CHAPTER

FIELD-PROGRAMMABLE LOGIC 2

What makes field-programmable logic (FPL) attractive in many applications are their low up-front costs and short turnaround times. That it should be possible to turn a finished piece of silicon into an application-specific circuit by purely electrical means — i.e. with no bespoke photomasks or wafer processing steps — may seem quite surprising at first, and section 2.1 demonstrates the basic approach. Sections from 2.2 onwards then give technical details and explain the major differences that set apart the various product families from each other, before the particularities of the FPL design flow get discussed in section 2.6.

2.1 GENERAL IDEA

The term "programmable" in field-programmable logic is a misnomer as there is no program, no instruction sequence to execute. Instead, pre-manufactured subcircuits get configured into the target circuit via electrically programmable links that can be done — and in many cases also undone — as dictated by so called **configuration bits**. This is nicely illustrated in figs.2.1 to 2.3 from [8] (copyright Wiley-VCH Verlag GmbH & Co. KG, reprinted with permission).

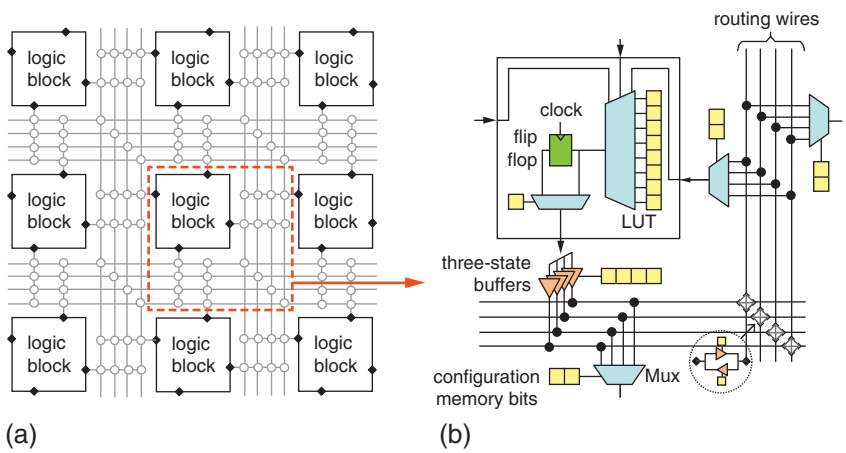
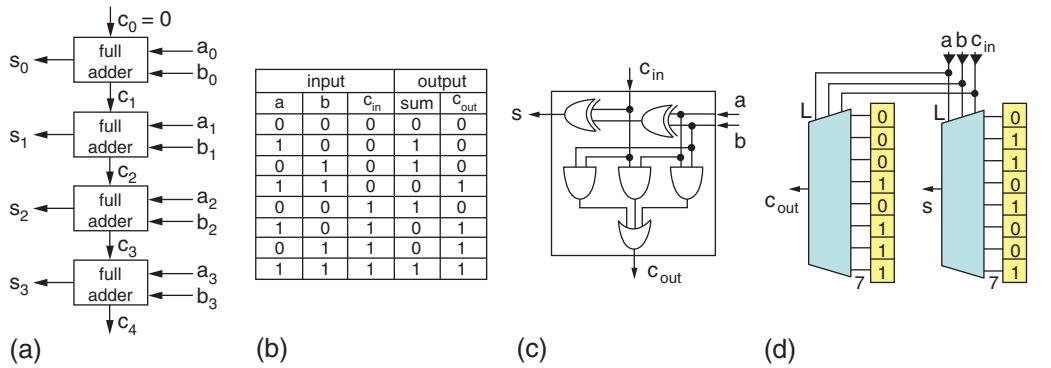


FIGURE 2.1

FPGA hardware resources before configuration. Top-level organization (a) and one configurable logic cell (b). In panel (b), each small square represents one configuration bit.



input			output	
a	b	c_{in}	sum	c_{out}
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

FIGURE 2.2

Target functionality ripple carry adder. General diagram (a), truth table of one full adder slice (b), gate-level circuit (c), and implementation as a lookup table (LUT) (d).

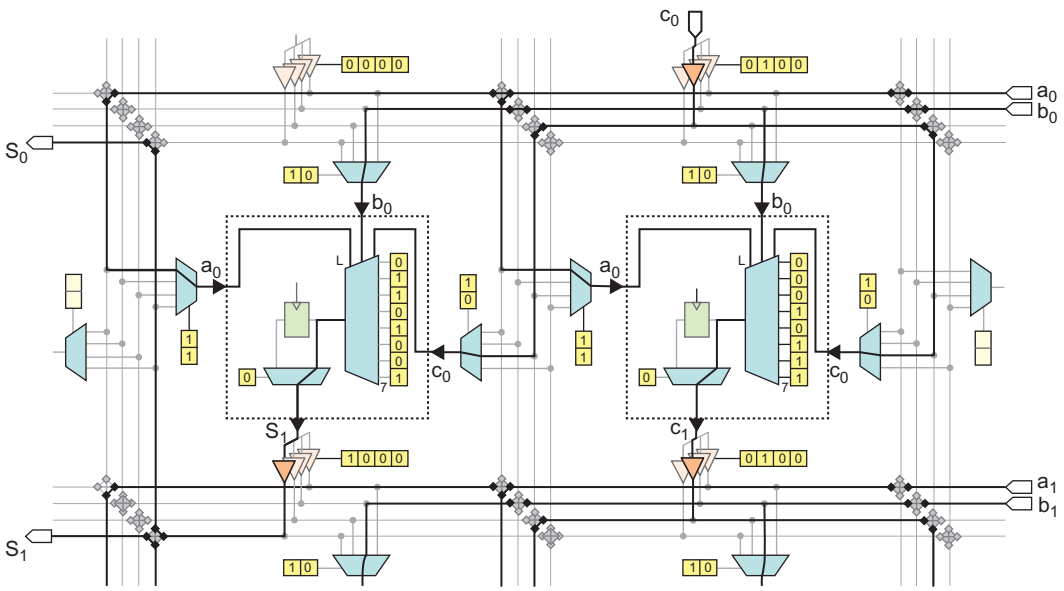


FIGURE 2.3

FPGA after configuration. One full adder slice implemented using the pre-fabricated resources. Highlighted lines show the wires activated by configuration bits.

Key properties of any FPL device depend on decisions taken by its developers along two dimensions. A first choice refers to how the device is actually being configured and how its configuration is stored electrically, while a second choice is concerned with the overall organization of the hardware resources made available to customers.[1]

[1] Customers, in this case, are design engineers who want to create their circuits using an FPL device (rather than as a semi or full-custom IC).

2.2 CONFIGURATION TECHNOLOGIES

Three configuration technologies coexist today, they all have their roots in memory technology.

2.2.1 STATIC MEMORY

The key element here is an electronic switch — such as a transmission gate, a pass transistor, or a three-state buffer — that gets turned "on" or "off" under control of a configuration bit. Unlimited reprogrammability is obtained from storing the configuration data in static memory (SRAM) cells or in similar on-chip subcircuits built from two cross-coupled inverters, see fig.2.4a.

Reconfigurability is very helpful for debugging. It permits one to probe inner nodes, to alternate between normal operation and various diagnostic modes, and to patch a design once a flaw has been located. Many RAM-based FPL devices further allow for reconfiguring their inner logic during operation, a capability known as **in-system configuration** (ISC) that opens a door towards reconfigurable computing.

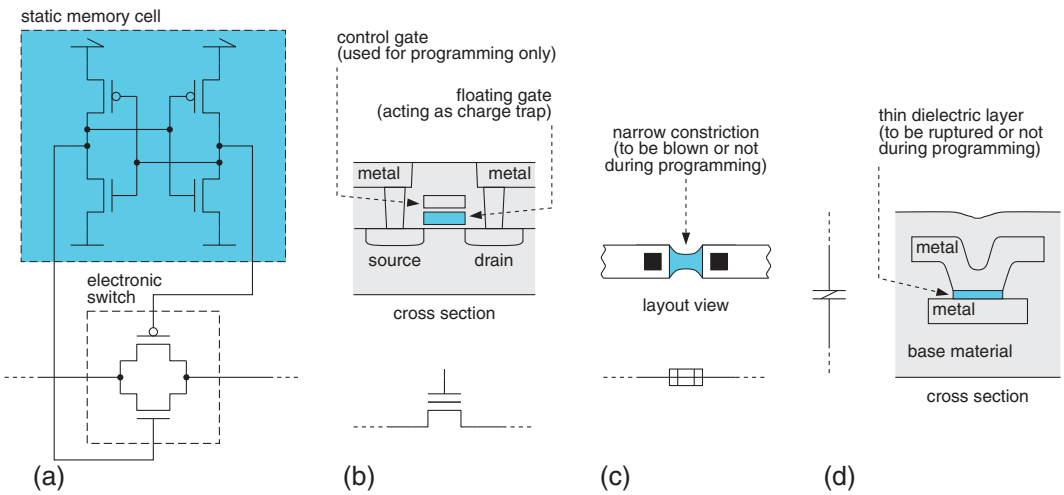


FIGURE 2.4

FPL configuration technologies (simplified, programming circuitry not shown). Switch steered by static memory cell (a), MOSFET controlled by a charge trapped on a floating gate (b), fuse (c), and antifuse (d).

As a major drawback of SRAM-based storage, an FPL device must (re-)obtain the entire configuration — the settings of all its programmable links — from outside whenever it is being powered up. The problem is solved in one of three possible ways, namely

(a) by reading from a dedicated bit-serial or bit-parallel off-chip ROM,
(b) by downloading a bit stream from a host computer, or
(c) by long-term battery backup.

2.2.2 FLASH MEMORY

Flash memories rely on special MOSFETs where a second gate electrode is sandwiched between the transistor's bulk material underneath and a control gate above, see fig.2.4b. The name **floating gate** captures the fact that this gate is entirely surrounded by dielectric material. An electrical charge trapped there determines whether the MOSFET, and hence the programmable link too, is "on"or "off".[2]

Charging occurs by way of hot electron injection from the channel. That is, a strong lateral field applied between source and drain accelerates electrons to the point where they get injected through the thin dielectric layer into the floating gate, see fig.2.5a. The necessary programming voltage on the order of 5 to 20 V is typically generated internally by an on-chip charge pump.

Erasure occurs by allowing the electrons trapped on the floating gate to tunnel through the oxide layer underneath the floating gate. The secret is a quantum-mechanical effect known as Fowler-Nordheim tunneling that comes into play when a strong vertical field (8 ... 10 MV/cm or so) is applied across the gate oxide.

Flash FPL devices are non-volatile and immediately live at power up, thereby doing away with the need for any kind of configuration-backup apparatus. The fact that erasure must occur in chunks, that is to say many bits at a time, is perfectly adequate in the context of FPL. Data retention times vary between 10 and 40 years. Endurance of flash FPL is typically specified with 100 to 1000 configure-erase cycles, which is much less than for flash memory chips.

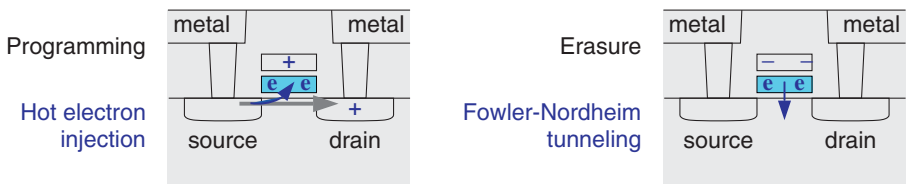


FIGURE 2.5

Flash memory operation during programming (a) and erasure (b).

2.2.3 ANTIFUSES

Fuses, which were used in earlier bipolar PROMs and SPLDs, are narrow bridges of conducting material that blow in a controlled fashion when a programming current is forced through. Antifuses, such as those employed in today's FPGAs, are thin dielectrics separating two conducting layers that are made to rupture upon applying a programming voltage, thereby establishing a conductive path of low impedance.

[2] More precisely, the presence or absence of an electrical charge modifies the MOSFET's threshold voltage and so determines whether the transistor will conduct or not when a voltage gets applied to its control gate during memory readout operations.

In either case, programming is permanent. Whether this is desirable or not depends on the application. Full factory testing prior to programming of one-time programmable links is impossible for obvious reasons. Special circuitry is incorporated to test the logic devices and routing tracks at the manufacturer before the unprogrammed devices are being shipped. On the other hand, antifuses are only about the size of a contact or via and, therefore, allow for higher densities than reprogrammable links, see fig.2.4c and d. Antifuse-based FPL is also less sensitive to radiation effects, offers superior protection against unauthorized cloning, and does not need to be configured following power-up.

Table 2.1 FPL configuration technologies compared.

Configuration technology	Non volatile	Live at power up	Reconfigurable	Unlimit. endurance	Radiation tolerance of config.	Area occupation per link	Extra fabr. steps
SRAM	no	no	in circuit	yes	poor	large	0
Flash memory	yes	yes	in circuit	no	good	small	>5
Antifuse PROM	yes	yes	no	n.a.	best	smallest	3

2.3 ORGANIZATION OF HARDWARE RESOURCES

2.3.1 SIMPLE PROGRAMMABLE LOGIC DEVICES (SPLD)

Historically, FPL has evolved from purely combinational devices with just one or two programmable levels of logic such as ROMs, PALs and PLAs. Flip-flops and local feedback paths were added later to allow for the construction of finite state machines, see fig.2.6a and b. Products of this kind continue to be commercially available for glue logic applications. Classic SPLD examples include the 18P8 (combinational) and the 22V10 (sequential).

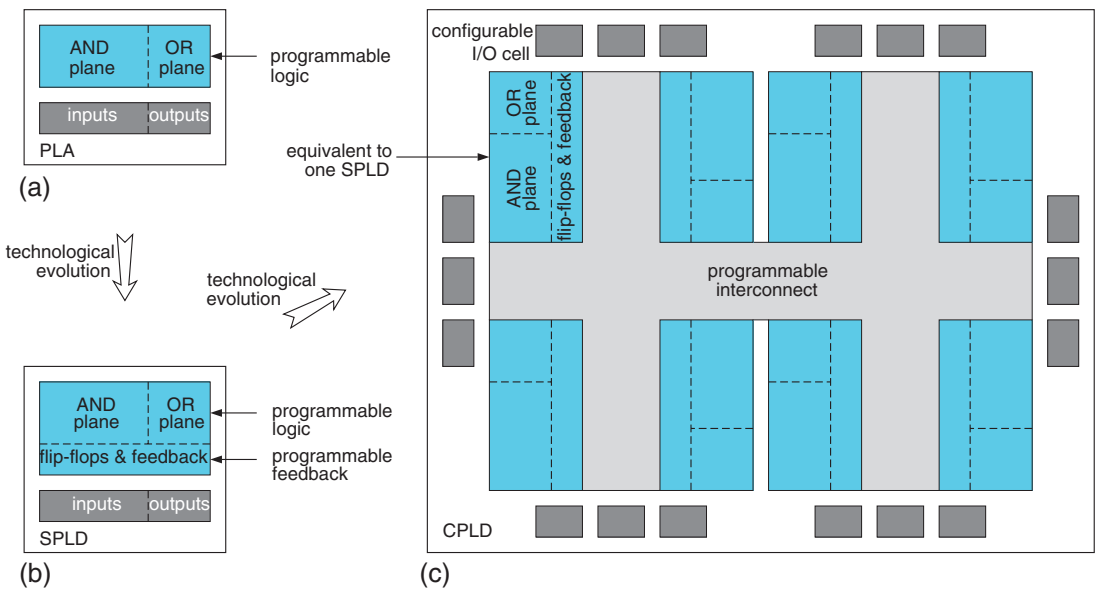


FIGURE 2.6

General architecture of CPLDs (c) along with precursors (a,b).

The rigid two-level-logic-plus-register architecture and the scanty resources (number of inputs, outputs, product terms, flip-flops) naturally restrict SPLDs to small applications. More powerful architectures had thus to be sought, and the spectacular progress of VLSI technology has made their implementation economically feasible from the late 1980's onwards.

2.3.2 COMPLEX PROGRAMMABLE LOGIC DEVICES (CPLD)

CPLDs simply followed the motto “more of the same”, see fig.2.6c. Many identical subcircuits, each of which conforms to a classic SPLD, are combined on a single chip together with a large programmable interconnect matrix or network. A difficulty with this type of organization is that a partitioning into a bunch of cooperating SPLDs has to be imposed artificially on any given application which neither benefits hardware nor design efficiency.

2.3.3 FIELD-PROGRAMMABLE GATE ARRAYS (FPGA)

FPGAs have their overall organization patterned after that of gate-arrays. Many **configurable logic cells** are arranged in a two-dimensional array with bundles of parallel wires in between. A switchbox is present wherever two wiring channels intersect, see fig.2.7.[3] Depending on the product, each logic cell can be configured such as to carry out some not-too-complex combinational operation, to store a bit or two, or both. As opposed to traditional gate arrays, it is the state of programmable links rather than fabrication masks that decide on logic functions and signal routing. The number of configurable logic cells greatly varies between products, with typical figures ranging between a few dozens and hundred thousands.

Owing to their more scalable and flexible organization, FPGAs prevail over CPLDs. They are differentiated further depending on the granularity of the configurable logic.

[3] While it is correct to think of alternating cells and wiring channels from a conceptual point of view, you will hardly be able to discern them under a microscope as they are superimposed for the sake of layout density.

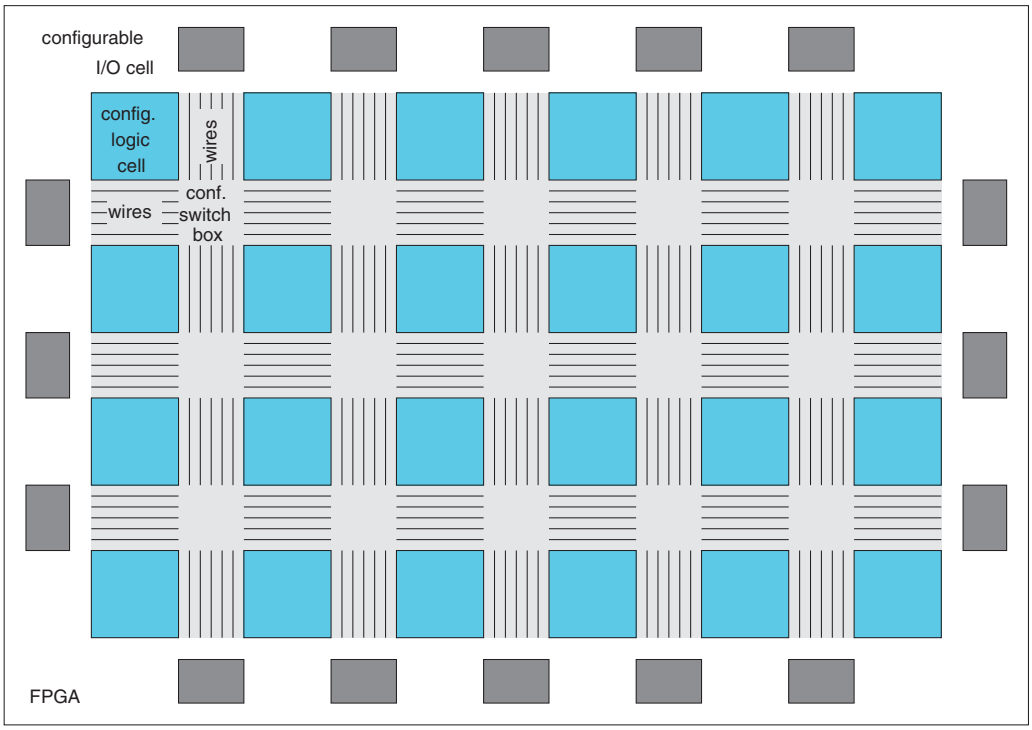


FIGURE 2.7

General architecture of FPGAs.

Fine-grained FPGAs. One speaks of a fine-grained architecture when the configurable cells are so simple that they are capable of implementing no more than a few logic gates and/or one bistable. In the example of fig.2.8a, for instance, each logic cell can be configured into a latch, or a flip-flop, or into almost any 3-input gate.

Coarse-grained FPGAs. Here cells are designed to implement combinational functions of four or more variables and are capable of storing two or more bits at a time. The logic cell of fig.2.8b has 16 inputs and 11 outputs, includes two programmable 4-input lookup tables (LUT), two generic bistables that can be configured either into a latch or a flip-flop, a bunch of configurable multiplexers, a fast carry chain, plus other gates. Of course, the superior functional capabilities offered by a coarse-grained cell are accompanied by a larger area occupation.

The gate-level netlists produced by general synthesis tools map more naturally onto fine-grained architectures because fine-grained FPGAs and semi-custom ICs provide similar primitives. This makes it possible to move back and forth between field- and mask-programmed circuits with little overhead and to defer final commitment until fairly late in the design cycle. Conversely, fine-grained FPGAs tend to be more wasteful in terms of configuration bits, routing resources, and propagation delays.

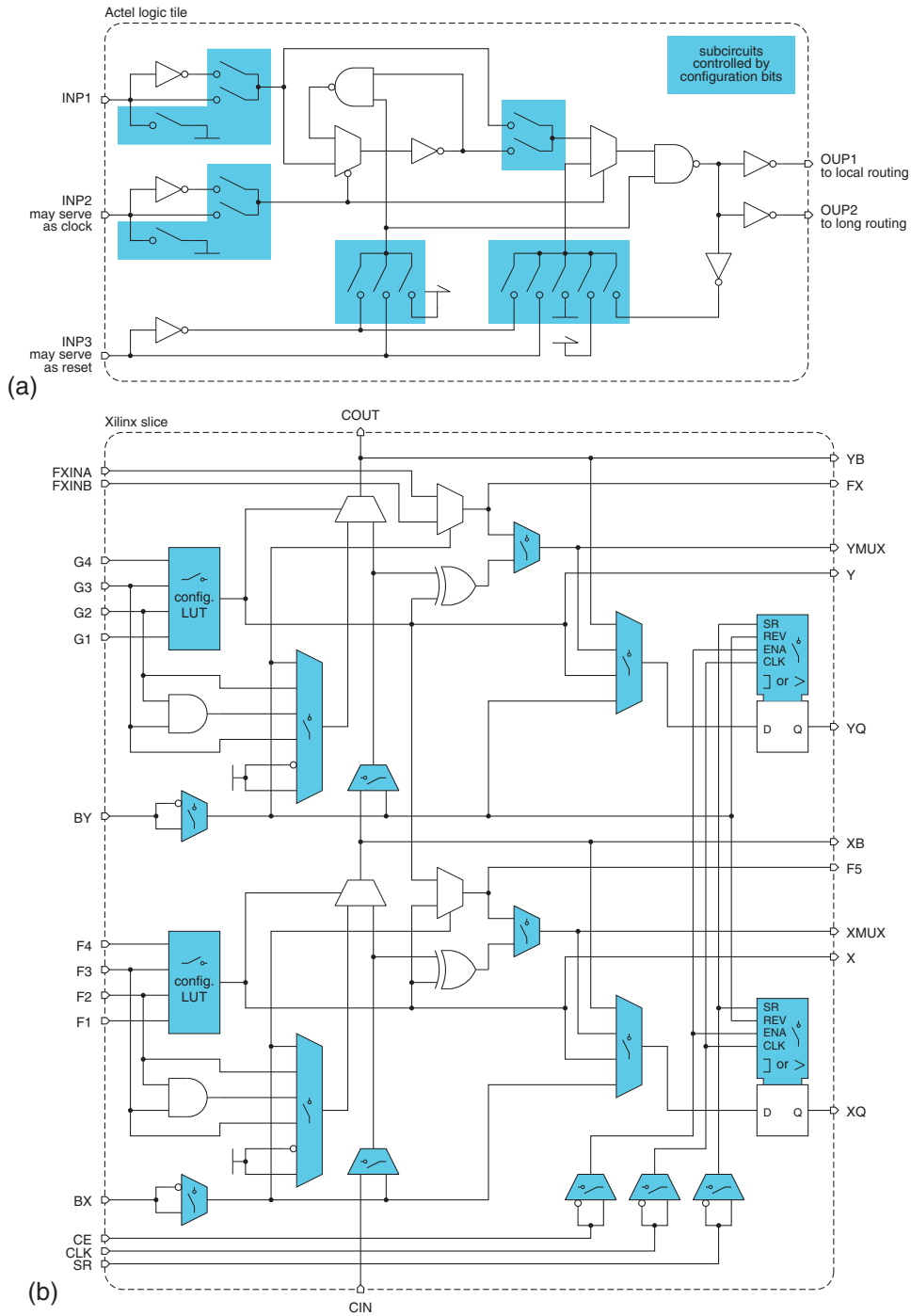


FIGURE 2.8

Fine-grained vs. coarse-grained FPGAs. A small (Actel ProASIC) (a) and a large logic cell (Xilinx Virtex-4, simplified) (b).

Vendors of coarse-grained FPGAs have done a fair bit to overcome the drawbacks of their idiosyncratic FPGA architectures by providing their customers with proprietary software tools that help them make good usage of the available hardware resources.[4] Another reason that contributed to the popularity of coarse-grained FPGAs is that on-chip RAMs come at little extra cost when combined with configuration from static memory. In fact, a reprogrammable LUT is nothing else than a tiny storage array. It is thus possible to string together multiple logic cells such as to make them act collectively like a larger RAM. In the occurrence of fig.2.8b, each of the two larger LUTs in each logic tile contributes another 16 bit of storage capacity.

Fig.2.9 indicates that the optimum trade-off for LUTs has shifted from 4 to 6 inputs over the last couple of generations. A comparison of figs.2.8b (2004, 90 nm) and 2.10 (2009, 40 nm) indeed confirms the trend towards coarser granularities. An excerpt from the datasheet of a competing product (2010, 20 nm) is shown in fig.2.11.[5]

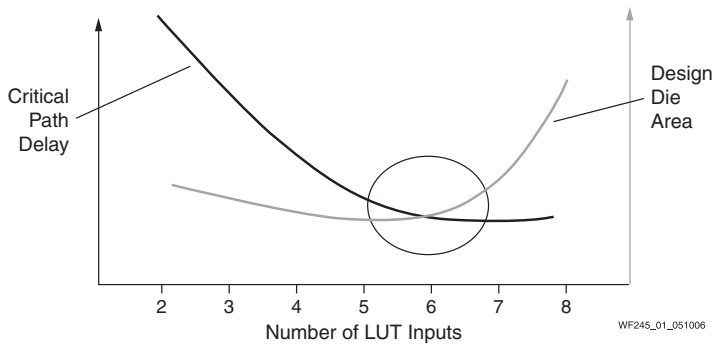


FIGURE 2.9

LUT granularity trade-offs at the 65 nm technology node (source: Xilinx, reprinted with permission).

[4] Such as Synplify Pro (Synopsys), Quartus (Altera), Vivado (Xilinx).

[5] FPL vendors refer to configurable logic cells by proprietary names. "VersaTile" is Actel's term for their fine-grained cells while Altera uses the name "adaptive logic module" (ALM) for their coarse-grained building blocks. Xilinx refers to their counterparts as "configurable logic blocks" (CLB). Depending on the product family, one a CLB may be composed of two "slices" each of which includes several LUTs and bistables. Cypress speaks of "universal digital blocks" (UDB).

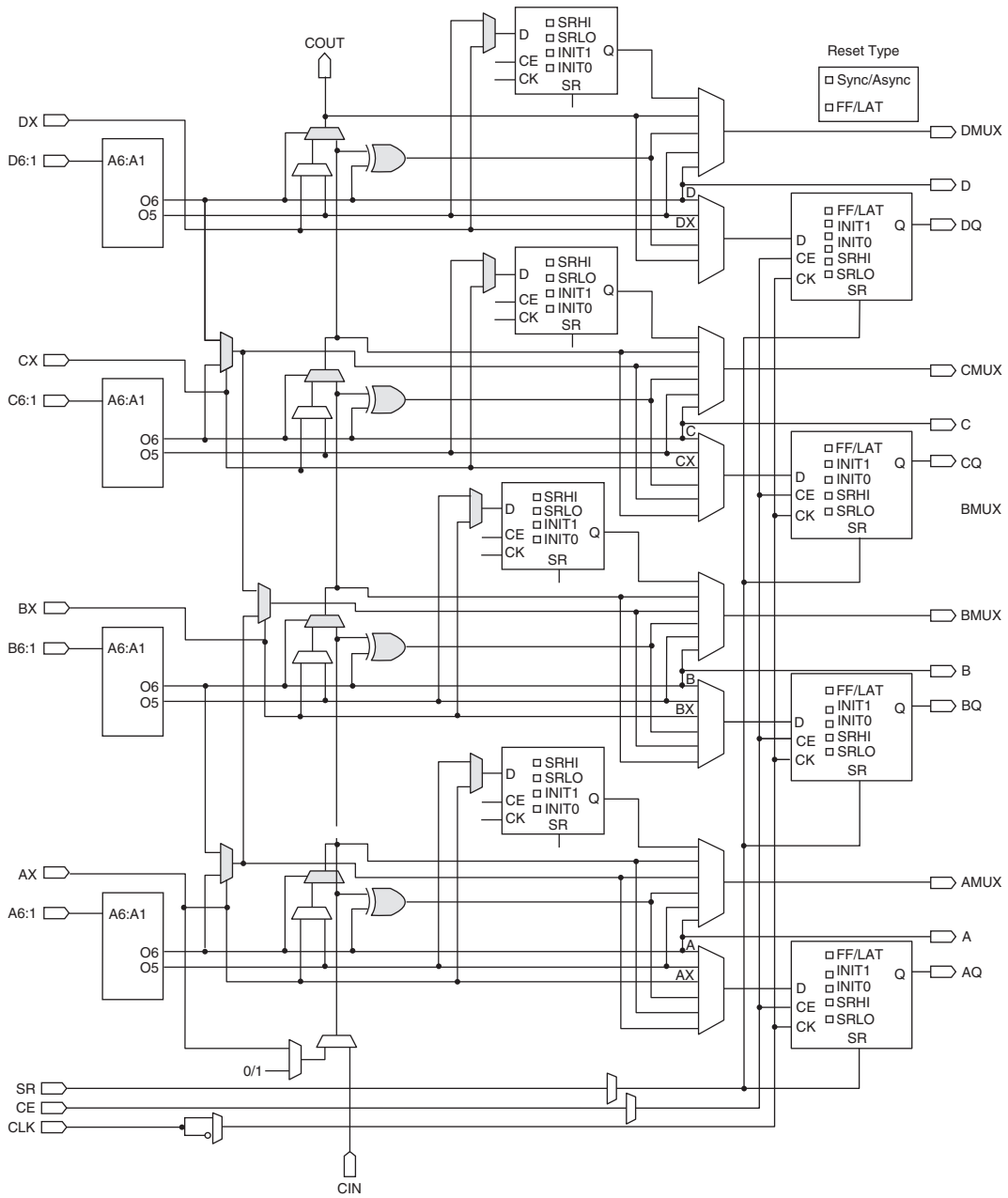


FIGURE 2.10

Logic slice from Xilinx Virtex-6, SLICEL with 4 6-input LUTs and 8 bistables (source: Xilinx, reprinted with permission).

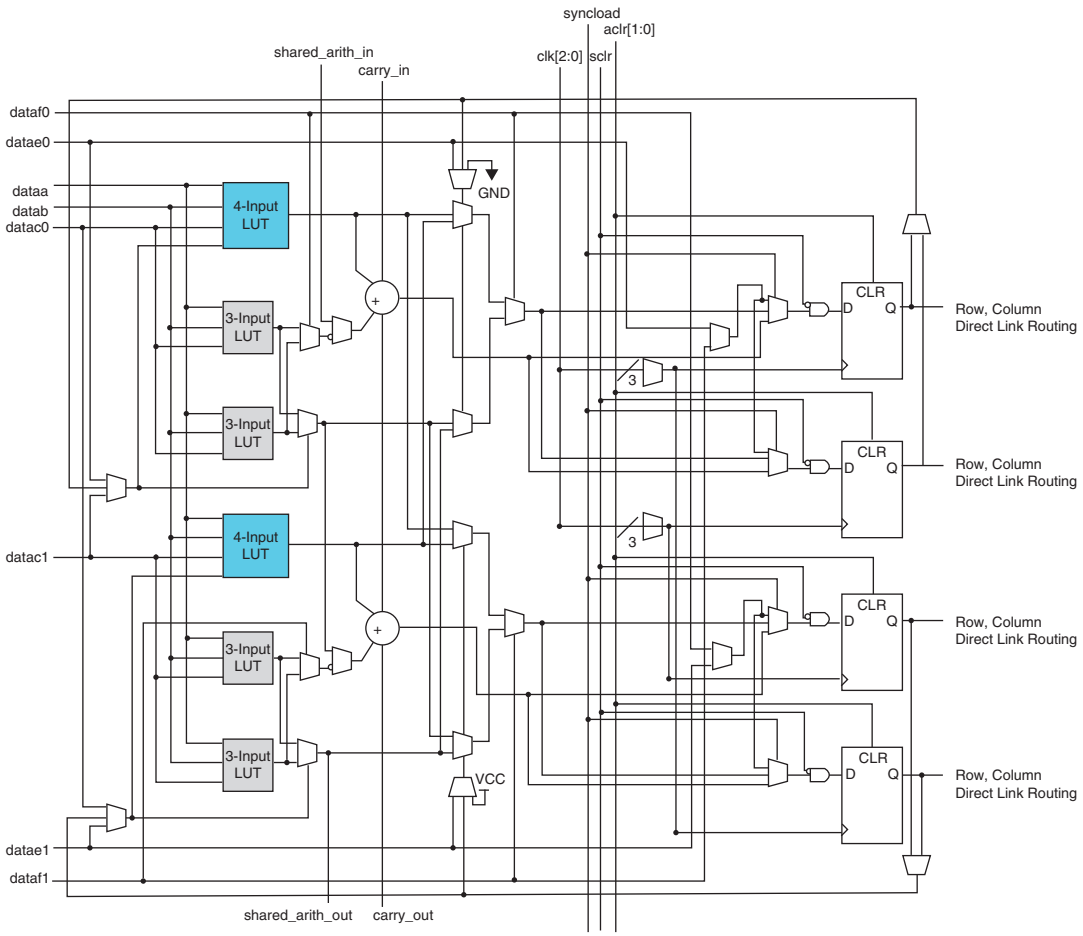


FIGURE 2.11

Adaptive logic module from Altera Stratix V (source Altera, reprinted with permission).

2.4 COMMERCIAL ASPECTS

2.4.1 AN OVERVIEW ON FPL DEVICE FAMILIES

Table 2.2 classifies major commercial CPLD and FPGA device families along the two dimensions configuration technology and hardware organization.

Table 2.2 Field-programmable logic product families.

Configuration technology	Overall organization of hardware resources		
	CPLD	FPGA	
		coarse grained	fine grained
Static memory (SRAM)		Xilinx Virtex, Kintex, Artix, Spartan. Lattice SC, EC, ECP. Altera Stratix, Arria, Cyclone. eASIC Nextreme SL[a]. Achronix Speedster[b].	Atmel AT6000, AT40K.
Flash memory	Xilinx XC9500, CoolRunner-II. Altera MAX. Lattice MACH 1,...,5. Cypress Delta39K, Ultra37000, PSoC 1,...,5LP[f].	Lattice XP[c], MACH XO.	Actel[d] ProASIC3, ProASIC3 nano, Igloo, Fusion[e].
Antifuse (PROM)		QuickLogic Eclipse II, PolarPro.	Actel MX, Axcelerator AX.

[a] Combines RAM-configurable LUTs with e-beam single via-layer customization for interconnect.
[b] Combines synchronous I/O with self-timed clocking inside.
[c] Combines on-chip flash memory with an SRAM-type configuration memory.
[d] Actel has been acquired by Microsemi in late 2010.
[e] Mixed-signal FPGAs.
[f] Mixed-signal CPLDs.

2.4.2 THE PRICE AND THE BENEFITS OF ELECTRICAL CONFIGURABILITY

For obvious reasons, the ability to (re)define a part's functionality long after it has left the fabrication line is extremely valuable in the marketplace. What's more, many applications that mandated a custom ASIC just a few years ago fit into a single FPL device today. Fueled by technological progress and continued price reductions, this trend is bound to carry on.

Yet, FPL is unlikely to rival hardwired logic on the grounds of integration density, unit costs, and energy efficiency unless there is an unforeseen technological breakthrough. This is because FPL must

accommodate extra transistors, programmable links, interconnect lines, vias, lithographic masks, and wafer processing steps to provide for configurability. Also, the required and the prefabricated hardware resources never quite match, leaving part of the manufactured gates unused.

In fact, comparisons of FPGAs against cell-based ASICs manufactured with a similar technology have exposed important overheads in terms of area, propagation delays, and power dissipation.

Overhead factors for				
area	timing	power	compared to	source
35	3.4...4.6	14	SRAM-based FPGAs	[9] (2007, 90 nm CMOS)
27	5.1	n.a.	idem	[10] (2013, 130 nm CMOS)

Opting for a reconfigurable FPGA, rather than for a mask-programmed ASIC, is thus likely to inflate the *AT*-product by more than two orders of magnitude. While antifuse technology, hardwired multipliers, etc. improve the situation, a significant penalty remains. The huge area overhead further explains why large FPGAs continue to be rather expensive, even when bought in substantial quantities. FPL vendors attempt to compensate for this by using the most advanced semiconductor fabrication processes available and have indeed gotten ahead of the average ASIC technology in recent years [11]. Table 2.3 shows what is possible today (2014Q1).

Fig.2.12 puts FPL devices in perspective with semi- and full-custom ICs. We would like to emphasize that this is a simplified overview and that numbers are quoted for illustrative purpose only. It should nevertheless become clear that each technique has its particular niche where it is the best compromise.

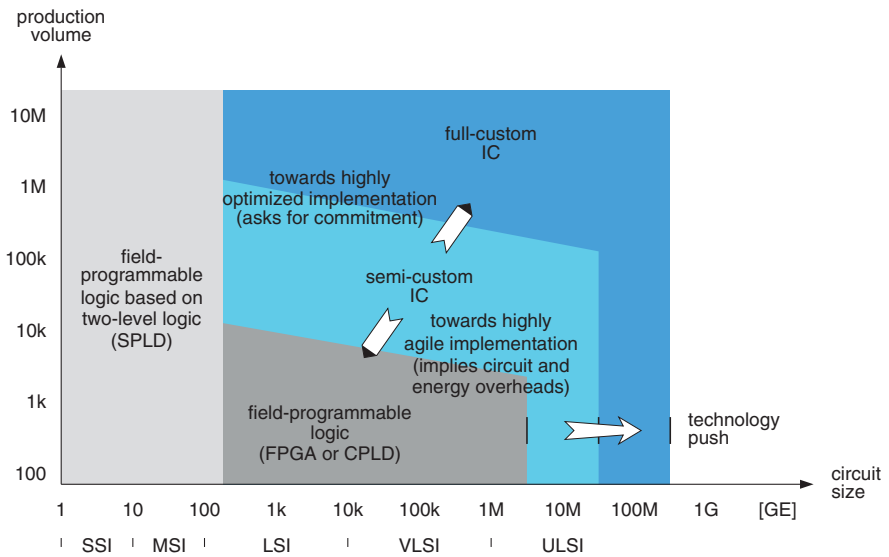


FIGURE 2.12

Application scopes of various implementation techniques as a function of circuit complexity and volume (simplified).

2.5 EXTENSIONS OF THE BASIC IDEA

From the above, it should be clear that configurable logic is best confined to those circuit portions that are subject to frequent changes when unit costs, energy efficiency, or operating speed are critical. Vendors have thus extended their FPL families beyond table 2.2 by combining them with less malleable but more cost-effective and more efficient hardware resources. The idea is to provide just as much configurability as needed to better compete with business rivals such as mask-programmed ICs, signal processors, and with competing FPL products.

Datapath units. Configurable logic cells are designed to implement small look-up tables and random logic functions. When used to implement multiplications on wide data words, for instance, the extensive configurability tends to become overly burdensome. Some FPGA families have, therefore, been equipped with configurable datapath units optimized for multiply-accumulate (MAC) and related arithmetic-logic operations, see fig.2.13. Compared to configurable logic cells, those specialized units come with fairly generous word widths and support digital signal processing applications much more efficiently in terms of hardware usage, throughput and energy provided the synthesis software consistently maps inner products and other suitable operations onto those coarse-grain computing resources.

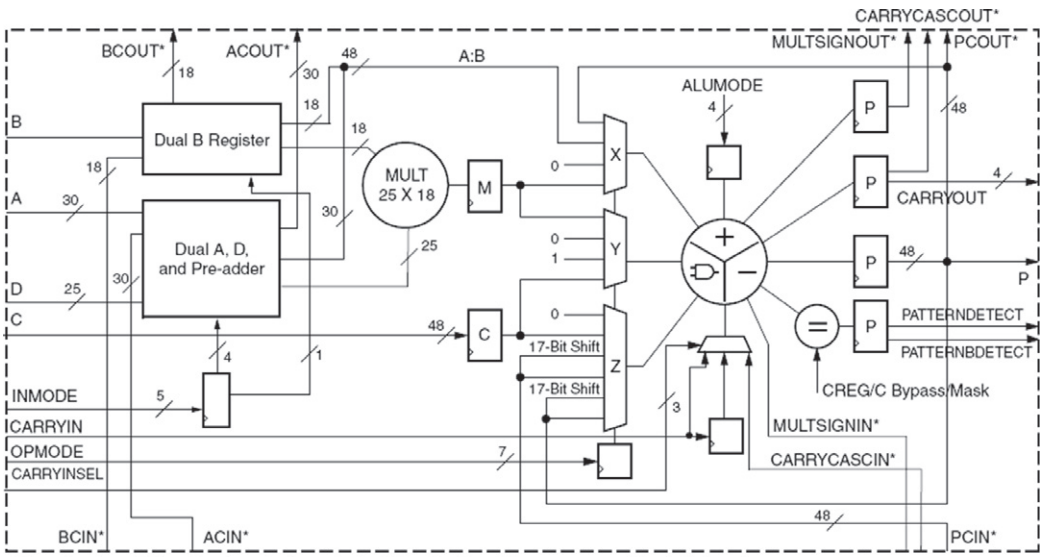


FIGURE 2.13

DSP48E1 slice from Xilinx Virtex-6 (source: Xilinx, reprinted with permission).

Hardwired building blocks. Almost all FPL devices feature hardwired subcircuits on the same die. This is because it makes no sense to tie up precious configurable resources for implementing fixed functions. Typical fixed-function blocks include

- SRAMs, FIFOs, clock recovery circuits, SerDes.
- Industry-standard functions and interfaces (such as PCI, USB, FireWire, Ethernet, WLAN, JTAG, LVDS, etc.).
- Analog-to-digital and digital-to-analog converters.
- Entire microprocessor and DSP cores (e.g. PowerPC, ARM).
- Weakly configurable analog subfunctions such as filters or phase locked loops (PLL).

Table 2.3 Maximum resources in two of the most advanced SRAM-based FPGA families. Competing products offer comparable features.

Vendor	Xilinx	
Product	Virtex-7	Virtex Ultrascale
Year introduced	2013	2014
Technology	20 nm CMOS planar	16 nm CMOS finFET
Configurable logic cells[a] [k]	1995	4407
Block RAM [Mbit]	68	115
DSP48 slices	3600	2880
I/O pins	1200	1456
Serial transceivers	96	104
PCI Express blocks	4	6
100G Ethernet blocks	0	7

[a] One Xilinx logic cell roughly corresponds to a 4-input LUT plus a flip-flop.

To give real-world numbers, table 2.3 reproduces selected data of high-end FPGAs from [12]. An extension that goes into a somewhat different direction are the

Field-programmable analog arrays (FPAA)**.** Electrically configurable analog circuits built from OpAmps, capacitors, resistors, and switchcap elements, have begun to appear on the market in the late 1990s. The next logical step was the extension to mixed-signal applications. Advanced products that combine configurable analog building blocks with a micro- or digital signal processor and with analog-to-digital and digital-to-analog converters come quite close to the vision of field-programmable systems on a chip. Vendors of field-programmable analog and mixed-signal arrays include Anadigm, Actel, Cypress, Lattice, and Silego.

Advanced configurable devices that include the right mix of hardwired blocks improve overall energy efficiency, help customers reduce time to market even further than pure FPL parts, and cut their product development and unit costs. Fig. 2.14 shows an example for mixed-signal applications. The trend towards (re)configuring larger, more powerful entities (ALUs, datapath units, memories, etc. rather than gates and LUTs) and towards combining (re)configurable logic with processor cores and fixed

function blocks is expected to continue. This will naturally lead to platform ICs, a concept that carries electrical configurability further and that is to be discussed in section 3.2.9.

For all enthusiasm about those phantastic capabilities and prospects, note that evaluating FPL devices may be frustrating. As opposed to full-custom ICs, manufactured gates, usable gates, and actual gates are not the same. **Manufactured gates** indicate the total number of GEs that are physically present on a silicon die. A substantial fraction thereof is not usable in practice because the combinational functions in a given design do not fit into the available lookup tables exactly, because an FPL device only rarely includes combinational and storage resources with the desired proportions, and because of limited interconnect resources. The percentage of **usable gates** thus depends on the application. The **actual gate** count, finally, tells how many GEs are indeed put to service by a given design. The three figures frequently get muddled up, all too often in a deliberate attempt to make one product look better than its competitors in advertisements, product charts, and datasheets.

Example

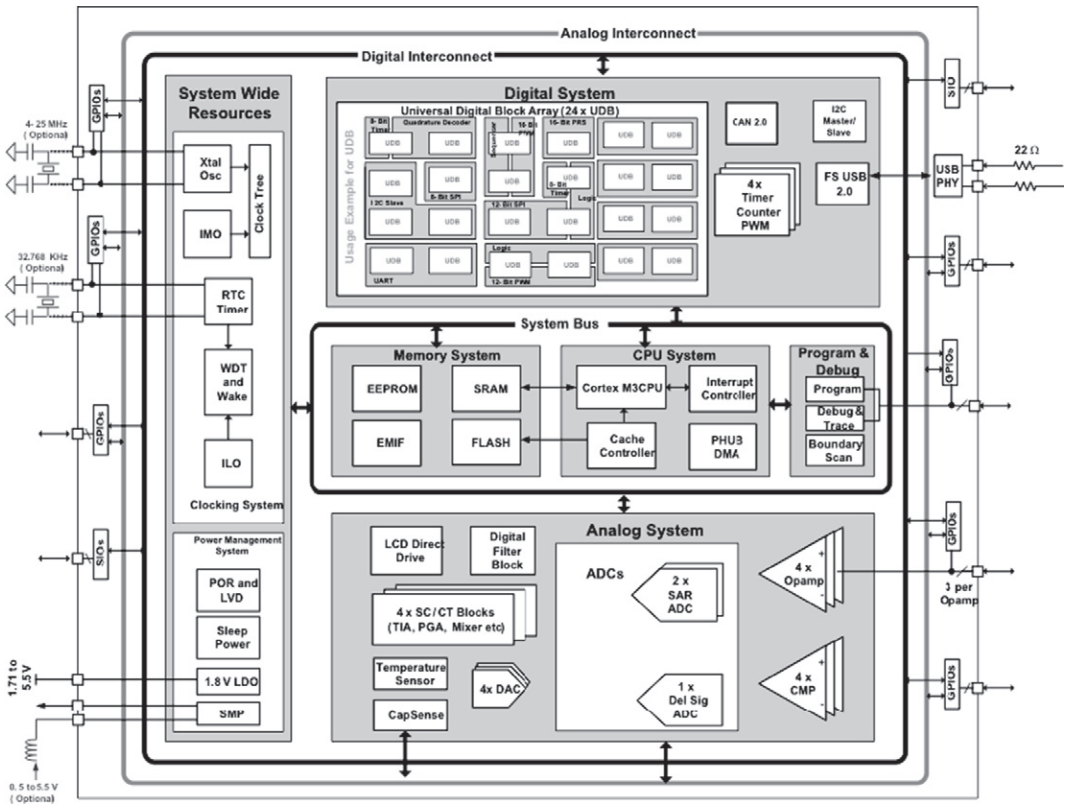


FIGURE 2.14

Block diagram of Cypress mixed-signal PSoC 5LP device (source Cypress, reprinted with permission).

□

That the available resources often get specified using proprietary units other than gate equivalents adds to the confusion. And as exposed in the quote below (after Kevin Morris), this is just the tip of the iceberg:

> Certainly one of the problems with FPGA technology is that you're constantly comparing different things. Apples and oranges, [Xilinx] configurable logic cells and [Altera] adaptive logic modules, field-programmable elements and [largely] hardwired [datapath] units, total negative slack and fastest clock, dynamic power at 20 °C and quiescent power at 85 °C, prices today for quantity 1000 and prices for 9 months from now at quantity 250 000. The list is almost endless, and useful comparison data is virtually impossible to gather.

Hint: It often pays to conduct benchmarks with a few representative designs before undertaking serious cost calculations and making a misguided choice. This also helps to obtain realistic timing figures that take into account interconnect delays.

2.6 THE FPL DESIGN FLOW

Front-end design

The front-end flow is essentially the same as for ASICs. However, depending on the target product, there are a couple of particularities that affect architectural choices. In the occurrence of the popular coarse-grained FPGAs, these include:

- ± Look-up tables are cheap and typically come in chunks of 64 entries each (6 inputs)
- − Routing dominates over gate delay (due to configuration switches and larger dies). ⇝ Large, deep combinational networks put at a disadvantage.
- − Routing resources are limited. ⇝ The ideal architecture consists of many loosely connected circuit blocks, each of which fits into one logic cell, logic slice, adaptive logic module, or whatever is the name of the configurable function block.
- + Flip-flops come in generous numbers. ⇝ Pipelining is essentially free.
- ∼ Data and/or state coding schemes other than minimum bit encoding sometimes yield better solutions. ⇝ Check one-hot, Gray, and Johnson coding, for instance.
- + Parts come with sophisticated on-chip clock preparation circuits (nets, drivers, PLLs), but their number is limited. ⇝ A few large clock domains work best.
- − Asynchronous reset networks compete for global interconnect resources with clocks in some products. ⇝ Synchronous or partial resets tend to facilitate routing.[6]
- + Parts come with sophisticated input/output circuits (adjustable, LVDS, synchronization).
- + Parts include on-chip block RAMs (depending on product).
- + Many parts include weakly configurable datapath units (discussed in section 2.5).
- ± Hardwired function blocks (such as datapaths, multipliers, adders, and memories) come with fixed word widths. ⇝ While it may be difficult to make good use of them, they usually outperform LUT-based alternatives.
- + Some parts are available with one or more on-chip microcontrollers.
- − Devices come in thousands of variations. ⇝ May be confusing.
- − Parts come in fixed sizes. ⇝ Circuit complexity matters mostly when up- or downgrading from one size to the next.
- − Tools may make suboptimal decisions without designers becoming aware of.

Observation 2.1. *As all resources come with coarser granularities, the cost matrix of FPGAs is not the same as with ASICs. Generally speaking, it pays to be aware of the realities of the target platform before writing RTL synthesis code.*

[6] (Re-)configuring an FPL device can be understood as the strongest possible way to re-initialize the circuit's state, making it possible to dispense with a separate reset mechanism in certain applications.

Back-end design

Back-end design for field-programmable logic (FPL) differs considerably from the one depicted in fig.1.13. As FPL parts come with everything prefabricated, there is no need for actually placing cells or for routing interconnect lines. Instead, the gate-level netlist obtained from HDL synthesis is mapped onto the existing configurable cells in the target device and gets reoptimized to make the best possible usage of the logic resources available. EDA software further decides how to configure the switches and line drivers such as to obtain the wiring specified in the netlist. The combined result is then converted into a **configuration bit stream** for download into the FPL device. As FPGA and CPLD products come with many diverse architectures, FPL vendors make available proprietary tools for this procedure.

Apart from short turnaround times, several more factors make the design process simpler and more efficient, and so contribute to the success of FPL.

- Built-in processor cores, interfaces, and other standard functions greatly help to accelerate the FPL design cycle when compared to a custom design where the same functionality must be obtained from macrocells and virtual components.
- Many issues that must be addressed in extenso when designing a custom IC are implicitly solved. There is no need to agonize over subordinate details such as I/O subcircuits, clock and power distribution, embedded memories, testability, and more, see fig.2.15.
- Design tools are more affordable and up-front costs considerably lower than with any other hardware alternative.
- To aid with debugging, vendors provide so-called "signal taps" (Altera) or "chip scopes" (Xilinx) that help monitoring the waveforms on user-selected circuit nodes. These can be thought of as Virtual Components (VCs) that get temporarily inserted into the payload circuitry to sample signals and to store the values in on-chip memories for later inspection.

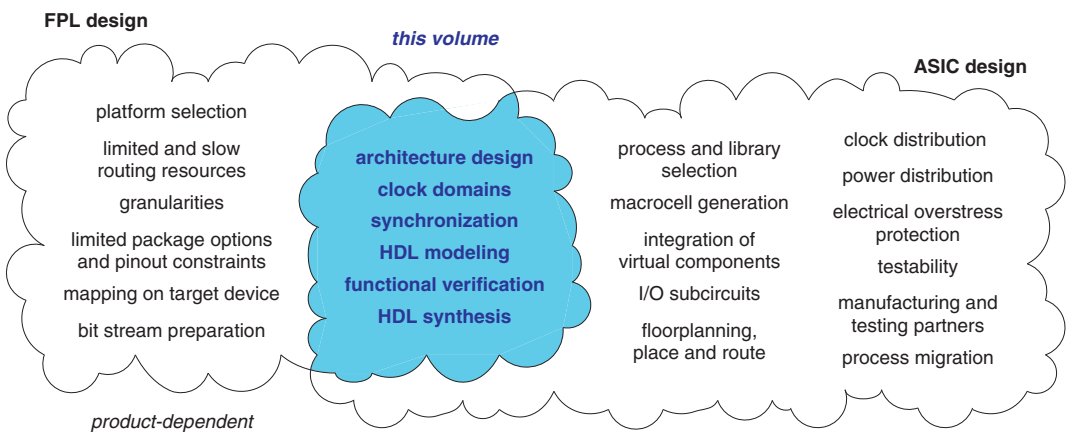


FIGURE 2.15

Primary concerns of FPL customers and full-custom ASIC designers.

Whoever has acquired the skills for designing full-custom ICs is in an excellent position for developing semi-custom ICs and for working with FPL, but not necessarily the other way round. The present volume has been written with all three approaches in mind and begins with those topics that matter for all audiences. If you ultimately plan to limit yourself to configuring FPL devices, you may be satisfied with skipping the sections from 7.2.4 onwards through 7.3. Conversely, you should then pay special attention to section 3.2.9.

Technical details on commercial FPL devices are distributed over thousands of datasheets and white papers prepared by Altera, Xilinx, and other vendors named in table 2.2. Open websites such as [13] [14] help to keep track of products and manufacturers. More condensed background information is available from references such as [15] [16] [17] [18]. [19] specifically discusses FPGAs in image and video processing applications.

2.7 CONCLUSIONS

Field programmable logic is ideal for

- Prototyping and other
- Situations where agility is crucial because specifications are subject to change at any time,
- Products that sell in modest quantities or where time to market is paramount, and for
- Products that need to be reconfigured from remote.