Architectures and Technologies for FPGAs

Introduction

The FPGA (Field Programmable Gate Array) is the newest concept in programmable logic. Previously the most complex programmable logic device was the Complex Programmable Logic Device, the CPLD. The CPLD concept is a simple extension of the basic PLD. Taking a small PLD device design and repeating it multiple times on the same die provides large resources in a single device. It is then necessary is to provide interconnect resources to allow each repeated cell to share resources and to communicate with one another and the I/O cells. The individual repeated cells are called macrocells which are, of course, relatively large and functionally complex.

Before the FPGA, the next level up from the PLD in solution alternatives was the sea-of-gates gate array. This is a fixed die, consisting of transistors, that is customized by the user by specifying the interconnect of the transistors. The user, in actuality, specifies the interconnection between a set of functional primitives such as NAND gates and flip-flops, which the gate array vendor has predefined and placed in a library. The gate array is not user programmable and must be customized by the vendor in the manufacturing process. Delivery of first articles is many weeks, and non-recurring costs are usually above ten thousand dollars.

Between the CPLD and the gate array is the FPGA which borrows from the solutions above and below. The FPGA logic cells are small and have less functionality than those of the CPLD. Thus they are a move toward the sea-of-gates concept in the Gate Array ASIC. Since the FPGA logic cells are smaller than those of the CPLD, there are many more of them in the same die. The FPGA logic cells are arranged in a rectangular array as in the gate array sea-of-gates concept. Between each logic cell is a routing channel so that multiple interconnect wires can run vertically and horizontally across the chip. Programmable connection points are provided where the logic cell I/O enters the routing channel and at the cross points where vertical routing channels meet horizontal routing channels. By appropriate programming of the cell I/O connections and the cross channel connections, signals can be routed throughout the chip.

Although the FPGA concept is relatively simple, realization of the FPGA is complex. There are many interrelated technology and architecture issues which must be addressed to produce a successful device. Success in this context means a device which can make maximum use of the available resources to accommodate large designs, achieve the highest possible performance that the semiconductor process technology has to offer, and give the designer flexibility (in, for example, pin assignments). This application note is intended to explain key factors in technology and architecture issues and how they relate. From this understanding, benefits to the designer will emerge. Different FPGA approaches have very different characteristics that can make the difference between a design achieving the required performance or being able to fit into a specific device. The material in this note is intended to help the design engineer make choices that will help achieve design goals.

Detailed Architecture

The global form of an FPGA is shown in *Figure 1*. The layout is a matrix of logic cells with a grid of routing channels running between the cells. I/O cells surround the array and allow access to the ex-

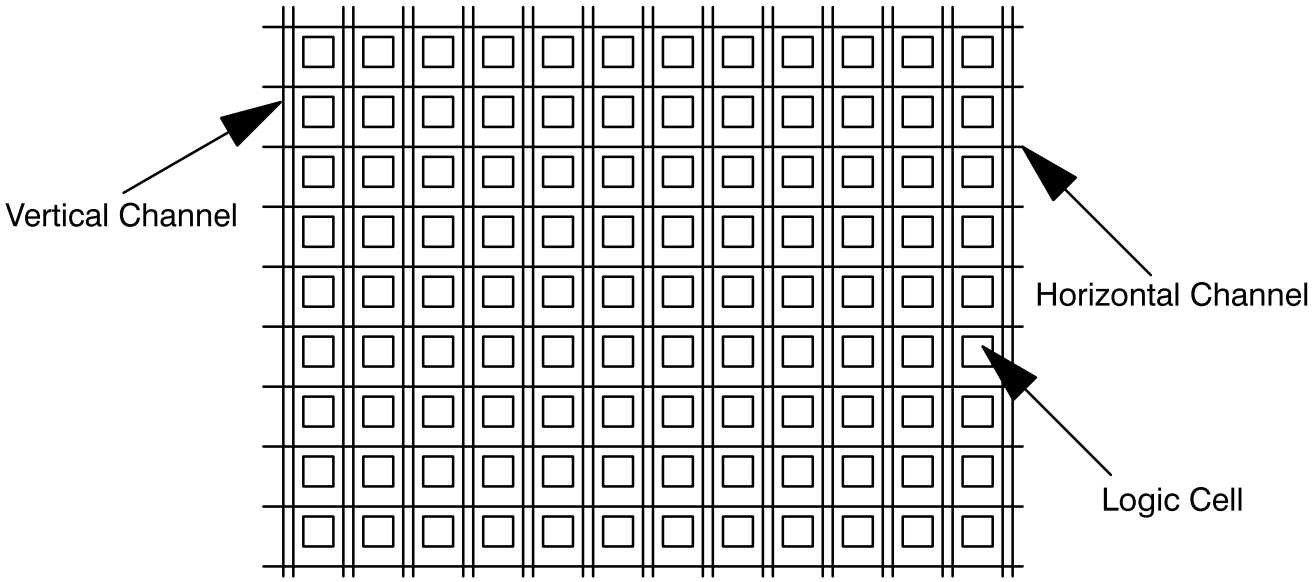

Figure 1. Global FPGA Architecture

ternal pins of the device. The programmable connections are located where the vertical and horizontal routing channels cross; where the I/O of the logic cells meet the routing channels; and where the I/O cells around the periphery meet the routing channels. In contrast to the CPLD, the FPGA usually has no programmability within the logic cells themselves (Some FPGA architectures do include programmable elements within the logic cell. Beyond this fundamental architecture, FPGAs can differ widely in the details. Key considerations are: (1) the number of wires in the routing channels, (2) the flexibility in the interconnect programmability where channels meet channels and where logic cells meet channels, and (3) the functionality contained within the logic cell. The details of the architecture choices in these key considerations are not obvious and are closely tied to the semiconductor process technology that is used to realize the device. The remainder of this section focuses on some of the more important aspects of these details: to show how they influence architecture choices and what impact the choices have on the performance, cost, and utility of the final product.

FPGA Logic Cells

There are three approaches to the form of a logic cell in FPGA implementations.

One approach is to make the logic cell as complex as in the CPLD. Such an implementation is termed a Coarse-Grain Logic Cell. An example is shown in *Figure 2*. This cell contains multiple flip-flops, several multiplexers, a combinatorial function block, and a variety of different inputs. Each of the flip-flops may be bypassed in order to implement combinatorial functions. From a first level analysis, it is clear that this logic cell can implement complex logic as well as register intensive functions. Three important characteristics are significant to note. First, the cell is complex and, as will be explained later, unless there is programmability in the cell, this can lead to inefficient use of the available logic. Second, there are a small number of cell outputs (two in this case). Third, many of the inputs are dedicated to flip-flop control and are not usable for other purposes if the flip-flop is not required. Timing analysis of this cell can be complicated. Since the cell possesses a large amount of functionality, it reduces the burden on the cell-to-cell interconnect.

The second approach in the cell architecture is to make the logic cell very simple. This approach is termed the Fine-Grain Logic Cell. An example is shown in *Figure 3*. This cell contains no flip-flops and very minimal logic. Only one output is available. The cell can realize simple AND-OR logic implementations and, because of its simplicity, it can

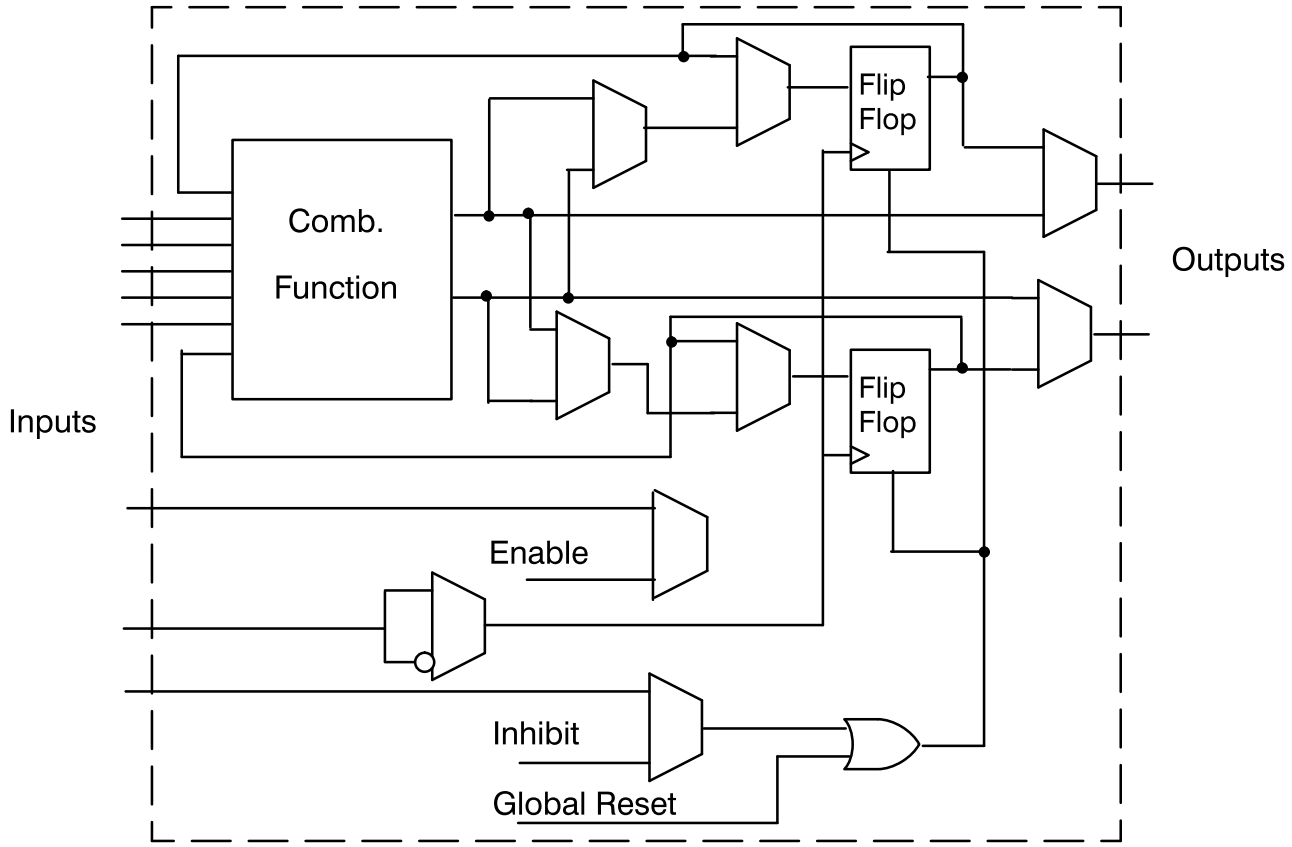

Figure 2. Coarse-Grain Logic Cell

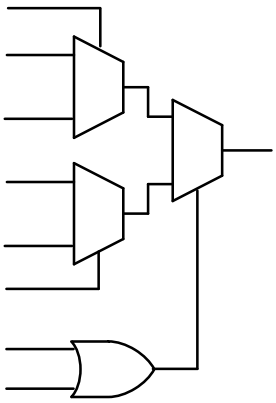

Figure 3. Fine-Grain Logic Cell

have very low propagation delays through the cell and there tends to be a high cell utilization. This cell will rely heavily on the cell-to-cell routing resources. The greater use of routing may add to the overall delay negating the low delay of the cell.

The third approach is a hybrid between the coarse-grain and fine-grain extremes but with some variations that enhance the other trade offs in the global architecture design. To understand the hybrid role, the relationship of the fine-grain and coarse-grain logic cell approaches to the global architecture design must be understood.

The global architecture relation to the logic cell type can be illustrated by starting from the basic FPGA concept and identifying and working through the implementation trade offs. Consider a sea of logic cells surrounded by the routing channels. Assume that the routing channels are very large and the interconnect completely flexible. Then, as in the custom sea-of-gates ASIC, the logic cells can be small, consisting of elementary logic primitives. The first trade off is that the FPGA is not a gate array where gates are sacrificed for routing. In the FPGA, the routing resources are limited and unused gates offer no increase in interconnect capability. This suggests

that the logic cell possess more than minimal functionality and should include at least one flip-flop and wide AND-OR combinations to realize complex logic. It also suggests that the logic cell have multiple outputs so that if some logic functions are very simple, more than one of these functions could be implemented in the same cell. This allows maximum utilization of the logic cell resources. Continuing this pattern would further increase functionality in the logic cell. Bear in mind, however, that the logic cell lacks programmability in the examples shown here and that directing signals within the logic cell is done with multiplexing and judiciously selecting input combinations. Further expansion of this would waste resources in the signal directing logic and increase logic cell propagation delay. This is detrimental to the objective of implementing the desired logic function. Added circuitry and controls in the logic cell to maximize its flexibility do not contribute to the realization of the desired function. That flexibility is the task of the interconnect. There is, therefore, an optimum logic cell complexity which lies somewhere between the coarse- and fine-grained extremes. With finite routing resources available, but without fuse related constraints, the optimal cell would look something like the cell in *Figure 4*. This cell design is simple and symmetric yielding small propagation delays regardless of the signal path through the cell. Since the interconnect is not a factor, a large number of inputs is provided so that logic functions of many variables can be implemented. The cell also provides for the realization of any logic function up to a given number of variables. Multiple outputs are also provided. The flip-flop may be used or the "D" input of the flip-flop is available as an output for combinatorial only functions. The large number of outputs allows the cell to be split, implementing two or more simple logic functions in the same cell or sharing logic across function in the same cell.

Programmable Connections

Now examine the interconnect surrounding the sea of logic cells. There are two key issues in the implementation of the programmable connections. First is size. The programmable interconnect circuit may

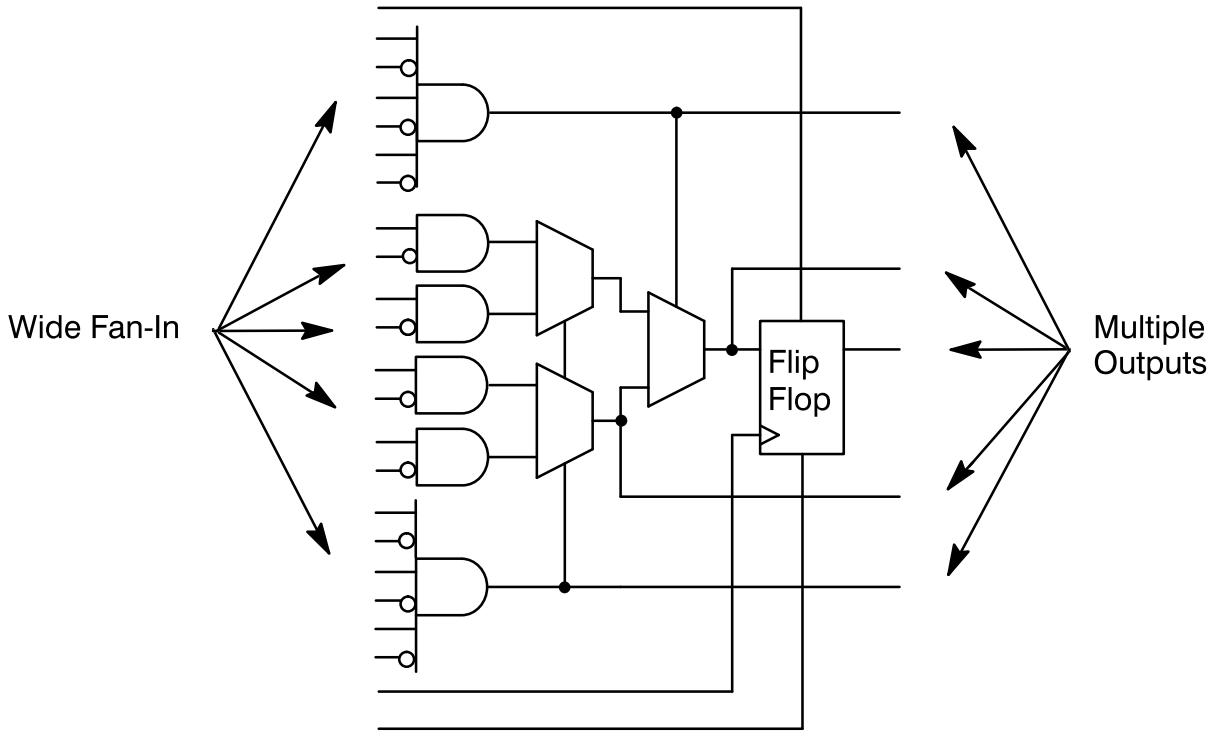

Figure 4. An Optimal Logic Cell

use an area which limits the number of interconnects which can be put into a given area. Second is the electrical characteristics of the interconnect. The interconnect may not look like a wire to the signal that it is carrying. Resistance and capacitance of the interconnect can affect the propagation delay of the signal.

Infinite routability is not realistic. The routing channels can contain only a finite number of wires and the interconnect possibilities where wires cross or meet is not endless. There are three classes of ways to connect two wires: RAM-based connections, large fuse technology, and via fuse technology. Consider the rectangular area where horizontal and vertical wiring channels cross. At this intersection there is potentially a connection possibility at each wire crossing (intersection). An ideal case of the user programmable interconnect possibilities are shown in *Figure 5*. Each circle at the intersection point represents a potential user-programmable connection. This scheme offers a great deal of interconnect capability. There is a lot of redundancy in the interconnect possibilities. Once a wire is connected to a signal, it is dedicated to that signal throughout the extent of that wire. Wires may be segmented so that they can support local interconnect. The implementation of the connection mechanism (fuse, RAM cell, etc.) may be larger than the dimension of the wire size and inter-wire spacing as shown in *Figure 6*. Because of the size of the connection cell for connection "O," programmable connection at points "X" are not possible. Further programmable connection cells cannot fit into the area unless the interconnect wires are spread apart. This latter approach is not an effective alternative since it upsets the regularity and fit of the wiring channels and the logic cells. The only alternative is to limit the number of connection possibilities.

An example of this situation is RAM-based programmable interconnect. RAM-based connectivity uses a memory cell to control the connection of one wire to another. The memory cell is far larger than the wire intersect area. What is done is to limit the connection possibilities to a small fraction of the possible signal paths and limit the number of wires in the routing channel. The result is that the routability of the FPGA becomes what is known as "interconnect constrained." That is, the number of wire connection possibilities is so small that the interconnect limits the realization of functions with a given logic cell architecture. To compensate for this, coarse-grained logic cells are usually chosen and programming internal to the logic cell may be added. The ability to perform more function in a logic cell tends to make up for the inability to implement the equivalent function in a set of interconnected logic cells. It is difficult to quantify the results of such choices. Can a large complex logic cell adequately compensate for interconnect constrained situations? There is no clear answer and the results are dependent on what is to be implemented in the FPGA. However, it can be determined by realizing various types of functions in various architecture forms that the interconnect-constrained architec-

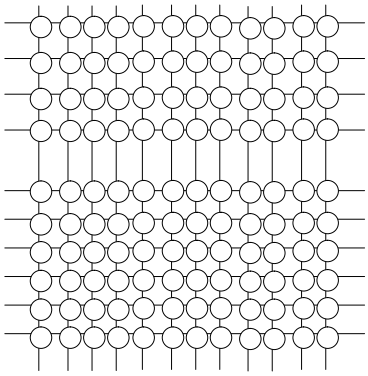

Figure 5. Connection Points at Routing Channel Intersection

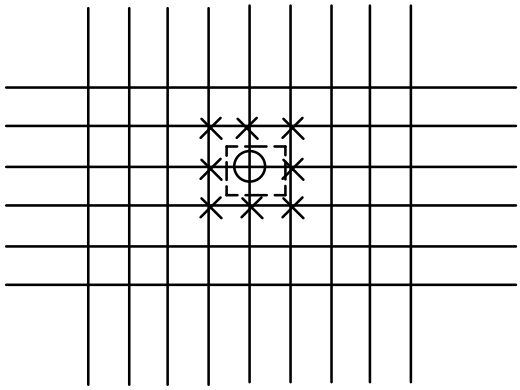

Figure 6. Connection Limitations due to Programmable Connect Cell Size

tures tend to have routability limitations which are exacerbated when any constraints are placed on the device pin/signal association.

Architectures implemented with large-fuse technology tend to have the same characteristics as RAM-based connectivity except they are not as severe. Both approaches limit the number of outputs in the logic cell to only one or two. This is because increased outputs add to the number of potential connections, which places a further burden upon an already stressed interconnect mechanism. Because the number of outputs is small, simple functions will tend to waste logic resources in the interconnect-constrained, complex logic cell architectures. Therefore, these architectures may tend to be more efficient in implementing complex state machines than the fine-grained logic cell architectures with the same interconnect capability.

Of the programmable interconnect technologies available for FPGAs, (RAM-based, large-fuse, and via-fuse) the optimum interconnect is achieved by via-fuse technology. This technology realizes an antifuse in the same physical area as that used by a normal semiconductor process via which connects two layers of metal. The via fuse will be described in detail in a later section. This technology approaches the interconnect characteristics found in sea-of-gates gate arrays and almost completely eliminates programmable interconnect as a factor in FPGA wiring channel and logic cell architectures.

The electrical characteristics of the interconnect technology play a major role in the performance of the FPGA. A first level summary of the technology impact on the interconnect (not including wire delay) is given in *Table 1*. When the connection is made, it exhibits some ON resistance in series with the logic signal path. When a connection is OFF it presents a shunt capacitance from the logic signal wire to ground. A single ON fuse followed by a single OFF fuse represents an RC combination in the signal path. The product of the ON resistance and the OFF capacitance of this combination is given in the Time Constant column of the table.

Timing Model

The timing model is a representation of signal delays in an actual FPGA that allow the designer to determine the performance of a design when it is realized in a particular device. The nature of the timing model is of concern to the designer since it affects the level of difficulty in determining performance. All FPGAs, by virtue of their architectures, inherently have variable timing models. The pin-to-pin propagation delay depends upon the number of logic cells cascaded together to achieve a given logic function. There are two types of variable timing models: simple and fine structured. In the simple variable timing model, the pin-to-pin propagation delay is chiefly dependent upon the number of cascaded logic cells, signal fan out, and the wire delay that would normally be encountered in a sea-of-gates gate array. The number of interconnect points in the signal path and the logic function implemented in the logic cell tend to have secondary and lesser effects on the timing. Architectures suitable for the simple timing model will have actual device delay characteristics which are independent of where the logic cell is placed in the array. In the fine structured variable timing model, pin-to-pin propagation delay is strongly dependent upon not only the simple model factors but also the number of programmable interconnects in the signal path and the function implemented in the logic cell. In these cases, the logic cell itself has a variable timing model due to its complex structure and antisymmetry. Programmable interconnect points with unfavorable electrical characteristics raise the effect of the number of interconnect points in a signal path to being a first order effect.

Table 1. Electrical Characteristics of Fuse Technologies

Technology	ON Resistance	OFF Cap.	T	Tgate
Via Fuse	50 ohms	1 fF	0.05 ps	50 ps
Large Fuse	400 ohms	5 fF	2 ps	400 ps
RAM Cell	800 ohms	10 fF	8 ps	800 ps

The actual performance of devices does not differ by these orders of magnitude. This is because the OFF capacitance of a large number of no-connect fuses is small compared to the metal and gate input capacitances. Therefore the fuse series resistance is the dominant component in limiting performance due to programmable interconnect. To put this factor into perspective, *Table 1* includes a column which is the time constant for one series fuse connected to a gate with a capacitance of 1 pF. Note that with as few as five programmable interconnects in the path from the signal source to one gate, the time constant, for some technologies, can be as large as the gate delay itself.

Interconnect and Logic Cell Trade Off Summary: Advantages and Weaknesses

The technology has a profound effect on the total FPGA architecture. SRAM and large fuse based technologies cause interconnect-constrained architectures which force non-optimal logic cell architectures. In general, interconnect-constrained architected logic cells tend to be large and complex to make up for the interconnect limitations. Moreover, these complex logic cells tend to be wasteful of resources in certain applications. Such architectures are characterized by routing and capacity limitations and an inability to fit a design when there are pinout constraints (user fixes signals to particular pins). Router and fitter software may take many iterations to fit a design.

It is clear that the small-size fuse technology, combined with well chosen routing channel wire complement and an optimum complexity logic cell, will yield a high performance, small die size device. *Figure 7* shows system performance of a fixed benchmark fitted into devices of the three technologies described above. The system performance is plotted versus the semiconductor process technology line width in order to perform an apples to apples

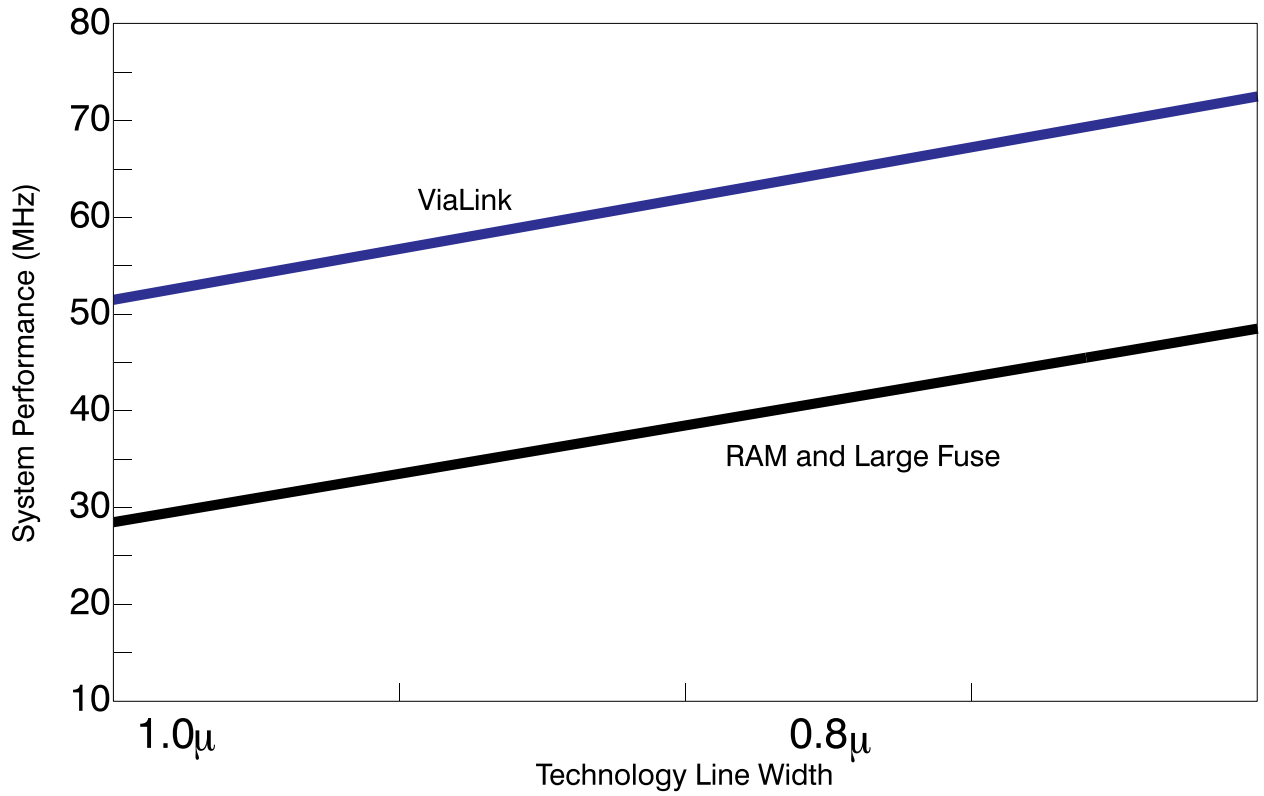

Figure 7. Performance Relative to Connect Technology

comparison of the key factors in the FPGA implementation. As expected, the architectures optimized for ViaLink™ exhibit a significant performance advantage over the large-fuse and RAM-based interconnect approaches.

Comparison to CPLDs

The architecture of the FPGA manifests itself in the device characteristics in much the same way that the sea-of-gates gate array architecture does. The two major influences on the device characteristics are the small logic cell size and the channel routing. First, the cell sizes are small and considerably less complex than those of the CPLD. Therefore, the propagation delay through the FPGA logic cell is much smaller than that through the CPLD macrocell. Functions such as multiplexing, which need only one cell per signal path, will typically achieve much higher performance in the FPGA. In contrast, functions which require cascading of many logic cells to implement may be at a performance disadvantage in the FPGA. Complex state machines with a lot of decoding are in this category. This does not mean that use of the FPGA is to be avoided. In the example to follow, a complex state machine is implemented successfully. Secondly, an abundance of routing resources can permit complicated interconnects as well as convenient handling of buses.

Cypress pASIC380™ Family FPGA Architectures

The previous architecture discussions have pointed out the strong relationship between the technology, the architecture of the FPGA, and the device characteristics. The 380 family possesses a unique technology which impacts all of the remaining architecture trade offs positively. The discussion of the 380 family begins, therefore, with a presentation of the interconnect technology.

pASIC380 Family Fuse Technology

In usual integrated circuits two crossing metal lines that are on different layers may be connected by a via. A via is a small hole in the insulating glass that lies between the two layers of metal. This small hole, which is about the size of the metal lines themselves, is filled with metal from above making the connection to the underlying metal line. The programmable via is a modified via used in standard CMOS semiconductor processing. The modification consists of depositing a thin layer of amorphous silicon in the via hole so that the silicon separates the two layers of metal. As manufactured, this special via has a resistance in excess of 1 gigaohm and an insignificantly small capacitance (about 1 fF). Its size is no larger than the standard via normally used to connect two layers of metal. A cross section of the programmable via is shown in *Figure 8*. A programming pulse applied across the programmable via causes a change in the characteristics of the silicon layer forming a bidirectional conductive link between the top and bottom metal. This programmed link has a series resistance of about 52 ohms and in practice is no more than 65 ohms. The parasitic capacitance is no larger than a normal metal to metal via. The technology is appropriately termed "ViaLink."

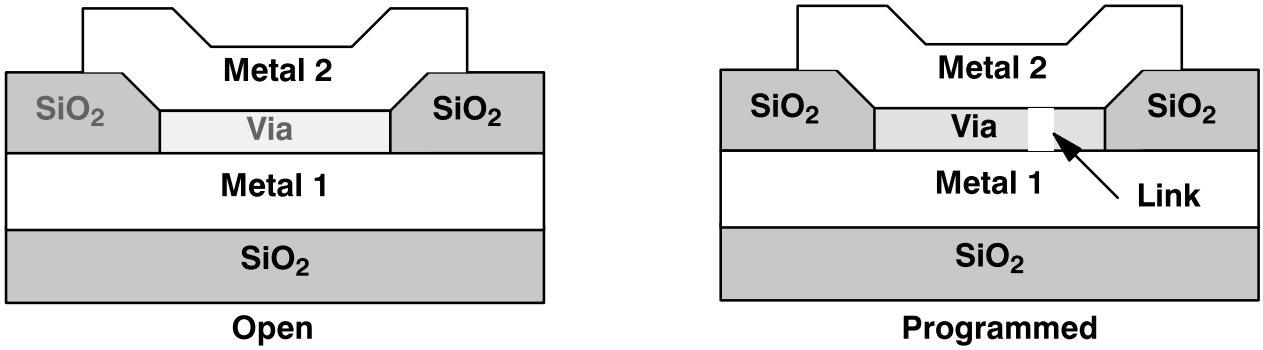

Figure 8. The ViaLink

Routing

ViaLink technology has significant impact on FPGA architecture. Since the programmable site is no larger than the associated metal interconnect wires, there is no real restriction on the number of interconnect points (fuses) and no fuse related restrictions on the number of wires in the interconnect channels. The 380 family routing scheme is architected with this added freedom.

Four types of signal wires are employed in the routing channels:

- segmented wires
- quad segmented wires
- express wires
- clock wires

Segmented wires are wires that extend only from one routing channel to the next, both vertically and horizontally. At the channel junction, a horizontal segmented wire may be programmed to interconnect to a vertical segmented wire at points called cross links. In *Figure 9*, programmable cross links are denoted by the open circle at intersections of vertical and horizontal wires. Also at the channel juncture, the segmented wire may be continued in the original horizontal or vertical direction by connection to another segmented wire running in the same channel. This connection is provided by a pass link. These links are denoted by an "x" in the figure. Segmented wires are most applicable for local wiring around or between adjacent logic cells.

Quad segmented wires are similar to the segmented wires described above except that the wire extends across four logic cells before it is segmented. Like segmented wires, the quad segmented wires may be continued to the next quad segmented wire by a pass link. The quad segmented wires are applicable to signal distribution over a larger but still local group of logic cells.

Express wires are similar to segmented wires except they do not include pass links. An express wire will therefore run the entire length of the device. These wires are most suitable for global signals within the device. Routing software with specific knowledge of the device architecture will automatically route signals over the appropriate wire type.

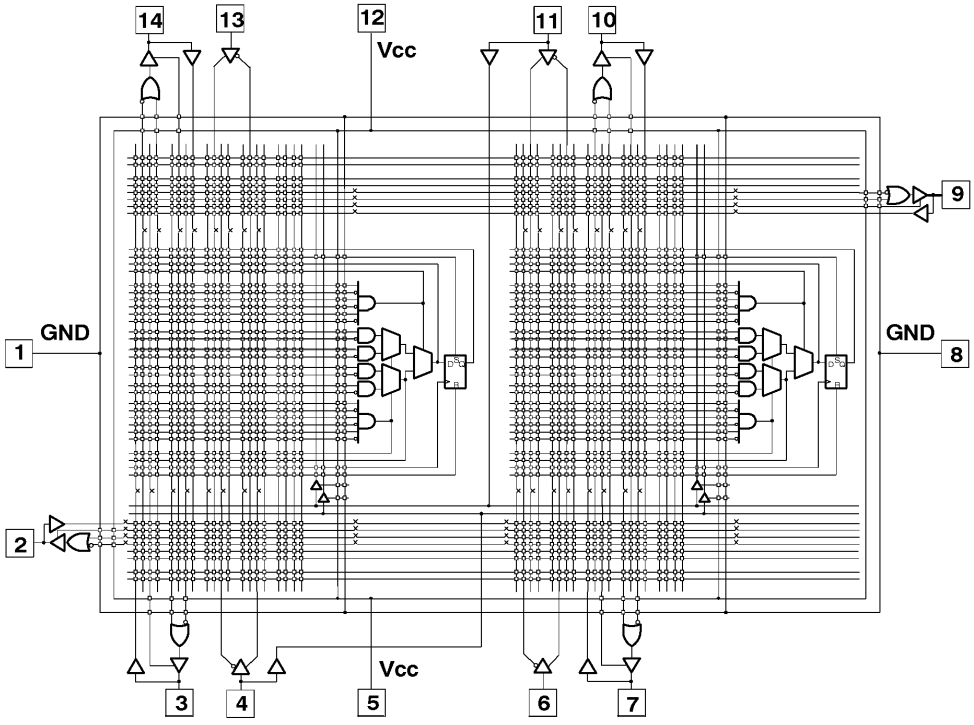

Figure 9. Simplified pASIC380 Family Model

Clock wires are special signal lines that include an array of buffers for minimal skew. Clock wires are similar to express wires except that the cross links are limited. This is to insure that the clock wires are lightly loaded by programmable interconnects and can be used maximally in routing high-speed clocks or reset signals globally throughout the device with minimal skew. The source of the signal on the clock wires is specific device pins with the designation "I/CLK." After passing through the special input buffers, the signal is routed horizontally across the center of the die, as shown in *Figure 10*. There are four high drive buffers. One pair drive clock 1 and clock 2 to the upper half of the column of logic cells, and the other pair drive the two clocks to the lower half column of logic cells. There is a cluster of these buffers for each column of logic cells in the array. The buffers can be enabled to drive the clock lines or disabled if a clock is not required in a given column.

Vertical channels include all three wire types plus V_{CC} and ground wires. The V_{CC} and ground connections allow unused inputs of any logic cell to be tied to an appropriate logic level. The vertical channels run to the left of each logic cell column and extend the full height of the device. The I/O wires, which run from each of the logic cells to the right of the vertical channel, intersect the wires of the vertical channel with cross links at all segmented wires and at judicious points for express wires. At the extreme ends of the vertical channels are I/O cells that connect to the device pins. The number of wires in the vertical channel is chosen to be commensurate with the number of inputs and outputs of a logic cell, the added wires for V_{CC}, ground, and the I/O cells at the device periphery. There are 24 of these wires.

Horizontal channels provide connection by way of cross links from vertical channel to vertical channel

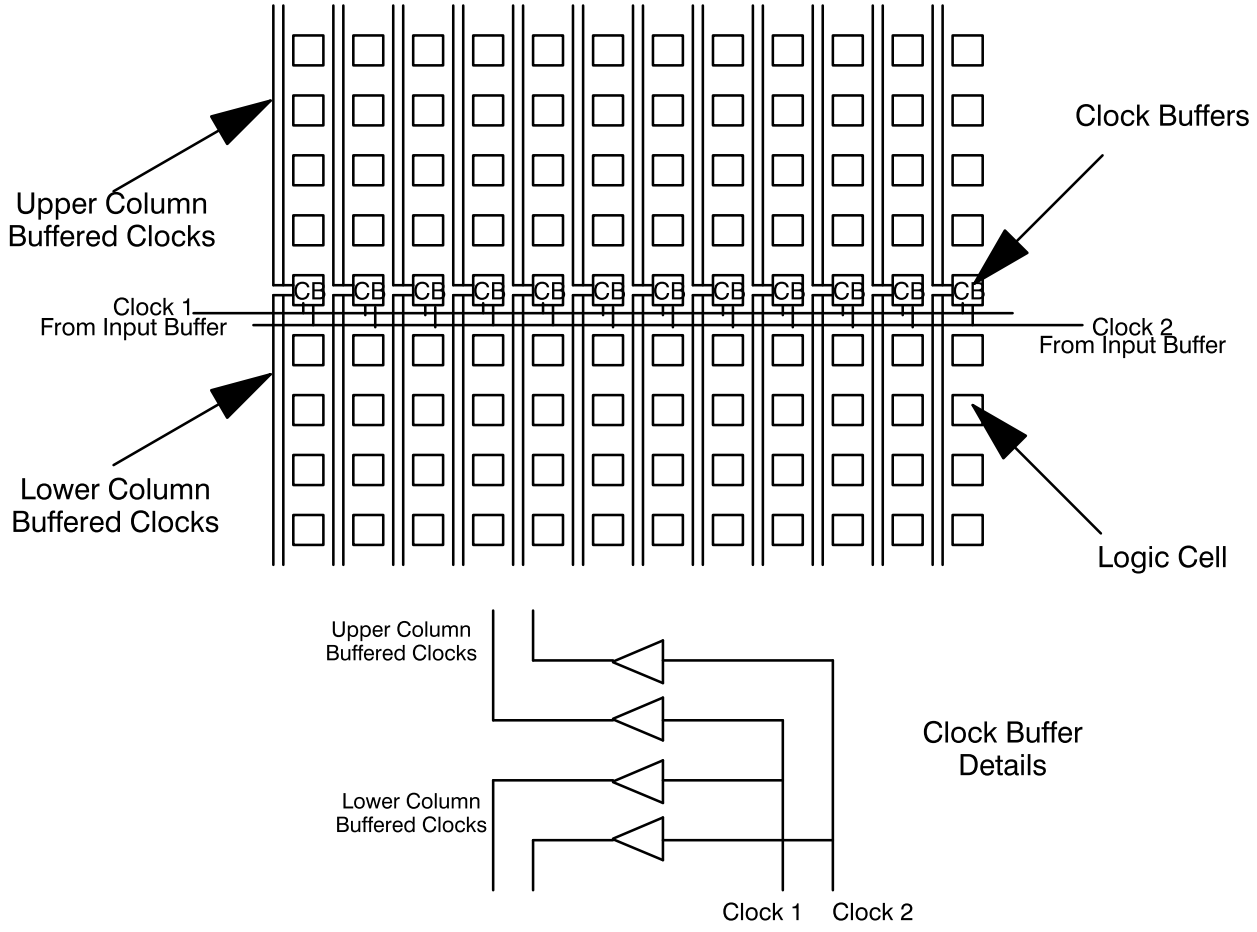

Figure 10. pASIC380 Family Clock Distribution

and from the vertical channels to I/O cells on the left and right periphery of the device. All wire types are included in the horizontal channels (which contain 12 wires each) except for the clock wires. (These are the dedicated wires that carry the clocks to the buffers.)

I/O Cells

There are three types of interface buffers that connect the internal array to the device pins. The dedicated input buffer provides high drive internally and generates both true and complementary versions of the input signal. This high drive capability allows signals coming from these input only buffers to fan out to a larger number of cells than the normal I/O cell. The clock input buffer is similar to the dedicated input buffer except that it provides a third output that is routed to the internal clock distribution buffers described previously. The I/O cell provides a bidirectional connection to the devices pins. The cell can be used as input only, output only, or a bidirectional pin connection. Internally the cell has an output enable, an input data connection, and two output data connections which are ORed together to produce the output. This cell is shown schematically in *Figure 11*. The output driver provides 8 mA drive level (I_{OH} and I_{OL}).

Logic Cells in the 380 Family

Since the routing resources of the 380 family are abundant and without expectation of being interconnect constrained, there is freedom in the logic cell architecture to choose the optimum complexity. The 380 family logic cell is shown in *Figure 12*. This cell has been optimized to maintain the speed advantage of the ViaLink technology while insuring maximum logic flexibility.

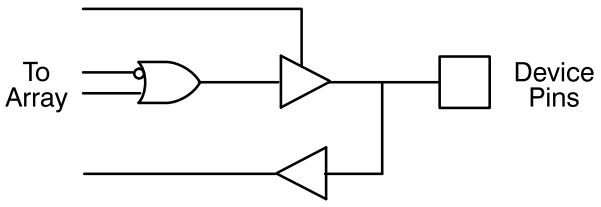

Figure 11. Bidirectional I/O Buffer

The logic cell consists of two 6-input AND gates, four 2-input AND gates, three 2-to-1 multiplexers and a D flip-flop. This cell represents approximately 30 gate equivalents of logic capability. The cell has 23 logic and control inputs and 5 outputs. The arrangement of the gates permits 14-bit-wide gating functions and can realize all possible Boolean transfer functions of up to three variables. The D flip-flop possesses asynchronous set and reset inputs to independently control the output state. The multiplexer and logic feeding the D input allow the flip-flop to be configured as D, T, JK, or SR.

The outputs of the logic cell include the Q output of the flip-flop (QZ) plus four other outputs tapped at selected points within the logic cell. The OZ output is the same as the D input to the flip-flop. The OZ output facilitates combinatorial functions. The three other combinatorial outputs tap the logic cell at selected places. If simple logic functions are to be implemented, the multiple outputs permit more than one of these functions to be realized in a single logic cell. Maximum use of the available logic can be made. Note the ability to provide this multifunction utilization without any significant impact on routing. The additional utilization factor is ob-

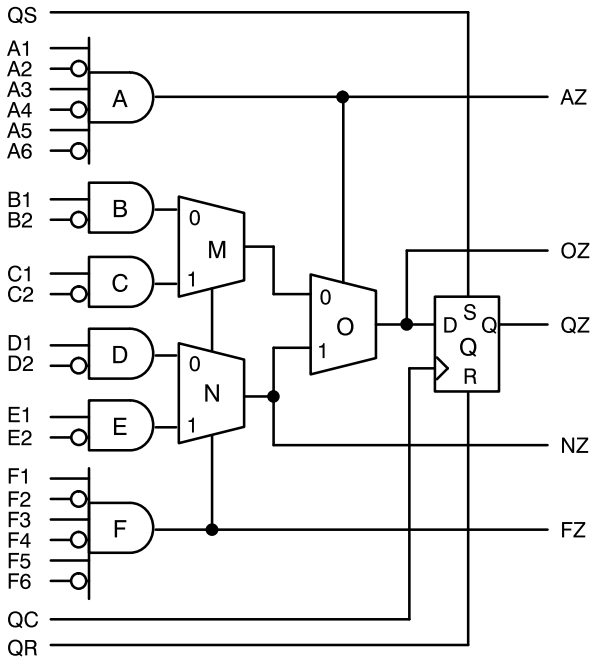

Figure 12. pASIC380 Internal Logic Cell

tained for free. When implementing multiple functions, the flip-flop may still be employed in many cases.

The logic cell is not so complex as to adversely impact propagation delay. The internal multiplexers are positioned to participate in implementing logic functions. Since the multiplexers are all in the path to the D input of the flip-flop, they contribute significantly to combinatorial logic function realization and are not expended on signal steering. The logic cell is also noticeably symmetric and regular. Combinatorial delays are thus also symmetric. That is, input to output delays tend to be roughly the same, although the AZ and FZ output will be faster than the others. Whereas some architectures bypass large sections of cell logic by the multiplexing, thereby making the cell delay dynamically changeable, the 380 logic cell delay is not subject to this condition.

Performance and Timing Model

An inherent characteristic of any FPGA is that the timing model is a variable model: logic implementation is accomplished by cascading a number of logic cells that is dependent upon the function to be implemented. For the non-ideal FPGA model, the device input to output propagation delay is a function of the number of logic cells in the signal path; the dynamics of the signal path in the logic cells; the number of programmable interconnects through which the signal traverses; the normal integrated circuit routing delay; and the I/O cell delay. This relationship can cause the variable timing model to be quite complex and depend upon the routing, placement, and cell dynamics. Since the 380 family has a fixed timing model for each logic cell, the cell configuration dynamics are not a factor in the 380 family timing model. Only the cell delays need to be summed for their contribution to the overall delay.

The 380 family programmable interconnect affects the propagation delay in much the same way as normal integrated circuit interconnects. That is, the fuses contribute to the delay as if they were slightly longer wires. This characteristic greatly reduces the complexity of the timing model and the variability in the timing results when a design is fitted to a device.

Conclusions and Summary

This application note is a first introduction to FPGAs. Specifics of the Cypress 380 family of devices were presented. It was shown that the fuse or connection technology has a very strong influence on the architecture of the FPGA logic cells and the interconnect scheme. Specifically the physical size of the interconnect (fuse or RAM cell) and its ON resistance are major influences on the FPGA logic cell complexity and interconnect architecture. The low ON resistance, physically small fuse permits

- an interconnect scheme virtually unconstrained by the number and location of fuses
- an optimized logic cell architecture
- device performance with minimal limitations from the fuse electrical characteristics

When an FPGA is architected with the freedoms listed above, the results are significant benefits to the user:

- FPGAs where designs are easy to fit, i.e., large capacity (non interconnect constrained)
- Flexibility in pin assignments
  (user can define/keep fixed after modifications)
- High performance/low propagation delays
- Easy to use timing models

When these are primary concerns, the small fuse technology offer the greatest opportunity for extracting these benefits.