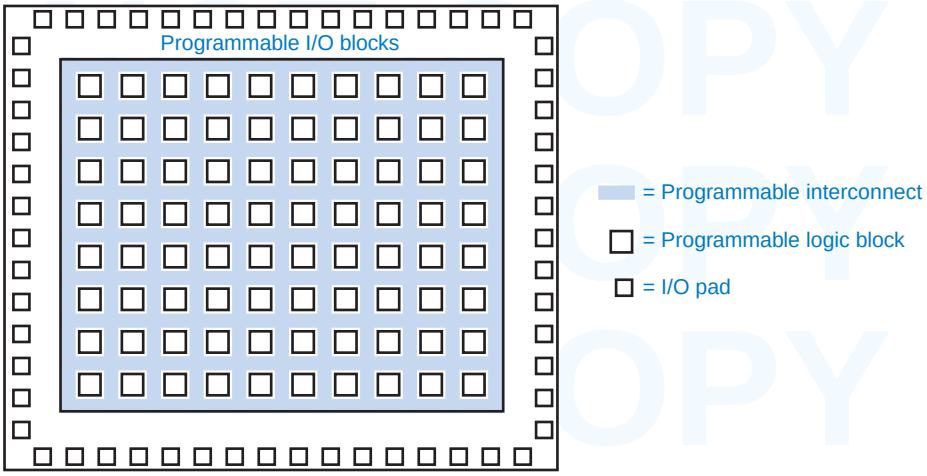

Figure 10-43
General FPGA chip architecture.

10.6 Field-Programmable Gate Arrays

A field-programmable gate array (FPGA) is kind of like a CPLD turned inside-out. As shown in Figure 10-43, the logic is broken into a large number of programmable logic blocks that are individually smaller than a PLD. They are distributed across the entire chip in a sea of programmable interconnections, and the entire array is surrounded by programmable I/O blocks. An FPGA's programmable logic block is less capable than a typical PLD, but an FPGA chip contains a lot more logic blocks than a CPLD of the same die size has PLDs.

Xilinx, Inc. invented FPGAs, and in this section we'll use one of their popular families, the XC4000E, to illustrate FPGA architecture.

10.6.1 Xilinx XC4000 FPGA Family

configurable logic block (CLB)

The programmable logic blocks in the Xilinx XC4000E family of FPGAs are called *configurable logic blocks (CLBs)*. The smallest part, the XC3003E, contains a 10×10 array of CLBs, and the largest, the XC4025E, contains a 32×32 array for a total of 1,024 CLBs. Xilinx also created extended XC4000EX and XC4000XL families, based on the XC4000E family, that have additional resources and features not discussed here. The largest member of the extended families, the XC4085XL, has 3,136 CLBs. Table 10-9 from Xilinx shows the family members that were available in 1999.

Like a CPLD family, the XC4000 family spans a range of device sizes and input/output capabilities. The table column labeled "Max. User I/O" refers to the maximum number of input/output blocks that are provided on-chip. However, XC4000 devices are available in a variety of packages, and not all of the I/Os are brought out to external pins of the smaller packages. As with CPLDs, the FPGA user has the opportunity to migrate a design from smaller to larger devices in the same package, or from a smaller package to a larger one.

Table 10-9 Resources in Xilinx XC4000-series FPGAs.

Device	CLB Matrix	Total CLBs	Max. User I/O	Flip-Flops	Max. RAM bits (no logic)	Max. Gates (no RAM)	Typical Gate Range (Logic and RAM)
XC4002XL	8×8	64	64	256	2,048	1,600	1,000–3,000
XC4003E	10×10	100	80	360	3,200	3,000	2,000–5,000
XC4005E/XL	14×14	196	112	616	6,272	5,000	3,000–9,000
XC4006E	16×16	256	128	768	8,192	6,000	4,000–12,000
XC4008E	18×18	324	144	936	10,368	8,000	7,000–15,000
XC4010E/XL	20×20	400	160	1,120	12,800	10,000	7,000–20,000
XC4013E/XL	24×24	576	192	1.536	18,432	13,000	10,000–30,000
XC4020E/XL	28×28	784	224	2,016	25,088	20,000	13,000–40,000
XC4025E	32×32	1,024	256	2,560	32,768	25,000	15,000–45,000
XC4028EX/XL	32×32	1,024	256	2,560	32,768	28,000	18,000–50,000
XC4036EX/XL	36×36	1,296	288	3,168	41,472	36,000	22,000–65,000
XC4044XL	40×40	1,600	320	3,840	51,200	44,000	27,000–80,000
XC4052XL	44×44	1,936	352	4,576	61,952	52,000	33,000–100,000
XC4062XL	48×48	2,304	384	5,376	73,728	62,000	40,000–130,000
XC4085XL	56×56	3,136	448	7,168	100,352	85,000	55,000–180,000

The “Flip-Flops” column counts all of the flip-flops in the device, two per CLB and two per I/O block, as we’ll see later. Only a fraction of the available flip-flops are used in a typical design, but this number is a figure of merit that designers look at when roughly sizing an FPGA for an design. The “Max. RAM Bits” column is another such figure of merit. As we’ll see, instead of being used for logic, each CLB can be configured as a small SRAM storing up to 32 bits.

The “Max. Gates” number is fuzzy. For the XC4002XL, the table claims that each CLB can perform the same function as about 25 gates in a discrete design. The XC4003E is even better, at 30 gates per CLB. Is this reasonable? And just what is a “gate”? Do XORs or wide NAND gates count as one gate? You can decide for yourself later, after you know more about the CLB architecture. At the same time, you can decide whether this column was created by engineering or by marketing people.

The last column of the table must certainly have been written by marketing folks, since the high end of each “typical” gate range is higher than the maximum in the column before it! Actually, there *is* a reasonable explanation for this seeming inconsistency. This column assumes that 20–30% of the CLBs are used for memory rather than logic, at 32 bits of SRAM per CLB. The minimum

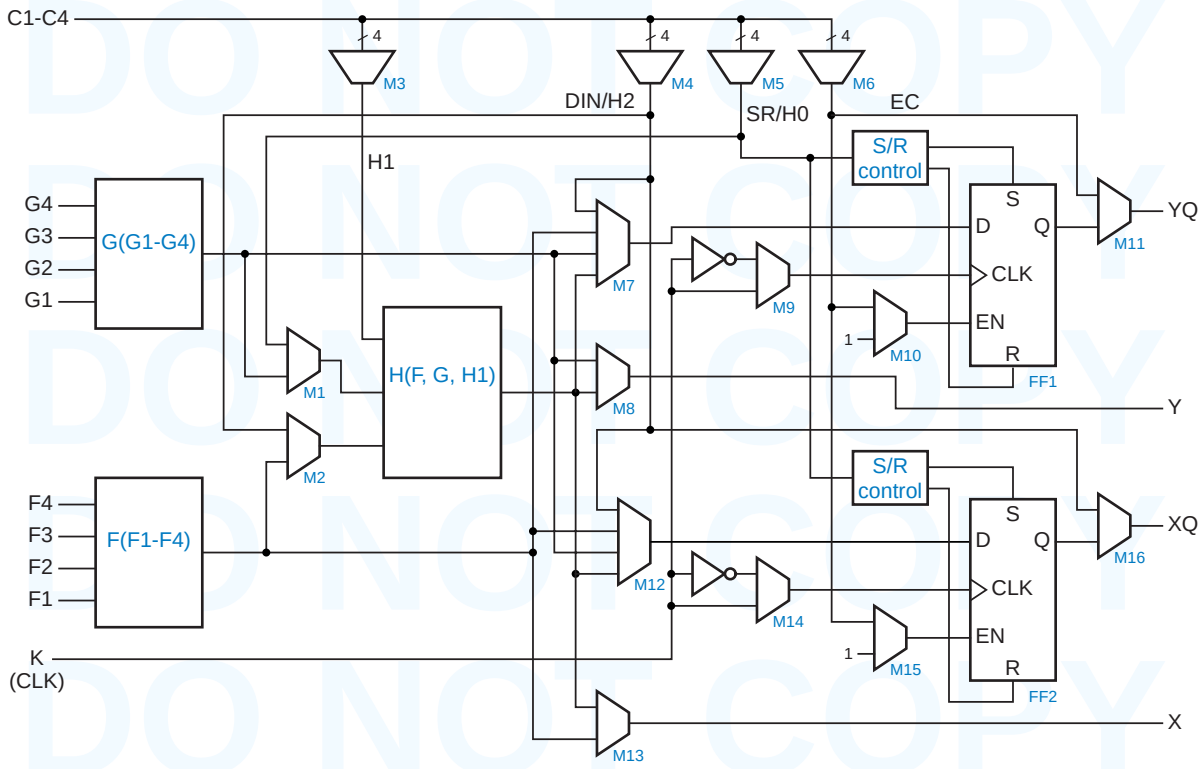

Figure 10-44 XC4000 configurable logic block.

gate-level implementation of an SRAM cell is a D latch using four gates (Figure X7.27 on page 650), so we can count each CLB as 128 gates when it's used as SRAM. The number in the last column, therefore, is the number of gates used for logic *plus* the number of equivalent gates used for SRAM.

10.6.2 Configurable Logic Block

Since an FPGA can have lots and lots of CLBs, it's important that we understand them first! Figure 10-44 shows the internal structure of an XC4000-series CLB.

The CLBs most important programmable elements are the logic-function generators F, G, and H. Both F and G can perform any combinational logic function of their four inputs, and H can perform any combinational logic function of its three inputs.

How do F, G, and H work? Given the task of building a "universal" function generator for 4-input logic functions, what kind of gate-level circuit would you come up with? Think about it, and we'll come back to it later.

As in our CPLD discussion, the trapezoidal boxes in Figure 10-44 represent programmable multiplexers. Notice that the outputs of F and G as well as

additional CLB inputs can be directed to inputs of H by multiplexers M1–M3, so it's possible to realize some functions of more than four inputs. A taxonomy of the functions that can be realized by F, G, and H in a single CLB is given below:

- Any function of up to four variables, plus any second function of up to four unrelated variables, plus any third function of up to three unrelated variables.
- Any single function of five variables (see Exercise 10.34).
- Any function of four variables, plus some second functions of six unrelated variables.
- Some functions of up to nine variables, including parity checking and a cascadable equality checker for two 4-bit inputs (see Exercises 10.35 and 10.36).

With appropriate programming of multiplexers M7–M8 and M12–M13, the outputs of the function generators can be directed to CLB outputs X and Y, or they can be captured in edge-triggered D flip-flops FF1 and FF2. The flip-flops can use the rising or falling edge of a common clock input, K, as selected by multiplexers M9 and M14. They can also make use of a clock-enable signal, EC, selected by M10 and M15. The sources of EC and three other internal signals are selected from a set of four miscellaneous inputs C1–C4 by multiplexers M3–M6 at the top of the CLB.

The XQ and YQ outputs of the CLB carry the flip-flop outputs out of the CLB. If a flip-flop is not used in the CLB, multiplexer M11 or M16 can select XQ or YQ to be a "bypass output" that is simply a copy of a CLB input selected by M4 or M6.

The block labeled "S/R control" from each flip-flop determines whether the flip-flop is set or reset at configuration. It also determines whether the flip-flop responds to a global set/reset signal (not shown) or to the CLB's SR signal selected by multiplexer M5.

Wow, that's a lot of programmability! Naturally, the configuration of CLBs within an XC4000 part, whether it has 3,136 CLBs or only 64, is not carried out by hand. The manufacturer provides a fitter tool that allocates, configures, and connects CLBs to match a higher-level design description written in ABEL, VHDL, Verilog, or schematic form.

Let's come back to our question of how to build a universal function generator for 4-input logic functions. It's a hard problem if you think about it at the gate level, but pretty easy if you think about it from another point of view. Any 4-input logic function can be described by its truth table, which has 16 rows. Suppose we store the truth in a 16-word by 1-bit-wide memory. When we apply the function's four input bits to the memory's address lines, its data output is the value of the function for that input combination.

This is exactly the approach that was taken by the FPGA designers at Xilinx. Function generators F and G are actually just very compact and fast 16×1 SRAMs, and H is an 8×1 SRAM. When a CLB is used to perform logic, the truth tables of logic functions F, G, and H are loaded into SRAM at configuration time from an external read-only memory. The programmable multiplexers in Figure 10-44 are also controlled by "SRAM," actually individual D latches that are also loaded at configuration time. This programming is done for all of the CLBs in the FPGA.

Besides convenience, using memory to store the truth tables has another important benefit. Any XC4000 CLB can be configured at start-up to be used as memory rather than logic. Several different modes can be configured:

- *Two 16×1 SRAMs.* F and G are used as SRAMs with independent address and write-data inputs. They share a common write-enable input, however.
- *One 32×1 SRAM.* The same four address bits are used for F and G, and a fifth address bit is applied to the H function generator and the write-enable circuitry to select between F and G, the upper and lower halves of the memory.
- *Asynchronous or synchronous*. For write operations, the SRAMS above can be configured to have "normal" asynchronous latching behavior, or they can be configured to occur on a designated edge of the K clock signal.
- *One 16×1 dual-port SRAM*. The two sets of address inputs are used to independently read and write different locations in the same SRAM. Only synchronous write operations are supported in this mode.

In these modes, function inputs F1–F4 and G1–G4 supply address, other CLB inputs H0–H2 provide data inputs and the write-enable signal, and data outputs are provided on the F and G generator outputs and can be captured in flip-flops FF1 and FF2 or exit at CLB outputs X and Y.

10.6.3 Input/Output Block

I/O block (IOB)

The structure of the XC4000 *I/O block (IOB)* is shown in Figure 10-45. An I/O pin can be used for input or output or both.

The XC4000 IOB has more "logic" controls than its cousin in the XC9500 CPLD. In particular, its input and output paths contain edge-triggered D flip-flops selectable by multiplexers M5–M7. Placing input and output flip-flops "up close" to the device I/O pins is especially useful in FPGAs. On output, relatively long delays from internal CLB flip-flop outputs to the IOBs can make it difficult to connect to external synchronous systems at very high clock rates. On input,

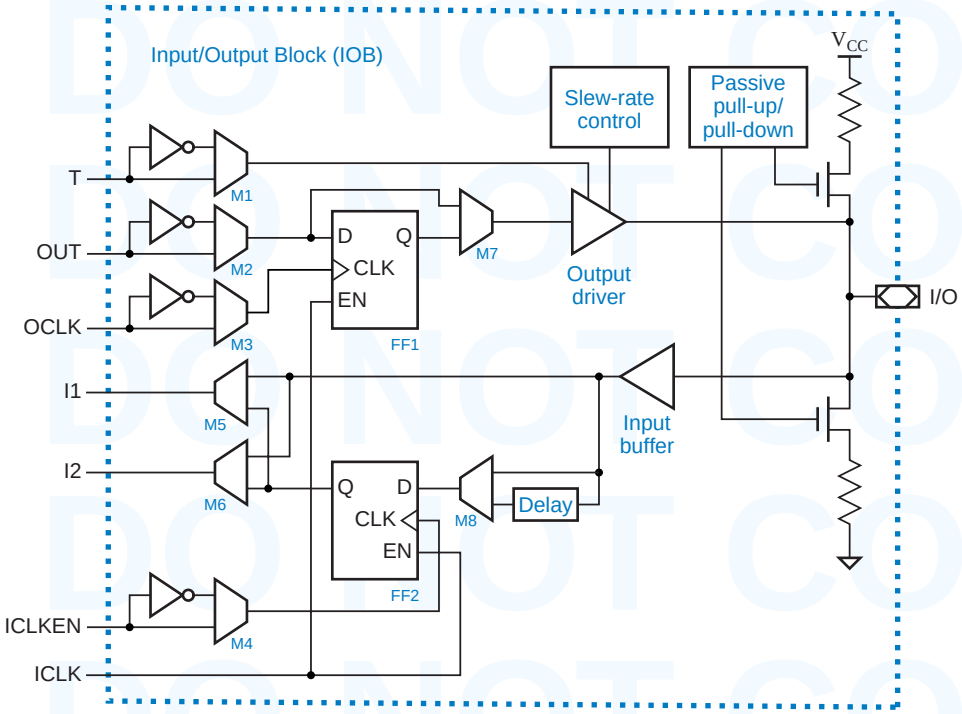

Figure 10-45
XC4000 I/O block.

long delays from the I/O pins to CLB flip-flop inputs can make it difficult to meet external system setup and hold times if external inputs are clocked directly into a CLB flip-flop without being captured first by a flip-flop at the IOB pin. Of course, using IOB flip-flops is possible only if the FPGA's external interface specifications allow "pipelining" of inputs and outputs.

For pipelined inputs, the XC4000 IOB actually goes one step further by providing a delay element, selectable by M8, in series with the D input of input flip-flop FF2. The effect of this element is to delay the D input relative to the FPGA's internal copies of the system clock, guaranteeing that the input will have a zero hold-time requirement with respect to the external system clock. This benefit comes at the expense of increased setup time, of course.

The IOB's other logic controls are selectable polarity, using multiplexers M1–M4, for its four inputs that come from the CLB array via the programmable interconnect. These inputs, OUT, T, OCLK, and ICLKEN, are the output bit, its three-state enable, the output clock, and the input clock enable, respectively.

Like the XC9500 IOB, the XC4000 IOB also has analog controls. The output driver's slew rate is programmable, and a pull-up or pull-down resistor may be connected to the I/O pin.

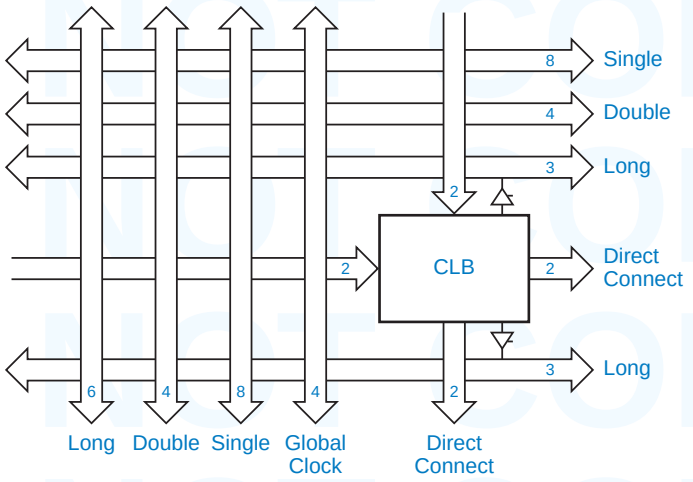

Figure 10-46 XC4000 general interconnect structure.

10.6.4 Programmable Interconnect

Well, we saved the best for last. The XC4000 programmable interconnect architecture is fascinating example of a structure that provides rich, symmetric connectivity in a small silicon area.

As we showed in Figure 10-43 on page 884, each CLB in an FPGA is embedded in the interconnect structure, which is really just wires with programmable connections to them. Figure 10-46 gives a little more detail of the XC4000's connection scheme. Wires are not really "owned" by any one CLB, but an XC4000 CLB array is created by tiling the chip with exactly the structure shown in the figure. For example, 100 copies of this figure make the 10×10 CLB array of an XC4003.

The number in each arrow indicates the number of wires in that signal path. Thus, you can see that a CLB has two wires (outputs) each going to the CLBs immediately below and to the right of it. It also connects to three groups of wires above, one below, and four to its left. Signals on these wires can flow in either direction.

The four wires in the group labeled "Global Clock" are optimized for use as clock inputs to the CLBs, providing short delay and minimal skew. The two "Singles" groups are optimized for flexible connectivity between adjacent blocks, without the small number and unidirectional limitation of wires in the "Direct Connect" groups.

It's possible to connect a CLB to another that's more than one hop away using "Single" wires, but they have to go through a programmable switch for each hop, which adds delay. Wires in the "Doubles" groups travel past two CLBs before hitting a switch, so they provide shorter delays for longer connections.

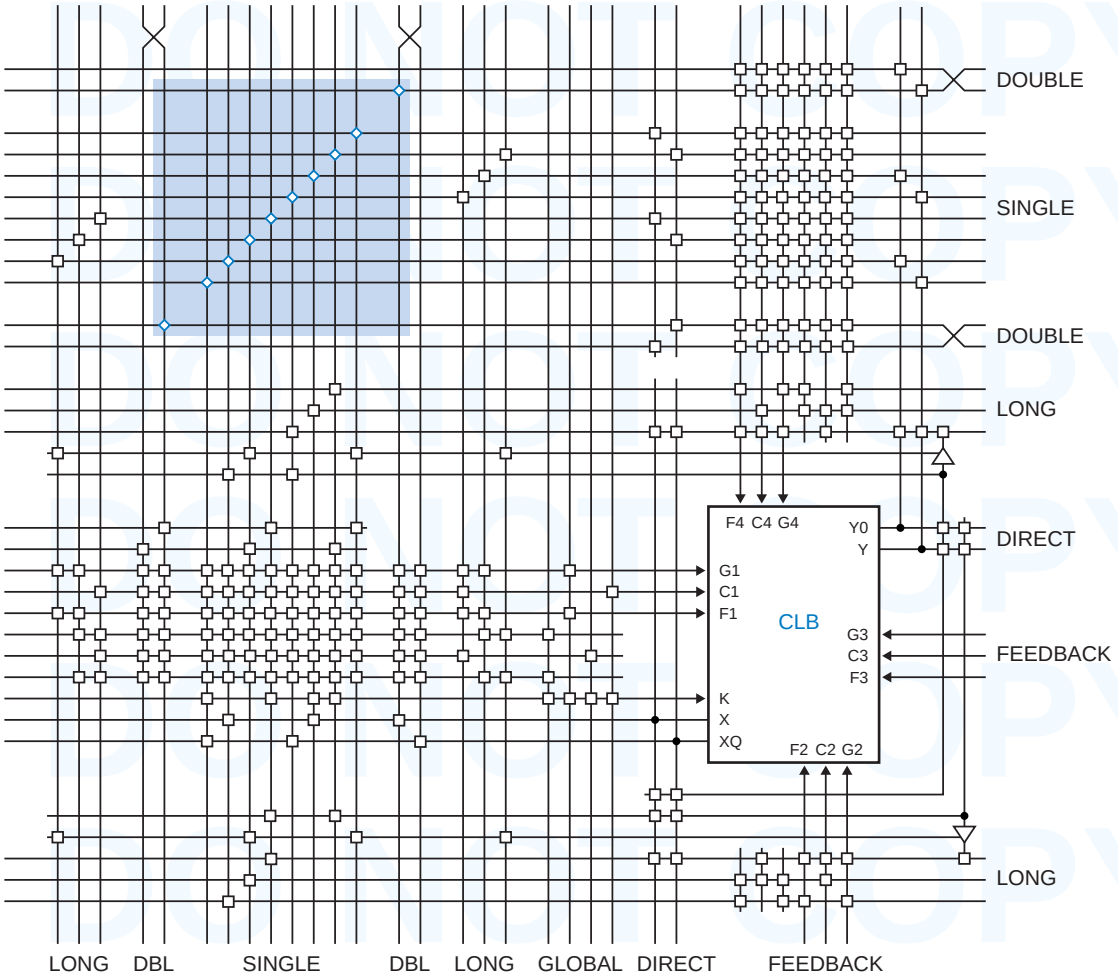

Figure 10-47 XC4000 CLB and wire connection details.

For really long connections, the "Long" groups don't go through any programmable switches at all; instead, they travel all the way across or down a row or column and are driven by three-state drivers near the CLB.

Figure 10-47 shows a CLB and wires in a lot more detail. The small squares are programmable connections—a horizontal wire is connected or not to a vertical one, depending on the state of the programming bit (again, in a latch) for that switch. Additional, specialized programmable interconnect is provided at the edges of the CLB array for connections to the IOBs.

In the figure, the shaded area in color is called a *programmable switch matrix (PSM)*. The PSM is shown in more detail in Figure 10-48. Each of the

programmable switch matrix (PSM)

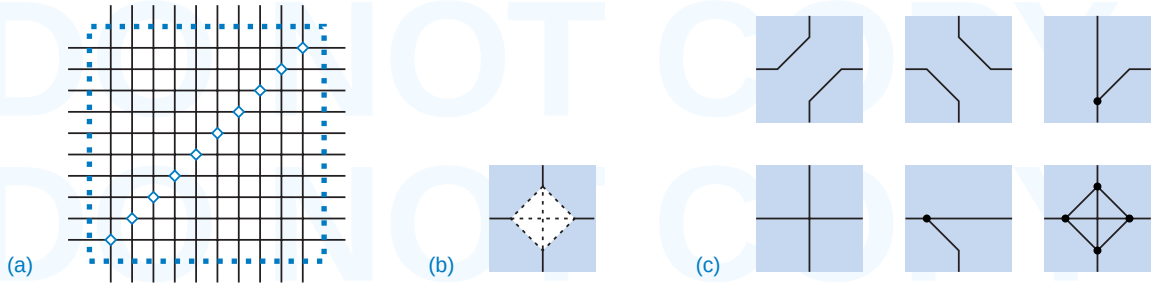

Figure 10-48 XC4000 programmable connections: (a) programmable switch matrix (PSM); (b) programmable switch element (PSE); (c) a few possible connections.

programmable switch element (PSE)

diamonds in (a) is a *programmable switch element (PSE)* that can connect any line to any other, as shown in (b). With four lines, there are 6 possible pairwise connections as shown in (b), and the PSE has a transmission gate for each one of them. Some, none, or all of the transmission gates in a PSE may be enabled—again, by configuration bits stored in latches. Thus, many different connection patterns are possible, as shown in (c).

The PSM is essential for hooking things up in the wiring structure of Figure 10-47. By enabling and disabling connections, PSEs extend or isolate wire segments in the "Single" and "Double" groups. More importantly, the PSM allows signals to "turn the corner" by connecting a horizontal wire to a vertical one. Without this, CLBs would not be able to connect to others in a different row or column of the array.

While the PSM is essential, using it has a price—signals incur a small delay each time they hop through a PSE. Therefore, high-quality FPGA fitter software searches for not just any CLB placement and wire connections that work. The "placement and routing" tool spends a lot of time trying to optimize device performance by finding a placement that allows short connections, and then routing the connections themselves.

Like CPLDs, FPGAs are judged by the flexibility of their architectures and the consistency of the results obtained from a fitter after small design changes are made. There's nothing more frustrating than making a small change to a large design and finding that it no longer meets timing requirements. Thus, FPGA manufacturers have learned to provide "extra" resources in their architectures to help ensure consistent results.

GOOD PRACTICE Placement and routing is actually a pretty well understood problem, because it is the major part of the "back end" of any custom chip design. Thus, the same kind of tools and the same tool vendors are involved with placement and routing for both FPGAs and ASICS. So, you might like to consider any FPGA experience that you get to be good practice for ASIC design!