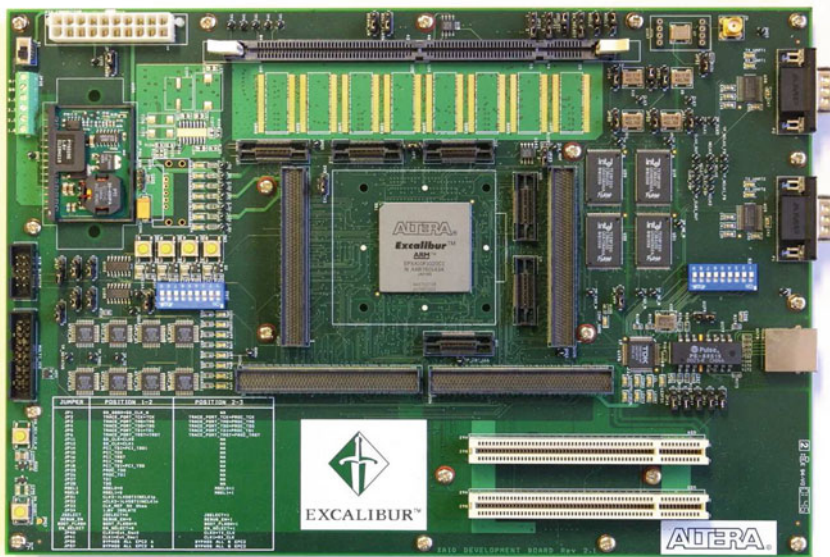


Fig. 1.11 First SoC FPGA excalibur

processor (ARM922 with peripherals) and an FPGA into a chip. On the other hand, Xilinx supported MicroBlaze as a soft-core processor and commercialized a PowerPC embedded FPGA core (Virtex II Pro). For a system LSI, a high-performance interface is important. So, FPGAs also provided serializer–deserializer (SERDES) and low-voltage differential signal (LVDS) for high-speed serial data transfer. In order to cope with the computational performance requirements for image processing, dedicated computation blocks of multipliers or multipliers + adders were embedded. Many-input logic blocks with high performance and density of integration were also introduced. However, such hard IPs are wasteful if unused, so multi-platform (or subfamily) with various product lineups for different target application were provided. For example, Altera introduced new products every two years: Stratix (2002, 130 nm), Stratix II (2004, 90 nm), Stratix III (2006, 65 nm), and Stratix IV (2008, 40 nm). In 1995, FLEX10K supported 100,000 gates and worked with a maximum of 100 MHz clock. In 2009, Stratix IV E had 8400,000 gates + DSP blocks corresponding to 1,5000,000 gates and was operational with a 600-MHz internal clock. The number of gates was multiplied 150 times. In the case of Xilinx, Virtex II Pro (2002, 130 nm) changed every two years with Virtex-4 (2004, 90 nm), Virtex-5 (2006, 65 nm), and Virtex-6 (2009, 49 nm). During that era, logic IC process evolved every 2 years and FPGAs quickly followed that trend.

(2) New vendors in the 2000s:

Two basic patents, Freeman’s patent and Carter’s patent which had been a great barrier for newcomers, expired in 2004 and 2006, respectively. Some new vendors then took the opportunity and entered the FPGA industry. SiliconBlue Technologies, Achronix Semiconductor, Tabula, Abound Logic (former M2000), and Tier Logic entered at that time.

SiliconBlue focused on the power consumption which is the weak point of conventional FPGAs and announced ultra-low-power iCE65 family for embedded application using TSMC 65 nm low leak process. It is an SRAM-based FPGA with embedded non-volatile configuration memory, achieving an operational power divided by 7 and a standby power by about 1000. Achronix commercialized high-speed FPGAs, the “Speedster family,” based on the research of Cornell University, USA. The most important characteristic is the token passing mechanism with asynchronous circuits. A data token, which takes the role of data and clock in a common FPGA, is passed by handshaking. The first product SPD60 using TSMC 65 nm process achieved almost three times the throughput of a common FPGA. The maximum throughput was 1.5 GHz.

Tabula’s FPGA reduced the cost by dynamic reconfiguration using the same logic cells for multiple functions. ABAX series by Tabula generates a multiple frequency clock from the system clock, and uses it both for the internal logic and dynamic configuration. By time multiplexing a fixed programmable logic region, the effective logic area can be increased. Tabula introduced a new “time” dimension into two-dimensional chips and called their products three-dimensional FPGAs. Abound Logic announced “Rapter” with crossbar switches and a scalable architecture, but closed in 2010. Tier Logic developed a novel

3D-FPGA whose SRAM configuration is formed with amorphous silicon TFT technology on the CMOS circuits in collaboration with Toshiba; however, due to fund shortage, the project was terminated.

2010s

(1) Technology advances and new trends:

In 2010, Xilinx and Altera started the shipping of 28 nm generation FPGAs that can be considered to be more advantageous than ASIC chips. Both companies added a mid-range product line to their high-end and low-end lines. For example, Xilinx changed its fabrication from UMC to TSMC both in Taiwan, and all products of the Xilinx 7 series (High-end Virtex-7, mid-range Kintex-7, and low-end Artix-7) are fabricated with a 28 nm process for low power and high degree of functionality. At that time, both Xilinx and Altera used TSMC for their foundries. The followings are technology trends in 28 nm generation FPGA.

(a) The trend of new generation SoC:

Around 2000, both Xilinx and Altera shipped the first generation of SoC products with FPGA, but their lifetime was relatively short. On the other hand, FPGAs with soft-core processors have been widely used. The demands for embedded hardware cores grew, and by using advanced technologies, CPU cores with enough performance capable of fulfilling such demands could be embedded. This promoted FPGAs for SoC, providing a 32bit ARM processor and enhanced I/O. They are called SoC FPGA, programmable SoC, or SoPD (System on Programmable Device). For example, Xilinx introduced a new family Zynq-7000 which integrates an ARM Cortex-A9 MPCore and the 28 nm 7 series FPGA programmable logic. Altera's new product, "SoC FPGA," integrated dual-core ARM Cortex-A9 MPCore and FPGA fabric into a device. A representative example is the Cyclone V SoC.

(b) Partial reconfiguration:

Partial reconfiguration is a functionality which reconfigures a part of an FPGA, while others are still under operation. The functions can be updated without stopping the system. Xilinx started to support this function in their high-end FPGA devices from Virtex-4 with its EDA tool (after ISE12). Altera also started to support this feature from Stratix V. Since the major two vendors started to support partial reconfiguration in their tool, this technique is becoming widely spread.

(c) 3D-FPGA (2.5D-FPGA):

Xilinx shipped multi-chip products placing multiple FPGAs on a silicon interposer with stacked silicon interconnect. It is called the 2.5D implementation. Unlike the 3D implementation of multiple chips with TSVs, whose cost tends to be high, 2.5D can mount chips without TSVs. Virtex-7 2000T with TSMC 28 nm HPL process integrated 200 million logic cells corre-

sponding to the largest ASIC with 68 billion transistors and 20,000,000 gates.

(d) FPGAs for automobiles:

Xilinx extended the Artix-7 FPGA and shipped XA Artix-7 FPGA which fully satisfies the AEC-Q100 standard for automobile. XA Artix-7 complements the programmable SoC XA Zynq-7000. Furthermore, the authentication of third-party tools is undergoing to satisfy the ISO-26262 standard. Altera and Lattice also tackle automobile solutions.

(e) C language design environment:

Recently, C language design environments have become popular in FPGA design. Xilinx Vivado HLS can translate the hardware description in C, C++, and System C to devices directly without RTL description. It can be used both from ISE and Vivado. On the other hand, Altera aggressively introduces the OpenCL environment. It is a C-base programming language running on various platforms: CPU, GPU, DSP, and FPGA and allows Altera's FPGAs to be used as hardware accelerators.

(f) Others:

In order to expand the I/O bandwidth of FPGAs with optical interfaces, optical FPGAs have been introduced. Radiation-hardened FPGAs are also being developed.

(2) The road map of process technology for FPGA:

After the 28 nm generation, Xilinx presented the 20 nm FPGA Kintex UltraScale, and the Virtex UltraScale provided a new architecture. The largest series Virtex UltraScale is corresponding to an ASIC with 50,000,000 gates. All of UltraScale devices use TSMC 20 nm process, but high-end Virtex UltraScale use the TSMC 16 nm FinFET. On the other hand, Altera shipped the Arria 10 for next-generation FPGAs, the "Generation 10" devices, and announced Stratix 10 FPGAs. They are all SoCs with embedded processors. Generation 10 devices are fabricated by Intel's 14 nm generation FinFET and TSMC 20 nm technologies. The high-end Stratix 10 can work at 1 GHz clock.

Logic IC process advances to the next generation every 2 years. The Intel processor is a representative example of such evolution; however, since the 2000s, FPGAs mostly caught up with that pace. On the other hand, ASICs followed the advances until the early 2000s and stalled for about 10 years at 130-90 nm, except for some special applications such as game machines. As shown in Fig. 1.12, FPGAs have been fabricated along with the technology road map. The pace is more than that of general-purpose processors. FPGA will use 28 nm, 20 nm, and 16/14 nm processes and will get a similar competitive performance to ASIC with 130 nm, 90 nm, or 65 nm, two or three generations behind.

(3) Oligopoly and industry restructuring:

In 2010, oligopoly continued in the FPGA industry. Major FPGA vendors, Xilinx and Altera, occupy more than 80% of the shares, and other parts are shared between Lattice and Actel. Actel, at the fourth place in the industry, was acquired

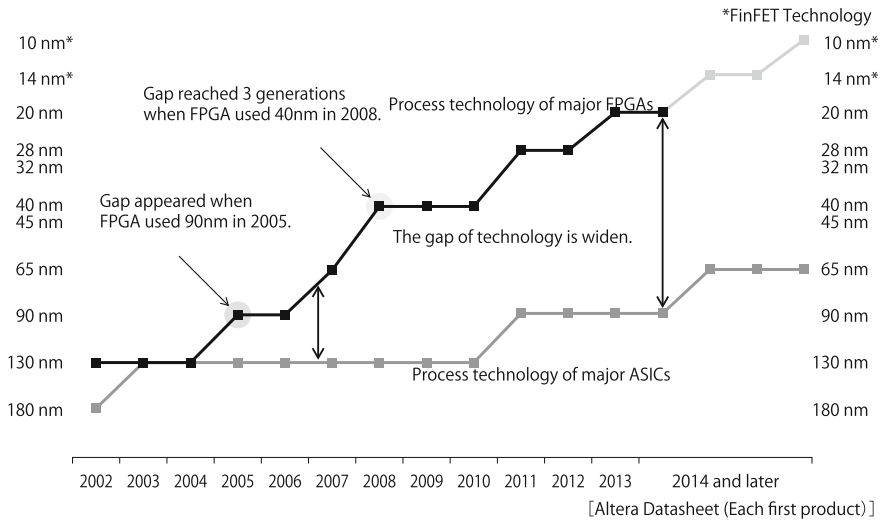


Fig. 1.12 Process road map of FPGA and ASIC

by Microsemi in 2010, and ships flash and anti-fuse non-volatile FPGAs as Microsemi FPGA.

Among the FPGA vendors established in the 1980s, Quicklogic focused on anti-fuse FPGAs, but it changed its strategy and has produced customer specific standard products (CSSP) for specific custom fields. CSSP is not an all-programmable product, but only a part of the chip is programmable. On the other parts, a lot of standard interface circuits are mounted to cope with customers' needs. Also, Atmel's FPGA technology is mostly combined with their AVR controllers, and they withdrew from FPGA industry. Among the new FPGA vendors established in the 2000s, SiliconBlue was acquired by Lattice, and Lattice introduced a new line of the iCE40 family with a 40 nm process. Tabula which proposed a low-cost dynamic reconfiguration finished its projects in March 2015. On the other hand, Achronix produced the Speedster22i FPGA family with Intel's 22 nm tri-gate process technology in 2015.

In the spring of 2016, the semiconductor industry entered a great restructuring era, and large-scale M&As have been carried out. The FPGA industry was naturally involved. Intel acquired the major FPGA vendor Altera in June 2015. The total operation reached 167 billion dollars. It was more than the amount of yearly sales of Altera, the largest scale in the FPGA history. Intel aims to occupy the market of data center and IoT by the integration of processors and FPGAs. For this purpose, Intel selected Altera's FPGA as an essential technology.

On the other hand, Qualcomm and Xilinx announced a strategic cooperation contract. Both companies support solutions for data center with ARM processors for servers and FPGA technologies. They focus on the basic technology

of cloud computing including big data analysis and data storage. Furthermore, Xilinx announced a multi-year strategic cooperation contract with IBM. By combining Xilinx FPGAs with IBM Power Systems and using the combination as an accelerator for specific applications, a highly energy efficient data center can be produced. Such systems are suitable for machine learning, network virtualization, high-performance computing, and big data analysis. They try to compete against the “Catapult” of Microsoft (in collaboration with Altera and Intel) with such strategic cooperation [5].

References

1. V. Betz, J. Rose, A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*, (Kluwer Academic Publishers 1999)
2. Z. Kohavi, *Switching and Finite Automata Theory*, 2nd edn. McGraw-Hill (1978)
3. S.D. Brown, R.J. Francis, J. Rose, Z.G. Vranesic, *Field-Programmable Gate Array*, (Kluwer Academic Publishers 1992)
4. S.M. Trimberger, *Field-Programmable Gate Array Technology*, (Kluwer Academic Publishers 1994)
5. A. Putnam et al., A reconfigurable fabric for accelerating large-scale datacenter services, in *ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, (2014), pp. 13–24