2 Background Information

2.1 Introduction

This chapter introduces the two main fields of research, FPGA routing algorithms and FPGA routing architecture, that are studied in this thesis. Section 2.2 provides some necessary background information that is assumed in various discussions, particularly in Chapter 3, about routing software. Section 2.3 describes several commercially available FPGA devices to provide a point of reference for the FPGA model that is used throughout this work, and particularly for the routing architecture results that are presented in Chapters 4 and 5.

2.2 Routing Algorithms

While the focus of this thesis is routing, this chapter begins with an overview of the entire CAD process that is necessary to implement a circuit in an FPGA. A typical CAD system for FPGAs consists of several interconnected programs as illustrated in Figure 2.1. The input to the CAD system is a functional description of a network, usually expressed in a standard format such as boolean equations. The equations are read by a logic optimization [Bray86] [Greg86] tool, which performs manipulations of the equations so as to optimize area, delay, or a combination of area and delay. This step usually performs the equivalent of an algebraic minimization of the boolean equations and is appropriate when implementing a circuit in any medium, not just FPGAs. To transform the boolean equations into a circuit of FPGA logic cells, the optimized network is fed to a technology mapping program [Kahr86] [Keut87] [Fran91]. This step maps the equations into logic cells, which also presents opportunity to optimize, either to minimize the total number of logic cells required (area optimization) or the number of logic cells in time-critical paths (delay optimization). The circuit of logic cells is then passed to a placement program [Hana72] [Rose85] [Sech87], which selects a specific location in the FPGA for each logic cell. Typical placement algorithms usually attempt to minimize the total length of interconnect required for the resulting placement.

The final step in the CAD system is performed by the routing software, which allocates the FPGA's routing resources to interconnect the placed logic cells. The routing tools must ensure that 100 percent of the required connections are formed, and may be required to maximize the speed performance of time-critical connections. Finally, the CAD system's output is fed to a programming unit that is used to configure the FPGA. Since routing software is the key step in the CAD system for the purposes of this thesis, the remainder of this section provides a brief introduction to the subject.

2.2.1 Routing Terminology

Software that performs automatic routing has existed for many years, with the first algorithms designed to route printed circuit boards. Over the years there have been many publications concerning routing algorithms, so that the problem is well defined and

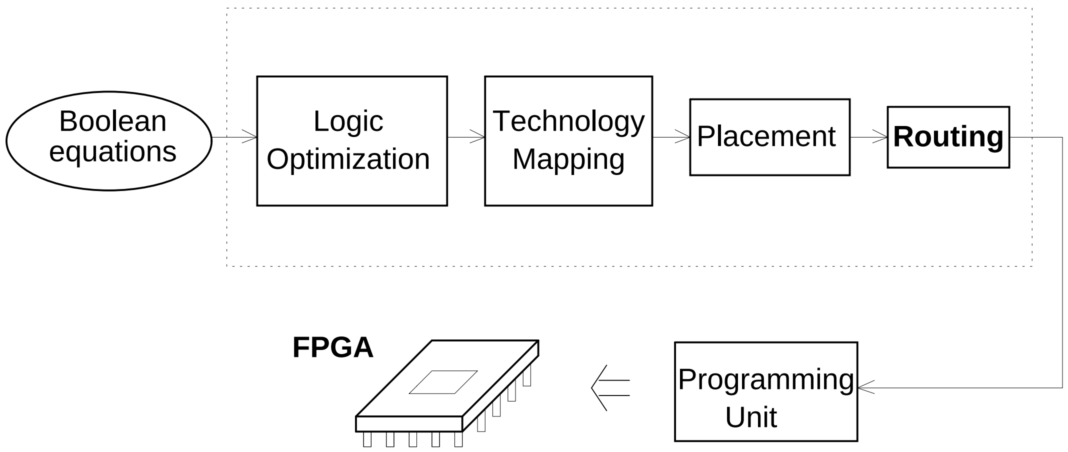

Figure 2.1 - *A Typical FPGA CAD System*

understood. The following list gives common routing terms, as they are defined for FPGA routing in this thesis:

- *Pin* - a logic cell input or output.
- *Connection* - a pair of logic cell pins that are to be electrically connected.
- *Net* - a set of logic cell pins that are to be electrically connected. A net can be divided into one or more connections.
- *Wiring segment* - a straight section of wire that is used to form part of a connection.
- *Routing switch* - a device that is used to electrically connect two wiring segments.
- *Track* - a straight section of wire that spans the entire width or length of a routing channel. A track can be composed of a number of wiring segments of various lengths.
- *Routing channel* - the rectangular area that lies between two rows or two columns of logic cells. A routing channel contains a number of tracks.

2.2.2 General Approach to Routing

Because of the combinatorial complexity involved, the solution of large routing problems usually requires a "divide and conquer" strategy. Following this philosophy, routing can be solved by a three-step process [Loren89]:

1. Partition the routing resources into routing areas that are appropriate for both the device to be routed and the routing algorithms to be employed.
2. Use a *global router* to assign each net to a subset of the routing areas. The global router does not choose specific wiring segments and routing switches for each connection, but rather it creates a new set of restricted routing problems.

3. Use a *detailed router* to select specific wiring segments and routing switches for each connection, within the restrictions set by the global router.

The advantage of this approach is that each of the routing tools can more effectively solve a smaller part of the routing problem. More specifically, since a global router need not be concerned with allocating wiring segments or routing switches, it can concentrate on more global issues, like balancing the usage of the routing channels. Similarly, with the reduced number of detailed routing alternatives that are available for each connection because of the restrictions introduced by a global router, a detailed router can focus on the problem of achieving connectivity. Its limited scope enables a detailed router to concentrate on resolving contention for routing resources that may exist among different nets.

The above routing strategy has been adopted in this thesis for FPGA routing. The routing resources are partitioned into horizontal and vertical routing channels.

2.2.3 Introduction to Global Routing

This section introduces global routing by describing the LocusRoute global routing algorithm [Rose90a] for standard cells. Although there are many other published techniques for global routing [Loren89] [Sech88] [Cong88], this specific algorithm is described as an example because a modified version of it is employed for FPGA global routing in this thesis. This algorithm has been chosen for FPGAs because, as described below, its primary goal is to balance the usage of the routing channels. This is important for FPGAs because the number of tracks per channel is pre-determined. Note that the description below is based on the standard-cell version of LocusRoute, and the main difference between this and the FPGA version is the definitions of the routing channels - the standard-cell program assumes only horizontal routing channels, whereas the FPGA version uses both horizontal and vertical channels.

2.2.3.1 The LocusRoute Global Routing Algorithm

The LocusRoute global router views the global routing problem as consisting of three main tasks:

1. For nets comprising more than two pins, determine which pairs of pins to connect together. This step decomposes a multi-point net into a set of two-point connections.
2. Determine a path through the routing channels for each connection.
3. Optimize the solution so that the usage of all of the routing channels is balanced.

The first task is solved by finding a minimum-spanning tree [Prim57] for each net. Basically, this technique breaks a net into a set of two point connections such that the total amount of interconnect required is minimized.

To solve the second task, LocusRoute models each routing channel as an array of grids, as shown in Figure 2.2. Each grid location contains a counter, originally set to zero, which is incremented by one for each connection that is globally routed through it. In this way, the algorithm is able to maintain a detailed account of the usage of each routing channel, so that it can avoid congestion. The algorithm considers alternative ways of routing each connection and chooses the one that passes through the least congested routing grids. Note that LocusRoute does not consider all of the possible ways that a connection can be routed, but rather it evaluates only a subset of the paths that have "two or fewer bends", as explained in [Rose90a].

After all of the connections have been globally routed once, LocusRoute optimizes the solution by sequentially ripping up and re-routing each connection. After repeating this procedure a small number of times, the final solution is output in a format suitable for the detailed router to be employed.

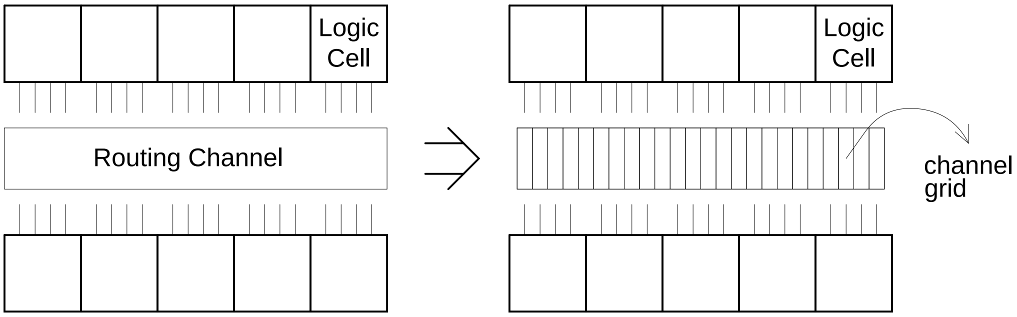

Figure 2.2 - *The Channel Grids Used by LocusRoute*

2.2.4 Introduction to Detailed Routing

This section provides an introduction to detailed routing by describing the *maze* routing technique. Although there exist many other detailed routing algorithms [Aker72] [Souk81] [Loren89], maze routing will be discussed because it is widely used due to its general applicability, and a variant of a maze router is employed as a comparison against the detailed routing algorithm for FPGAs that is described in Chapter 3.

2.2.4.1 The Lee Maze Router

Most maze routers can be considered to be a variant of the algorithm described in [Lee61]. This technique models the entire routing surface as a rectangular array of cells, where the size of each cell is defined so as not to violate the spacing rules for wiring segments. Connections are formed one at a time by selecting adjacent cells that reach from one end of a connection to the other. Once a grid location is occupied, either by a connection or by some sort of obstruction, it is marked as unusable. An array of routing cells is illustrated in Figure 2.3, where unusable cells are shaded and usable ones are not. The figure shows the detailed routes of three connections as they might be produced by a maze router.

The Lee algorithm implements the array of cells as a regular graph, with one vertex for each cell and one edge joining each pair of adjacent cells. A connection is routed by beginning at one of its ends and traversing the graph in a breadth first fashion until the other end is reached. The result is a diamond shaped wavefront that emanates from the first point, as illustrated in Figure 2.4. The numbers in the figure correspond to each step as the wavefront is propagated.

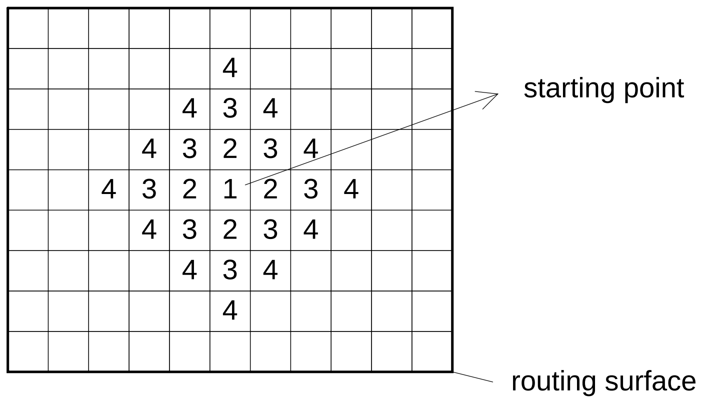

Figure 2.4 - *Maze Router Diamond Shaped Wavefront*

The main advantage of a maze router is that it is guaranteed to find a path from one end of a connection to the other, if one exists at the time the connection is routed. On the

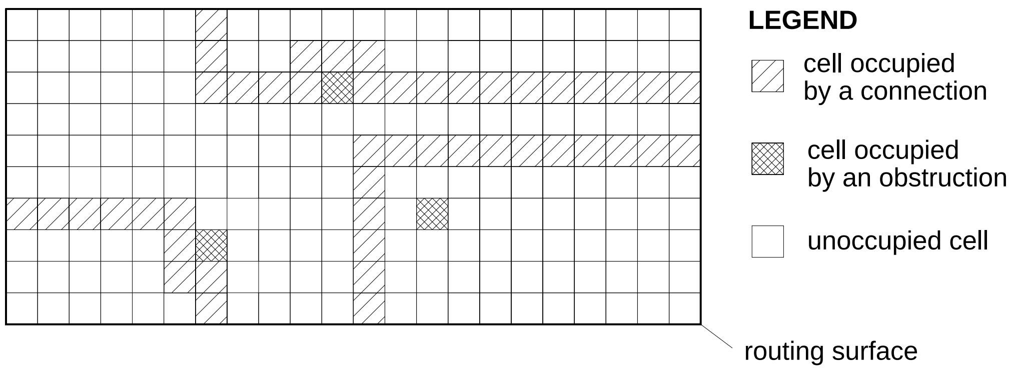

Figure 2.3 - *Maze Routers Model the Routing Surface by an Array of Cells*

other hand, because of its sequential nature a maze router is unable to consider the side-effects that the routing of one connection may have on another. Correspondingly, the main disadvantage of maze routing is the unnecessary blockage of as yet unrouted connections because of previous routing decisions.

2.3 Commercially Available FPGAs

This section provides a detailed description of three commercially available FPGA families, including those from Xilinx Co., Actel, and Altera. These particular FPGAs have been chosen because they are representative examples of state-of-the-art devices and they are in widespread use. Each device is described in terms of its general architecture, its choice of programmable cell, its routing architecture, and its CAD routing tools. Enough details are given, and in some cases specific comments are made, to show how the routing architecture of each device relates to the research contained in this thesis. In addition, at the end of the section, several recently introduced FPGAs are briefly described.

2.3.1 Xilinx FPGAs

The general architecture of Xilinx FPGAs is shown in Figure 2.5. It consists of a two-dimensional array of programmable cells, called Configurable Logic Blocks (CLBs), with horizontal routing channels between rows of cells and vertical routing channels between columns. Programmable resources are configured by Static RAM cells, and each routing switch is implemented as a specially designed transistor controlled by an SRAM bit. There are three families of Xilinx FPGAs, called the XC2000, XC3000, and XC4000 corresponding to first, second, and third generation devices. Table 2.1 gives an indication of the logic capacities of each generation by showing the number of CLBs and an equivalent gate count. The gate count measure is given in terms of "equivalent to an

MPGA of the same size." All FPGA manufacturers quote logic capacity by this measure, but it is questionable whether the figures quoted by each are realistic. The numbers given in Table 2.1, and in similar tables that appear later in this chapter, should be interpreted accordingly. The design of the Xilinx CLB and routing architecture differs for each generation, so they will each be described in turn.

2.3.1.1 Xilinx XC2000

The XC2000 CLB, shown in Figure 2.6, consists of a four-input look-up table and a D flip-flop [Cart86]. The look-up table can generate any function of up to four variables

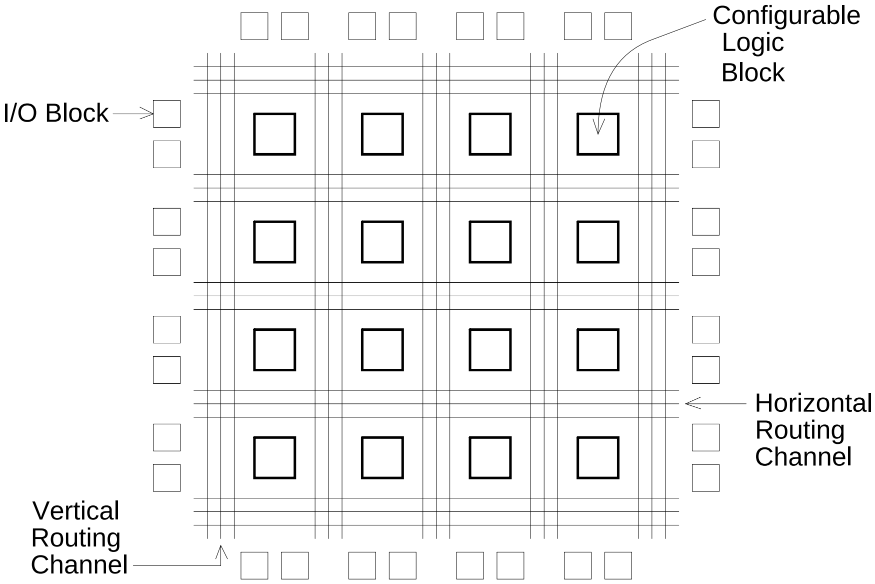

Figure 2.5 - *General Architecture of Xilinx FPGAs*

Series	Number of CLBs	Equivalent Gates
XC2000	64 - 100	1200 - 1800
XC3000	64 - 320	2000 - 9000
XC4000	64 - 900	2000 - 20000

Table 2.1 - *Xilinx FPGA Logic Capacities*

or any two functions of three variables. Both of the CLB outputs can be combinational, or one output can be registered.

As illustrated in Figure 2.7, the XC2000 routing architecture employs three types of routing resources: Direct interconnect, General Purpose interconnect, and Long Lines. Note that for clarity the routing switches that connect to the CLB pins are not shown in the figure. The Direct interconnect (shown only for the CLB marked with an ’*’) provides connections from the output of a CLB to its right, top, and bottom neighbours. For connections that span more than one CLB, the General Purpose interconnect provides horizontal and vertical wiring segments, with four segments per row and five segments per column. Each wiring segment spans only the length or width of one CLB, but longer wires can be formed because each switch matrix holds a number of routing switches that can interconnect the wiring segments on its four sides. Note that a connection routed with the General Purpose interconnect will incur significant routing delays because it must pass through a routing switch at each switch matrix. Connections that are required

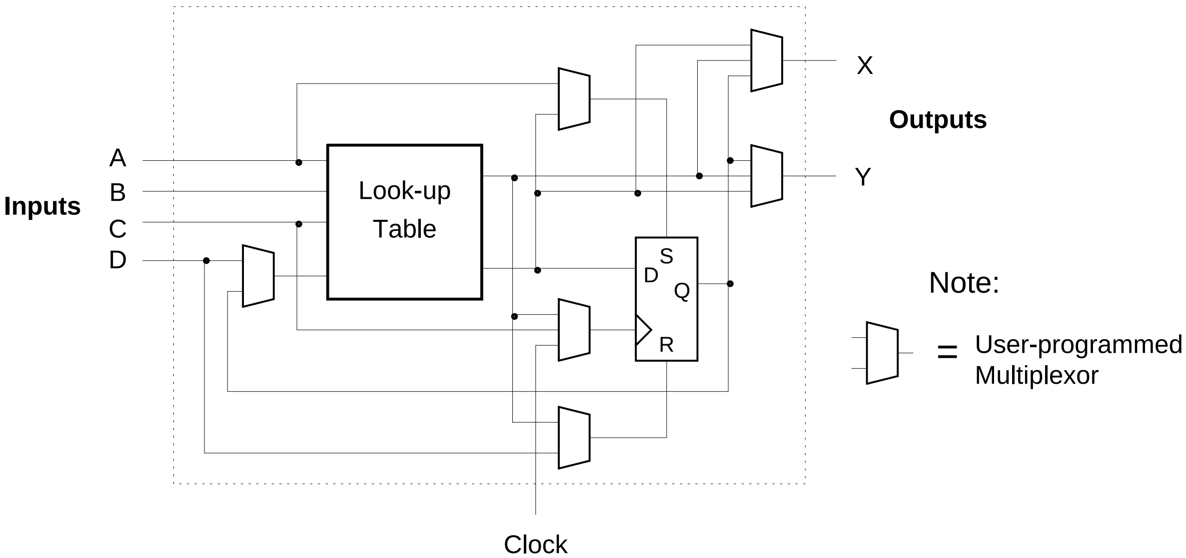

Figure 2.6 - *XC2000 CLB*

to reach several CLBs with low skew can use the Long Lines, which traverse at most one routing switch to span the entire length or width of the FPGA.

2.3.1.2 Xilinx XC3000

The XC3000 [Hsie88] is an enhanced version of the XC2000, featuring a more complex CLB and more routing resources. The CLB, as shown in Figure 2.8, houses a look-up table that can implement any function of five variables, any two functions of four variables, and some functions of up to seven variables. The CLB has two outputs, both of which may be either combinational or registered.

Figure 2.9 shows that the XC3000 routing architecture is similar to that in the XC2000, having Direct interconnect, General Purpose interconnect, and Long Lines. Each resource is enhanced: the Direct interconnect can additionally reach a CLB's left

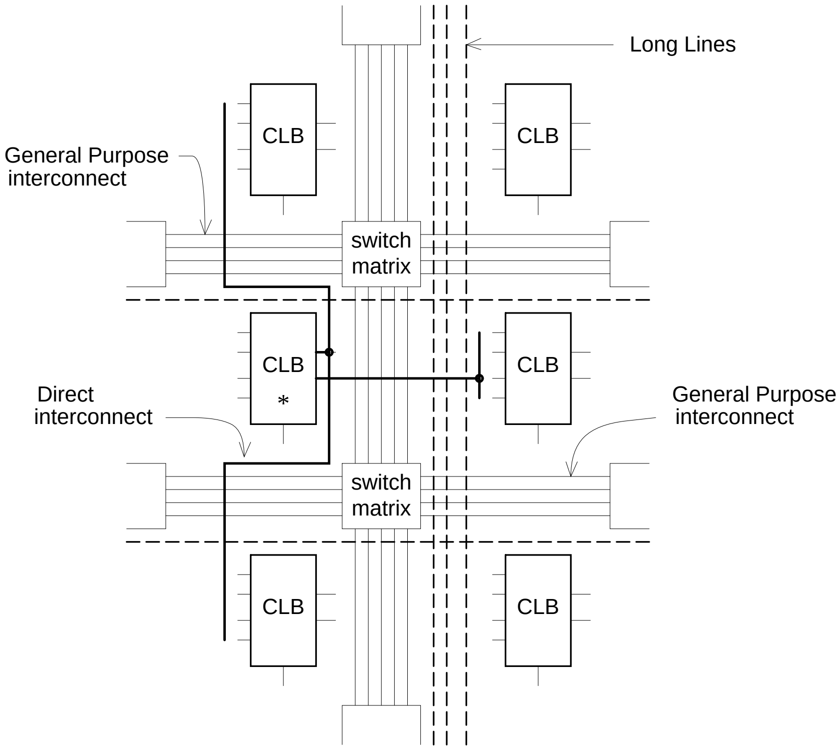

Figure 2.7 - *XC2000 Interconnect*

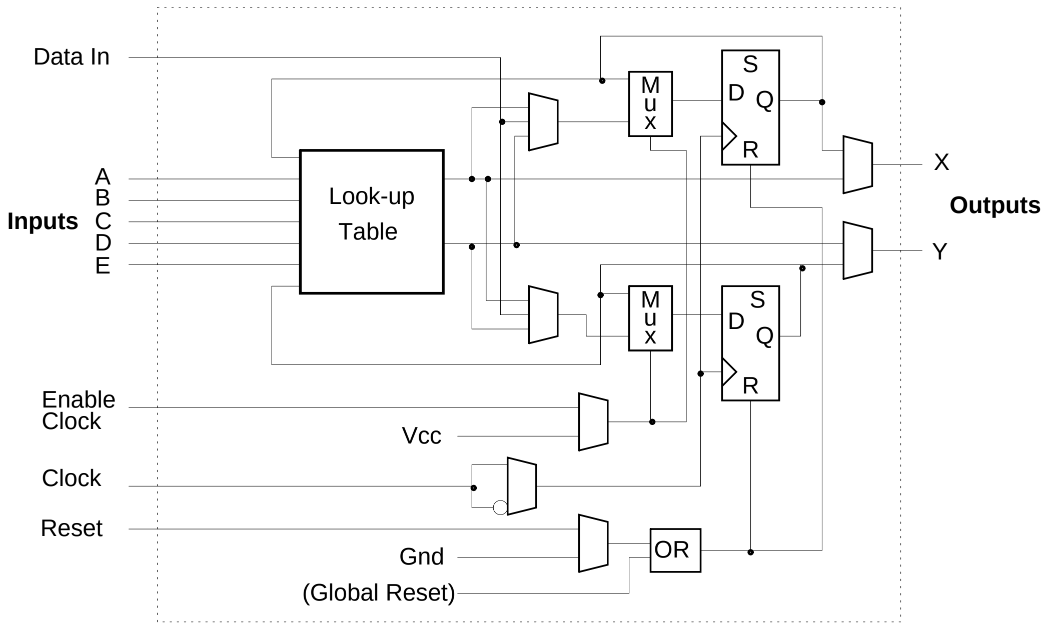

Figure 2.8 - *XC3000 CLB*

neighbour, the General Purpose interconnect has an extra wiring segment per row, and there are more Long Lines.

The XC3000 also contains switch matrices that are similar to those in the XC2000. Figure 2.9 depicts the internal structure of an XC3000 switch matrix by showing, as an example, that the wiring segment marked with an ’*’ can connect through routing switches to six other wiring segments. Although not shown in the figure, the other wiring segments are similarly connected, though not always to the same number of segments. This detail is included here because the results shown in Chapter 4 of this thesis suggest recommended values for the number of routing switches connectable to any wiring segment, as well as the number of wiring segments in a row or column. Those results indicate that, in terms of routability, the XC3000 contains too many routing switches per switch matrix and too few wiring segments in its rows and columns.

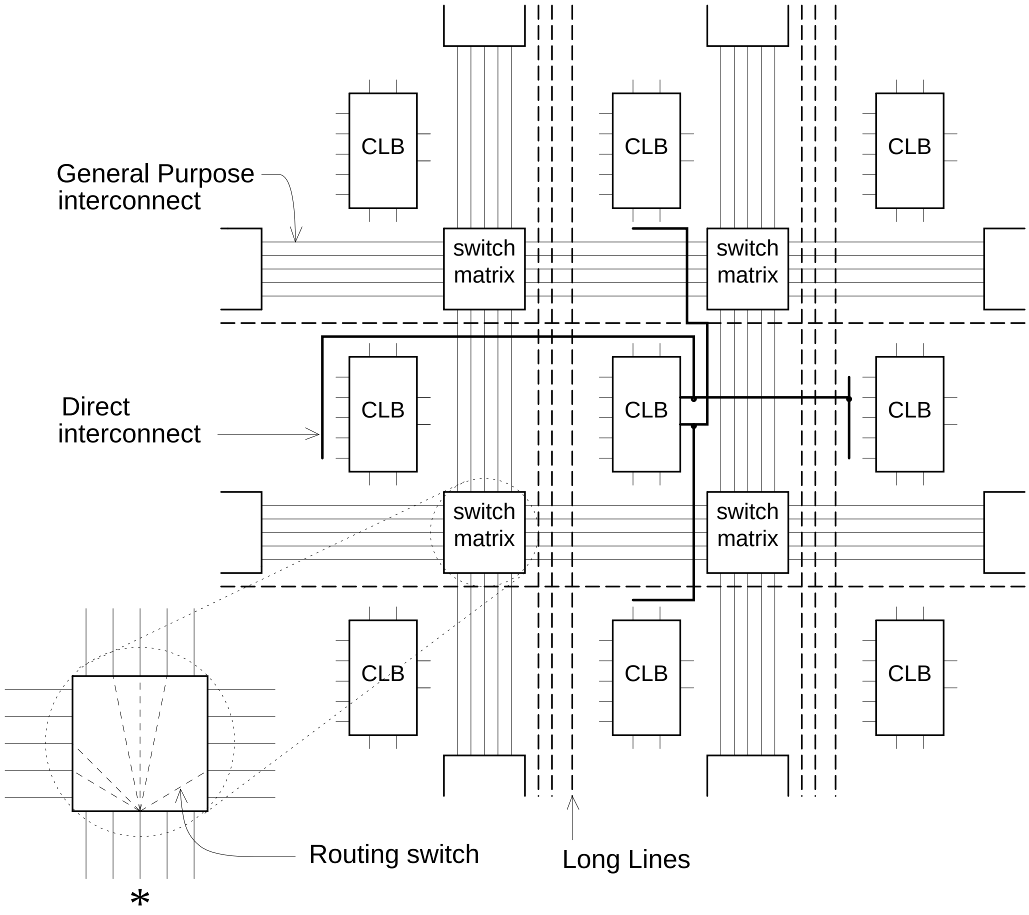

Figure 2.9 - *XC3000 Interconnect*

2.3.1.3 Xilinx XC4000

The XC4000 [Hsie90] features several enhancements over its predecessors. The CLB, illustrated in Figure 2.10, utilizes a hierarchical arrangement of look-up tables that yields a greater logic capacity per CLB than in the XC3000. The XC4000 CLB can implement two independent functions of four variables, any single function of five variables, any function of four variables together with some functions of five variables, or some functions of up to nine variables. The CLB has two outputs, which may be either combinational or registered.

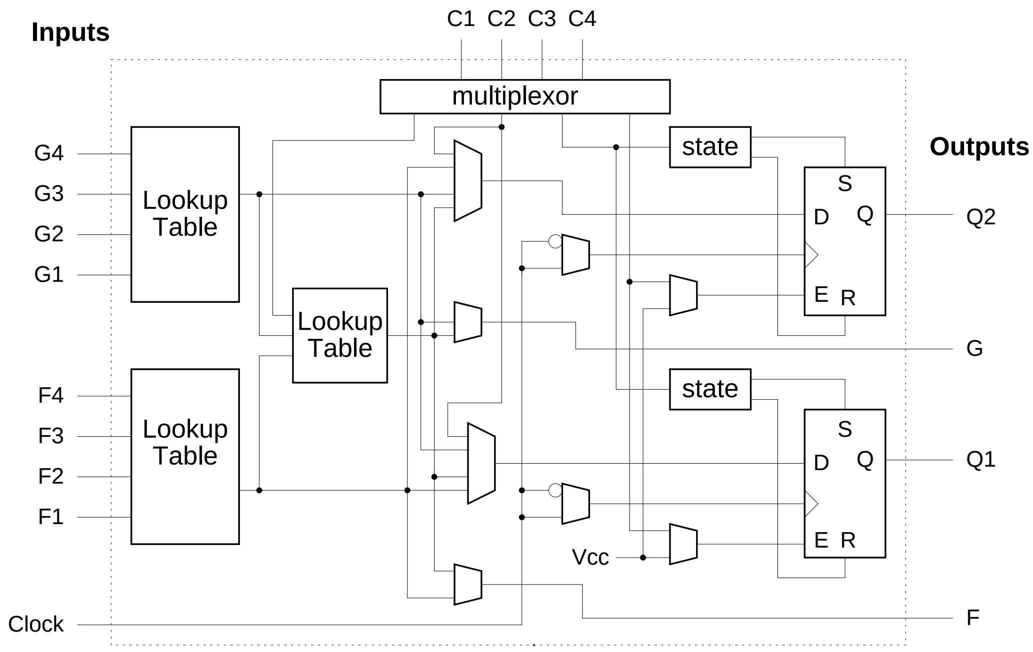

Figure 2.10 - *XC4000 CLB*

The XC4000 routing architecture is significantly different from the earlier Xilinx FPGAs, with the most obvious difference being the replacement of the Direct interconnect and General Purpose interconnect with two new resources, called Single-length Lines and Double-length Lines. The Single-length Lines, which are intended for relatively short connections or those that do not have critical timing requirements, are shown in Figure 2.11, where each X indicates a routing switch. This figure illustrates three architectural enhancements in the XC4000 series:

1. There are more wiring segments in the XC4000. While the number shown in the figure is only suggestive, the XC4000 contains more than twice as many wiring segments as does the XC3000.

2. Most CLB pins can connect to a high percentage of the wiring segments. This represents an increase in connectivity over the XC3000.

3. Each wiring segment that enters a switch matrix can connect to only three others, which is half the number found in the XC3000.

It is interesting to note these three enhancements here because they are all supported by the architectural research that appears in Chapter 4 of this thesis.

The remaining routing resources in the XC4000, which includes the Double-length Lines and the Long Lines, are shown in Figure 2.12. As the figure shows, the Double-length Lines are similar to the Single-length Lines, except that each one passes through half as many switch matrices. This scheme offers lower routing delays for moderately long connections that are not appropriate for the low-skew Long Lines. For clarity, neither the Single-length Lines nor the routing switches that connect to the CLB pins are shown in Figure 2.12.

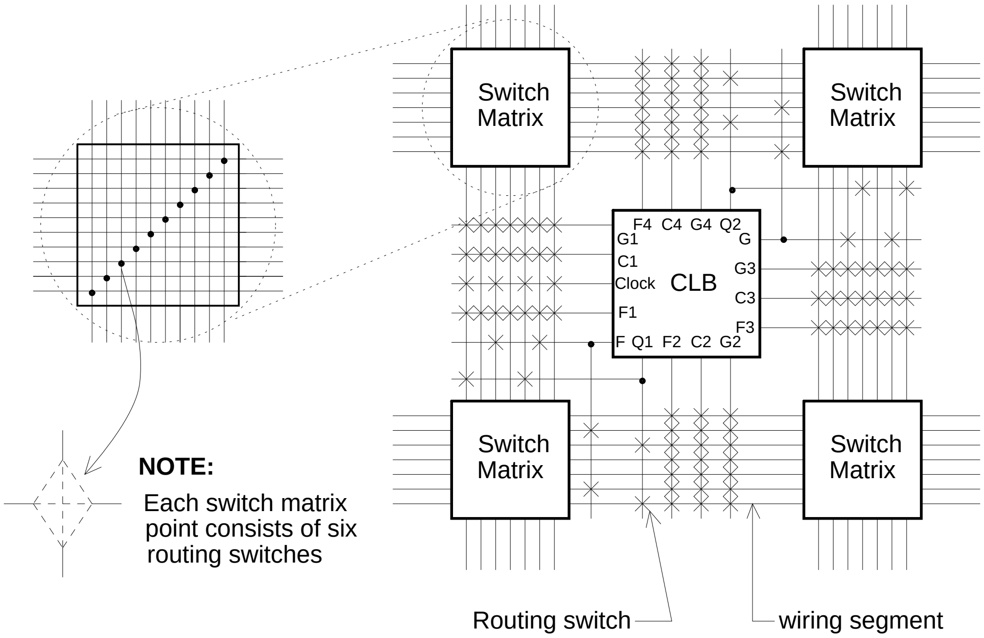

Figure 2.11 - *XC4000 Single-Length Lines*

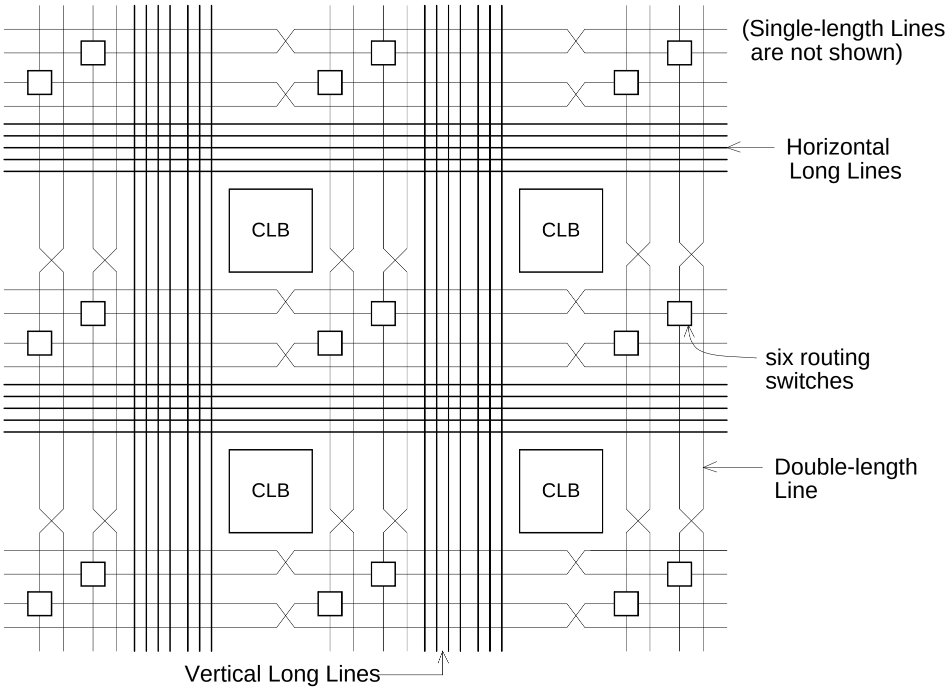

Figure 2.12 - *XC4000 Double-Length Lines and Long Lines*

2.3.1.4 Xilinx CAD Routing Tools

Xilinx routing tools are based on maze routers that are customized for the particular routing resources in each part. It was noted earlier in this chapter that maze routers are unable to consider the side effects that routing some connection in a particular fashion may have on other connections. This is a serious shortcoming because Xilinx routing structures have limited connectivity, and for this reason maze routing is probably not the best technique to use for Xilinx devices.

2.3.2 Actel FPGAs

The basic architecture of Actel FPGAs, depicted in Figure 2.13, is similar to that found in MPGAs, consisting of rows of programmable cells, called Logic Modules (LMs), with horizontal routing channels between the rows. Each routing switch in these FPGAs is implemented by a novel device called an anti-fuse [ElAy88], which normally resides in a high-impedance state but takes on a low resistance (about 500 ohms) when "programmed" by a high voltage pulse. Actel currently has two generations of FPGAs, called the Act-1 [ElAy88] and Act-2 [Ahre90], whose logic capacities are shown in Table 2.2.

2.3.2.1 Actel Act-1

The Act-1 LM that is shown in Figure 2.14 illustrates a very different approach from that found in Xilinx FPGAs. Namely, while Xilinx utilizes a large, complex CLB, Actel advocates a small, simple LM. Research has shown [Sing91] that both of these approaches have their merits, and the best choice for a programmable cell depends on the

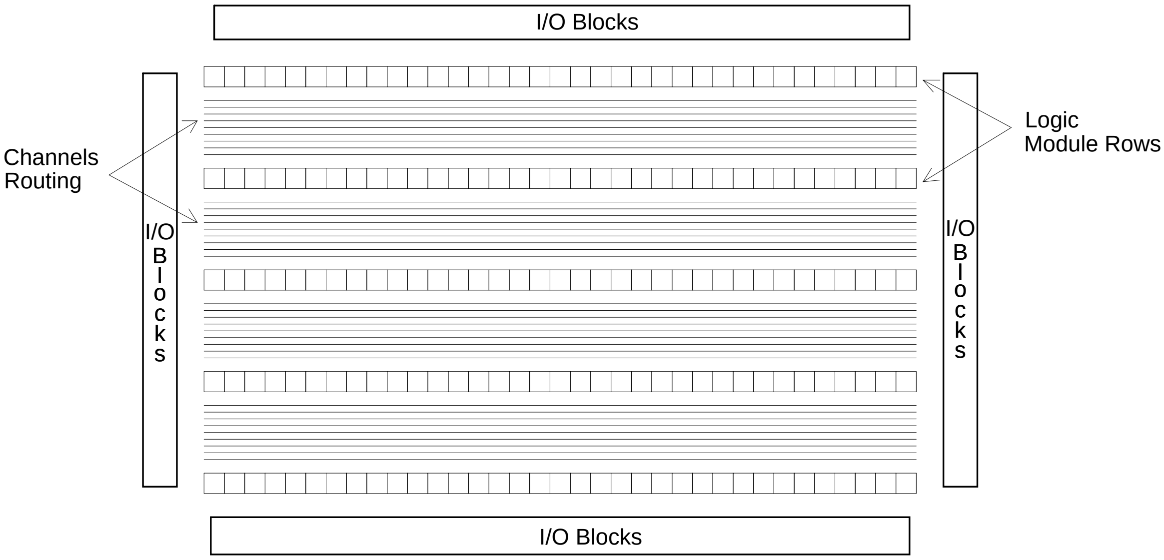

Figure 2.13 - *General Architecture of Actel FPGAs*

Series	Number of LMs	Equivalent Gates
Act-1	295 - 546	1200 - 2000
Act-2	430 - 1232	6250 - 20000

Table 2.2 - *Actel FPGA Logic Capacities*

speed performance of the routing architecture. As Figure 2.14 shows, the Act-1 LM is based on a configuration of multiplexers, which can implement any function of two variables, most functions of three, some of four, up to a total of 702 logic functions [Mail90].

The Act-1 routing architecture is illustrated in Figure 2.15, which for clarity shows only the routing resources connected to the LM in the middle of the picture. The Act-1 employs four distinct types of routing resources: Input segments, Output segments, Clock tracks, and Wiring segments. Input segments connect four of the LM inputs to the Wiring segments above the LM and four to those below, while an Output segment connects the LM output to several channels, both above and below the module. The Wiring segments consist of straight metal lines of various lengths that can be connected together through anti-fuses to form longer lines. The Act-1 features 22 tracks of Wiring segments in each routing channel and, although not shown in the figure, 13 vertical tracks that lie directly on top of each LM column. Clock tracks are special low-delay lines that are used for signals that must reach many LMs with minimum skew.

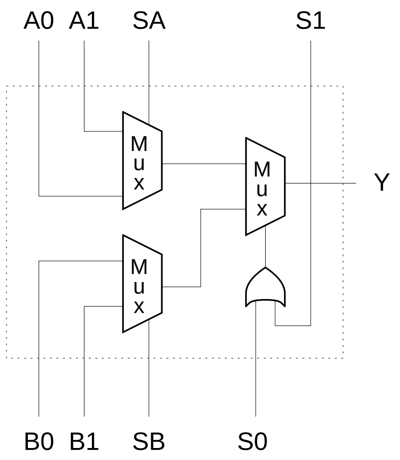

Figure 2.14 - *Act-1 LM*

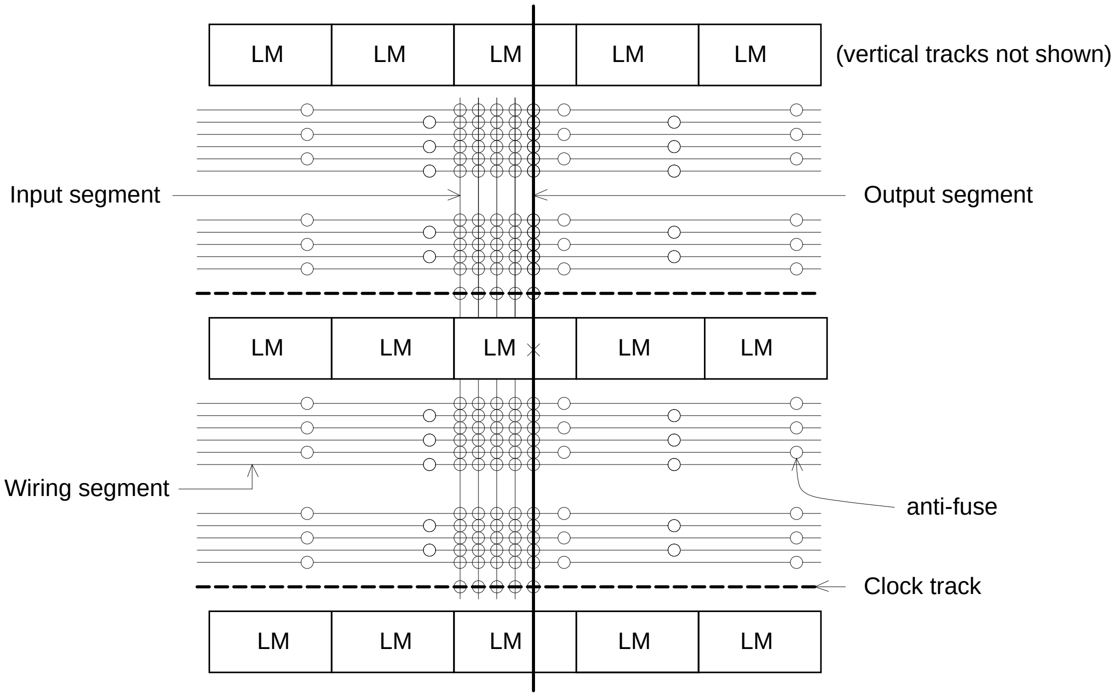

Figure 2.15 - *Act-1 Programmable Interconnect Architecture*

2.3.2.2 Actel Act-2

The Act-2 device, an enhanced version of the Act-1, contains two different programmable cells, called the C (Combinational) module and the S (Sequential) module. The C module is very similar to the Act-1 LM, although slightly more complex, while the S module is optimized to implement sequential elements.

The Act-2 routing architecture is also similar to that found in the Act-1. It features the same four types of routing resources, but the number of tracks is boosted to 36 in each routing channel and 15 in each column.

2.3.2.3 Actel CAD Routing Tools

The key CAD tool that is used to route Actel FPGAs is the segmented channel router described in [Green90]. This router uses a novel algorithm that guarantees that every connection will pass through at most a given maximum number of anti-fuses, if such a solution exists, and in this sense the algorithm produces an optimal result. Although channel routers are not generally appropriate for FPGAs, for reasons given in Chapter 3, it is possible to use this technique for Actel designs because of their high connectivity. Every LM input connects to all of the tracks either above or below it and each LM output connects to all the tracks in the channels spanned by its output segment. However, it is worthy of note that the research reported in Chapter 4 of this thesis indicates that this connectivity can be reduced, in which case it might be necessary to modify the routing algorithm to handle the reduced horizontal-vertical connectivity.

2.3.3 Altera FPGAs

Altera FPGAs [Alt90] are considerably different from the others discussed above because they resemble large Programmable Logic Devices. Nonetheless, they are functionally equivalent to FPGAs because they employ a two-dimensional array of programmable cells and a programmable routing structure, they can implement multi-level logic, and they are user-programmable. Altera's general architecture, which is based on an EPROM programming technology, is illustrated in Figure 2.16. It consists of an array of programmable cells, called Logic Array Blocks (LABs), interconnected by a routing resource called the Programmable Interconnect Array (PIA). The logic capacities of the two generations of Altera FPGAs are listed in Table 2.3.

The Altera LAB is by far the most complex logic cell of any of the FPGA families described thus far. A LAB can be thought of as an efficient PLD, as will be explained in the following paragraphs. Each LAB, as seen in Figure 2.17, consists of two major blocks, called the Macrocell Array and the Expander Product Terms.

The Macrocell Array is a one-dimensional array of elements called Macrocells, where the number of elements in the array varies with each Altera device. As illustrated

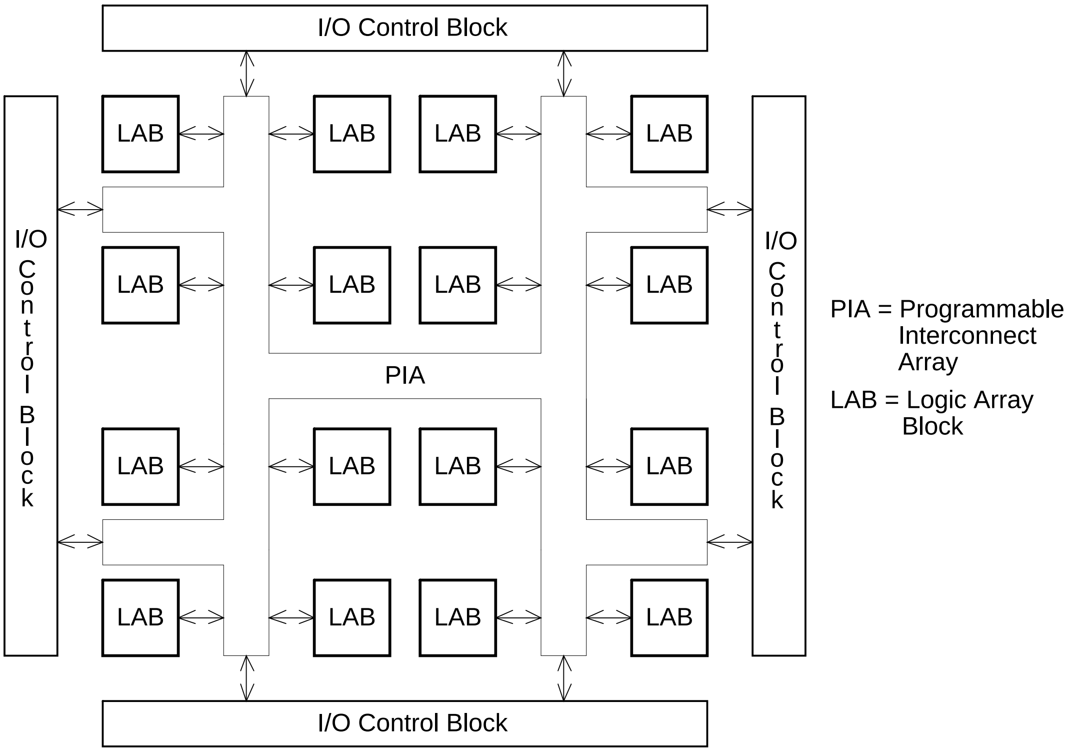

Figure 2.16 - *General Architecture of Altera FPGAs*

in Figure 2.18, each Macrocell comprises three wide AND gates that feed an OR gate which connects to an XOR gate, and a flip-flop. The XOR gate generates the Macrocell output and can optionally be registered. In Figure 2.18, the inputs to the Macrocell are shown as single-input AND gates because each is generated as a wired-AND (called a p-term) of the signals drawn on the left-hand side of the figure. A p-term can include any signal in the PIA, any of the LAB Expander Product Terms (described below), or the output of any other Macrocell. With this arrangement the Macrocell Array functions much like a PLD, but with fewer product terms per register (there are usually at least eight product terms per register in a PLD). Altera claims [Alt90] that this makes the LAB more efficient because most logic functions do not require the large number of p-terms found in PLDs and the LAB supports wide functions by way of the Expander Product Terms.

Series	**Number of LABs**	**Equivalent Gates**
EPM5000	1 - 12	2000 - 7500
EPM7000	N/A	2000 - 20000

Table 2.3 - *Altera FPGA Logic Capacities*

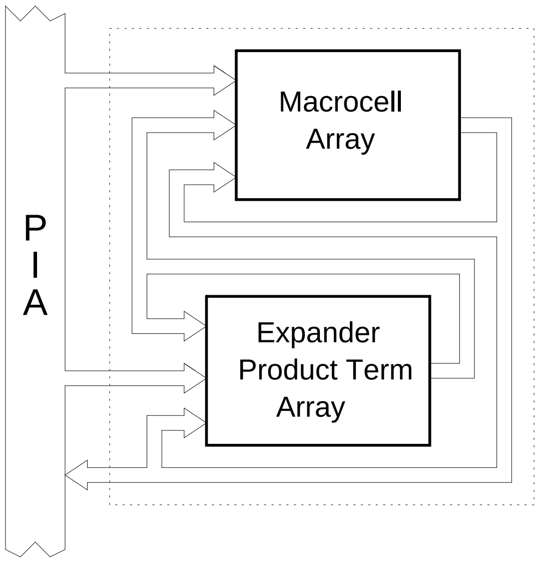

Figure 2.17 - *Altera LAB*

As illustrated in Figure 2.19, each Expander Product Terms block consists of a number of p-terms (the number shown in the figure is only suggestive) that are inverted and fed back to the Macrocell Array, and to itself. This arrangement permits the implementation of very wide logic functions because any Macrocell has access to these extra p-terms.

The Altera routing structure, the PIA, consists of a number of long wiring segments that pass adjacent to every LAB. The PIA provides complete connectivity because each LAB input can be programmably connected to the output of any LAB, without constraints. With this arrangement, routing an Altera FPGA is trivial, since there are no routing constraints. However, as mentioned previously for Actel FPGAs, this level of connectivity is excessive and could probably be reduced, given an appropriate routing algorithm.

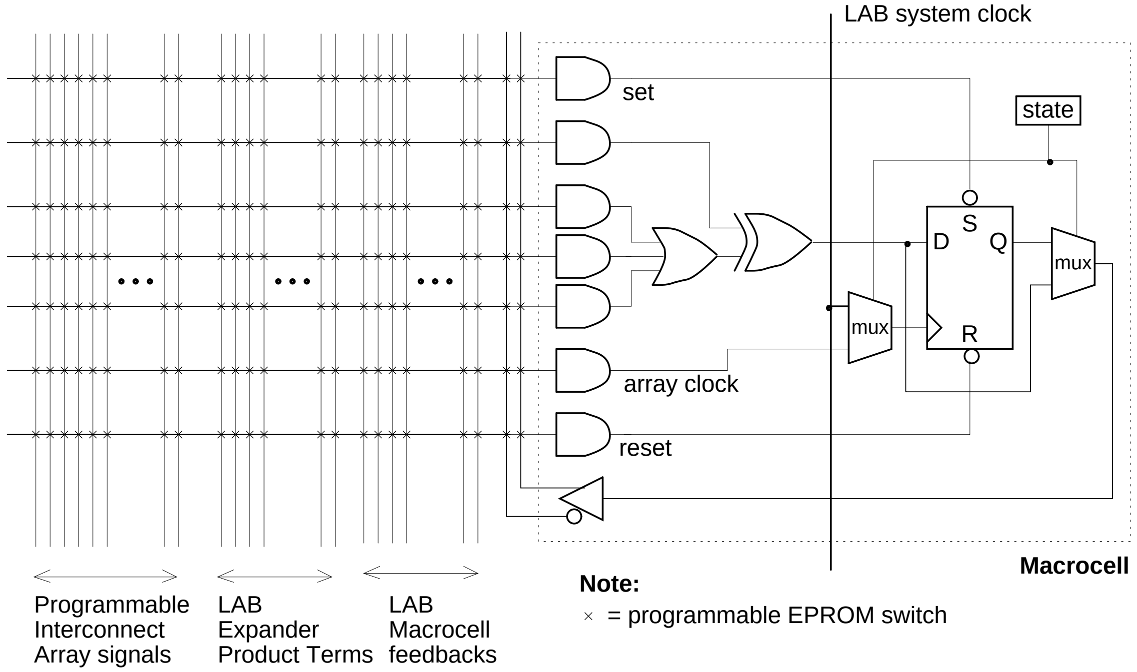

Figure 2.18 - *Altera Macrocell*

2.3.4 Other FPGAs

Four recently introduced FPGAs are described briefly in this section, including those from Plessey Co., Plus Logic, Advanced Micro Devices, and Quicklogic.

2.3.4.1 Plessey FPGAs

The Plessey FPGA, described in [Ples89], is called an Electrically Reconfigurable Array. It consists of a two-dimensional array of logic cells overlayed with a dense interconnect resource. With the routing resources placed on top of the logic cells, these devices resemble the Sea-Of-Gates architecture used in some MPGAs. Each Plessey logic cell is relatively simple, containing an eight-to-two line multiplexer that feeds a NAND gate, and a transparent latch. The multiplexer is controlled by a Static RAM block and is

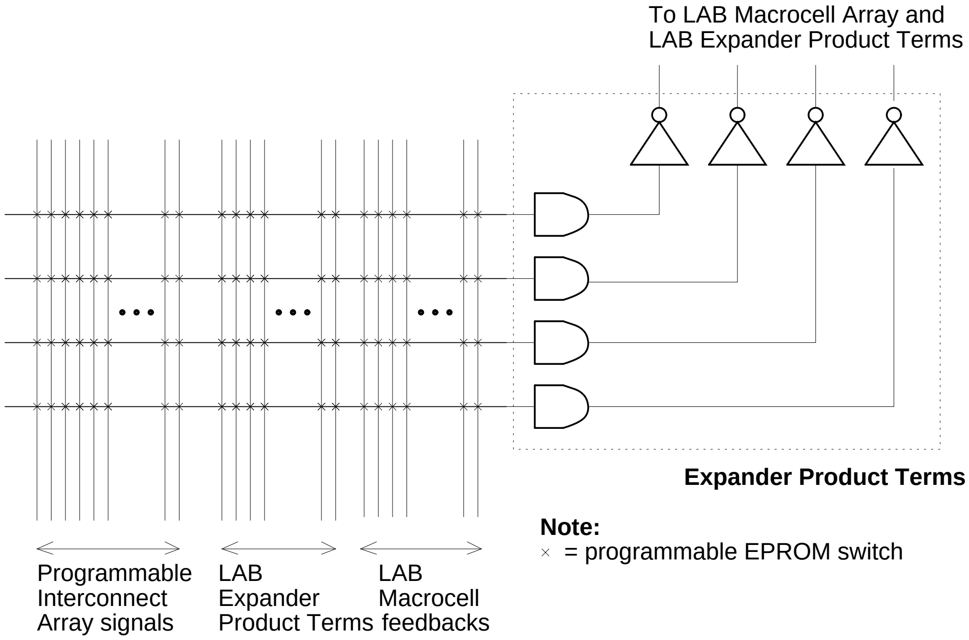

Figure 2.19 - *Altera Expander Product Terms*

used to connect the logic cell to the routing resources, which comprise wiring segments of various lengths: Local interconnect for short connections, Short Range interconnect for moderate-length connections, and Long Range interconnect for long connections.

2.3.4.2 Plus Logic FPGAs

The Plus Logic FPGA [Plus90] consists of two columns of four logic cells, called Functional Blocks (FBs), that can be fully interconnected by a Universal Interconnect Matrix (a full cross-bar switch). Compared to the three FPGA architectures that were described in detail at the first of this section, this device is most like an Altera FPGA, but the FBs represent more complex logic cells. Each FB comprises a wide AND plane that feeds an OR plane, like a PLA (programmable AND/programmable OR) device. The OR plane feed a third plane, which generates the nine (optionally registered) outputs of an FB. Each of these outputs corresponds to any function of two terms from the OR array and one output of any other FB. The programming technology used by Plus is EPROM.

2.3.4.3 Advanced Micro Devices (AMD) FPGAs

The AMD FPGA [AMD90], based on EEPROM technology, can be considered to be an array of PAL (programmable AND/fixed OR) devices that are interconnected by a switch matrix, much like an Altera FPGA. Each PAL block consists of an AND plane that feeds a large block, called a Logic Allocator (LA). The LA allocates a variable number of product terms (p-terms) from the AND plane to individual Macrocells, where each Macrocell provides an OR function of its p-terms, optionally registered. The Macrocell outputs are fed back to the other PAL blocks via the switch matrix.

2.3.4.4 Quicklogic FPGAs

The Quicklogic FPGA [Quick91] consists of a two-dimensional array of like cells called pASIC Logic Cells (pLCs). Each pLC comprises four two-input AND gates feeding two two-input multiplexers, which feed a third multiplexer. The two first-stage multiplexer's select lines are driven by a single six-input AND gate, and the second-stage multiplexer's select line is driven by another six-input AND gate. The second-stage multiplexer provides the cell output, which can be optionally registered by a D flip-flop. The pLCs are interconnected by horizontal and vertical routing channels that provide full connectivity - every horizontal routing track can be connected to every vertical track and every track that passes a logic cell can connect to all the pins on that cell. Programmed connections are formed in Quicklogic FPGAs using a unique anti-fuse. Compared to the Actel anti-fuse, these devices boast an extremely low on-resistance (50 ohms) and unprogrammed capacitance (one femtofarad). Compared with other FPGAs, the Quicklogic devices are most like those from Actel.