Speed Metrics For High-Performance FPGAs

November,1997 (Version 1.0) Application Brief XBRF015

Summary

Performance data (in terms of circuit speed) is provided for several key logic and routing functions implemented in XC4000XL-09 FPGAs, for purposes of overall system performance estimation. Performance data also is provided for equivalent implementations in the Altera FLEX 10K-2 family devices.

Xilinx Family

XC4000XL

Introduction

As the capacity of Field Programmable Gate Arrays has reached and exceeded the equivalent of 100,000 logic gates, these versatile devices are now capable of holding entire digital systems and sub-systems. These systems typically consist of many different types of logic and memory functions, ranging from simple state machines to complex multiply-accumulators.

FPGA devices are almost always used to implement synchronous logic systems, with synchronicity provided by a low number (ideally, just one) of master clock signals. Such synchronous designs take advantage of the multiplicity of registers provided in popular FPGA architectures. Maximum system performance, in terms of circuit speed and throughput, is determined by the maximum achievable frequency for the master clock(s). This, in turn, is a function of the longest worst-case delay between synchronizing registers along the critical path of the design.

(Note: There are other aspects of FPGA “performance” besides circuit speed and clock frequency, such as power consumption, I/O drive capability, reliability, and cost. However, within this application note, the word “performance” refers to circuit speed and system throughput, and is expressed in terms of maximum clock frequency.)

Unfortunately, unlike many other types of ICs, the ‘maximum system frequency’ supported by a particular FPGA device is not a parameter that the user can find in a data sheet. The data sheets do provide valuable information, quantifying the performance of individual resources within the FPGA, such as the delay through a look-up-table-based function generator. However, overall system performance also is dependent on how the FPGA’s individual logic resources are interconnected and used to generate larger, more-complex system functions (e.g., state machines, multiplexers, adders, and FIFO memory buffers), and, in turn, how those functions are integrated into the total system design. Many factors influence achievable performance, including the architecture of the FPGA device, the nature of the application, the efficiency of the development tools, and the skill of the designer.

After a design has been completed (that is, “placed and routed” within the FPGA), worst-case timing analysis can be performed using tools such as static timing analyzers and timing simulators. However, system performance can be difficult to estimate prior to implementing the design - for example, when evaluating whether a particular FPGA device is suitable for a given application. Complicating matters further, published system-level benchmarks from FPGA vendors can be misleading, as they do not indicate the relative levels of effort or the “tricks” that were used to implement a particular design.

In order to estimate “typical system performance” for the high-density XC4000XL FPGA family, a suite of specific designs were implemented that mimic the “components of delay” within a system-level design. For comparison purposes, these same functions can be replicated in other manufacturers’ LUT-based FPGAs. The results not only provide insight into achievable performance levels within these FPGAs, but also suggest design methodologies and techniques that can be used to reach given performance goals.

Performance and FPGA Architecture

The main architectural resources of static-memory-based FPGAs are I/O blocks, logic blocks, and programmable interconnects. I/O blocks interface the external package pins to the internal logic of the FPGA. An array of logic blocks, referred to as Configurable Logic Blocks [CLBs] in the Xilinx architecture, supplies the functional elements for implementing the user’s logic. The most-popular FPGA families have logic blocks based on 4-input look-up tables and accompanying flip-flops; this LUT/flip-flop pair is referred to as a Logic Cell. For example, the Xilinx XC4085XL device includes the equivalent of 7,448 Logic Cells, and the Altera EPF10K130V device contains 6,656

logic cells within their respective logic arrays.[1] Programmable interconnect resources provide the routing paths for connecting the I/O and logic blocks into networks. Since programmable switches reside in the routing paths, routing resources as well as logic resources introduce delays along circuit paths and complicate the task of estimating system performance prior to actual FPGA implementation.

In typical synchronous systems, the flip-flops within the logic blocks are used to synchronize logic functions implemented with the look-up tables. In many cases, multiple LUTs need to be cascaded in order to implement a complex logic function (flip-flops within individual blocks can be bypassed); the maximum number of LUTs along a signal path between two synchronizing registers often is referred to as the number of "levels of logic" for that circuit. Of course, as the number of levels of logic increases, so does the number of logic block and routing delays, reducing the maximum possible clock frequency.

Besides these three main types of resources, many FPGAs have additional resources that facilitate system-level design. Principal among these are global clock buffers and dedicated clock distribution trees that minimize clock skew, thereby increasing achievable performance. Other key features of the XC4000XL architecture include SelectRAM memory (high-speed, single- or dual-port distributed memory resources), dedicated arithmetic carry-generation logic, internal three-state buffers, and wide edge decoders, and registers in the I/O blocks.

With these considerations in mind, Xilinx has defined and implemented a set of performance metrics for the XC4000XL, as described in Table 1. Taken together, these metrics quantify the performance of FPGA resources used in typical system applications. (These metrics, along with the results of implementing them in the XC4000XL-09 device, are discussed later under the heading "Performance Metrics and Results".)

Performance and Design Style

As mentioned earlier, FPGA performance is affected by the design style and methodology. Often, system performance can be changed by altering the design itself and/or by employing an appropriate level of control over the "automatic place and route" implementation tools.

"Pipelining: The pairing of LUTs with registers in the CLB architecture facilitates the use of pipelining techniques to increase the clock frequency of FPGA designs. Where long circuit paths would preclude the use of a high-speed clock, intervening registers can be placed along those paths, thereby decreasing the maximum distances between synchronizing registers. In other words, the number of levels of logic is reduced. However, the addition of such "pipeline stages" does increase the latency of the design (that is, the number of clock cycles needed for a given set of input data to propagate through the entire design).

Timing-Driven Tools: The Xilinx XACT*step*™ development system allows the user to enter the desired performance goals (either in terms of clock frequency or delays along designated paths) as input parameters to the FPGA implementation tools. These performance goals become part of the "cost functions" used to control both the placement and routing algorithms; thus, use of this feature often results in better performance results along known critical paths. Furthermore, the "level of difficulty" for these algorithms can be adjusted by the user, who can match the amount of computation applied to the problem to the difficulty of the design task.

Floorplanning (Placement Control): Designers can apply their knowledge of the structure of the design and the desired direction of data flow by entering placement constraints. These take the form of directives specifying specific locations for particular I/O and logic blocks in the design; blocks can be assigned to a specific location or to a selected region of the array.

Intellectual Property and Design Re-Use: For many common functions, pre-implemented and pre-verified designs, often referred to as design cores, can be incorporated into a design. Since this portion of the design has already been implemented in the target architecture, its performance capabilities are known prior to inclusion in the system. Such cores can be purchased from an intellectual property vendor, or can be re-used portions of previous designs.

In some instances, special tools can create cores from parameterized functional descriptions. Examples include the PCI bus interface modules created by the Xilinx PCICore Generation Tool and the digital signal processing modules created by the DSP Module Generator in the Xilinx DSP Tool Kit.[2-3] In each of these cases, the module generators create physical implementations of the required function for the target architecture. These implementations are relationally placed macros (RPMs). Since the optimal relative placement of the logic blocks in the modules is fixed by the module generator, the timing characteristics of these modules are known prior to their incorporation in the system-level design.

Table 1: Description of LUT-based FPGA performance metrics

Metric #	FPGA Resource	Performance Metric (in Terms of Maximum Frequency)
1	I/O Drivers	I/O operations referenced to a clock
2	Internal Interconnect	Registers separated by X columns and Y rows
3	Cascaded Logic Blocks	N cascaded look-up tables, M bits wide
4	Function Generators - Multiplexers	N-bit wide multiplexer
5	Function Generators - AND terms	N-bit wide “AND” term (constant comparator)
6	Logic Blocks and Carry Generators	M cascaded adders, each N bits wide
7	Memories	N-elements deep dual-port memory

Performance Metrics and Results

Seven different basic types of performance metrics were devised, as described in Table 1. These metrics were implemented in both Xilinx XC4000XL-09 and Altera FLEX 10K-2 devices, representing the fastest FPGA devices available from those manufacturers at the time. Specifically, the XC4085XL-09 and EPF10K130V-2 devices were targeted; these were the largest devices available in these FPGA families at the time of publication. The XC4000XL-09 results proved to be fairly consistent for all family members; however, the results for the FLEX 10K family were considerably more varied, dependent on device type within the family and the “packing density” [i.e., the percentage of logic blocks actually utilized within a given device.]

The Xilinx designs were entered using the VHDL language, synthesized with Synopsys FPGA compiler, and implemented using the XACT*step* M1 v3.7 development tools. In order to achieve the best possible performance for FLEX FPGAs, the FLEX designs were entered in AHDL which is a prioprietary hardware description language, developed by Altera. The FLEX designs were synthesized and implemented using the Altera MAX+plusII v8.1 development tools. Timing-driven placement and routing was used with performance goals that were known to be unachievable in order to obtain the best case along a given path. No other special techniques were employed, except as noted in the descriptions below; thus, the metrics reflect not only the capabilities of the FPGA architectures, but also the efficiency of their development tools.

The results were recorded in terms of maximum possible clock frequency in MHz, as opposed to units of delay. Clock frequency is a more meaningful measurement for synchronous systems, since it relates directly to system throughput.

The results shown below were taken directly from the timing information supplied by the development system tools as extracted from the physical implementation of the design, with the exception of the I/O frequency metrics, which are calculated directly from data sheet parameters. All results assume worst-case conditions.

The first five metrics focus on individual “components of delay” within the FPGA architecture: I/O buffer delays, routing delays, logic block delays, and combinations of these. Metrics 3, 4, and 5 all involve logic block delays. While these may look somewhat similar at first glance, they each represent different mixes of logic block and routing delays, ranging from narrow, deep functions (metric 3) to wide, shallow functions (metric 5). The last two metrics highlight special architectural features: carry logic and on-chip memory.

1. Input/Output Frequency

I/O frequency metrics determine the maximum frequency at which data can be transferred to and from an FPGA via its pins using registered I/O signals. The I/O frequency is the recipricol of the sum of the worst-case clock-to-output valid and the input data set-up delays. Two I/O frequency metrics were examined, differing in the origin of the clocking signal.

Internal I/O Frequency (F_{IO-INT}) is computed using the clock-to-output valid delay of the output register (T_{OKPOF}) and the input data set-up delay (T_{PICK}). The clock-to-out valid delay is measured with respect to the active edge of the clock at the output register itself (i.e., internal to the FPGA), as illustrated in Figure 1. F_{IO-INT} does not take into account clock distribution delays internal to the FPGA. However, it does identify an upper bound for communication with a device that is “private” to the FPGA (such as a memory device connected only to the FPGA), or when various techniques such as phased-locked loops are used to reduce the skew between internal and external clocks.

For the XC4000XL-09 family, FIO-INT is computed directly from the published data sheet parameters; F_{IO-INT} = 208 MHz (1 / (3.5 ns + 1.3 ns). This information is not available in the FLEX 10K family data sheet.

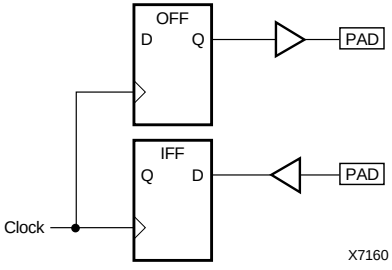

Figure 1: FIO-INT is derived from the clock-to-output delay of the output register with respect to the internal FPGA clock at that register and the input

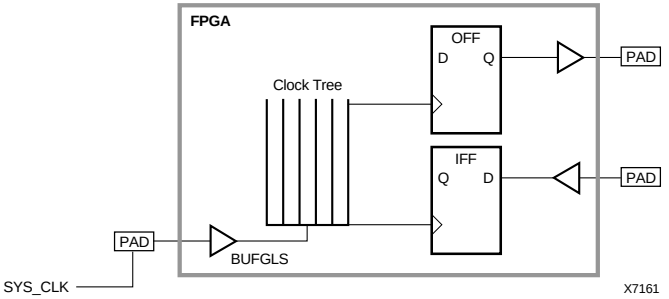

Figure 2: FIO-EXT is derived from the clock-to-output delay of the output register with respect to the external FPGA global clock and the input data set-up delay

External I/O Frequency (F_{IO-EXT}) is computed using the pin-to-pin global clock-to-output valid delay (T_{ICKOF}) and the input data set-up delay (T_{PSD}). The clock-to-out valid delay is measured with respect to the active edge of the clock at the global clock input pin (Figure 2); in other words, F_{IO-EXT} takes the global clock buffer and clock distribution delays into account. F_{IO-EXT} sets an upper bound on the communication rate to another FPGA using the same global clock signal. Smaller devices have significantly higher I/O performance, for instance the XC4013XL-09 = 91 MHz.

F_{IO-EXT} can be computed directly from published data sheet parameters. F_{IO-EXT} is 57 MHz for the XC4085XL-09 device, and 43 MHz for the FLEX10K130V-2.

2. Average Routing Delay

This metric quantifies the average delay associated with routing signals from one location in the FPGA's logic array to another. Specifically, this metric is the clock rate that can be supported when directly interconnecting two registers located at various positions in the array. Clock signals are supplied by a global clock network. Implementing numerous iterations of this metric with varying distances between the registers provides a good indication of routing performance in a system-level design; such designs typically include a broad mix of interconnect path lengths.

In the physical implementation, pairs of registers were configured as circular shift registers and placed in specific locations using placement constraints (Figure 3). Measurements were taken between two registers located in the same row (i.e., separated by horizontal distance only), two registers in the same column (i.e., separated by vertical distance only), and two registers located diagonally across the FPGA.

Within their respective architectures, the registers in the XC4000 CLBs and FLEX LABs are considered to be arranged vertically. In other words, the two registers within an XC4000 CLB and the eight registers in a FLEX LAB are, by definition, a single unit of vertical routing distance from each other. For example, a routing path of vertical distance 2 would go from the upper register in an XC4000 CLB to the upper register in the CLB directly below it, and from the uppermost logic element (LE) in a FLEX (Logic Array Block) LAB to the third LE in the same LAB. A routing path of vertical distance 9 would go from the register in the uppermost logic element (LE) of a FLEX LAB to the register of the uppermost logic element in the LAB directly below it.

Within a single implementation, measurements were made at horizontal distances ranging from 1 to 56 logic cells, vertical distances from 1 to 127, and diagonal distances from 1 to 191 (in Manhattan distances; that is, the sum of the horizontal and vertical distance). To the extent possible, all the trials were placed in the same horizontal, vertical, and diagonal plane, intentionally congesting the design so that the automated routers were forced to make choices and trade-offs between available routing resources. The XC4000XL design was implemented in VHDL with placement constraints in a.ucf file; the FLEX design was implemented in AHDL with placement constraints in a .acf file. It should be noted that the original Altera design failed to route, and as a result the vertical routing pattern for the FLEX design had to be modified by spreading it over 8 columns in order to achieve a successful route.

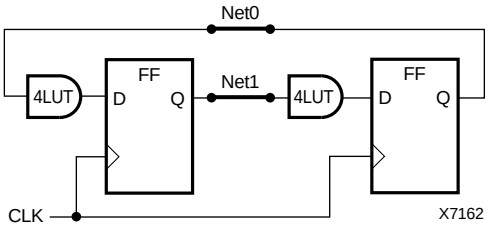

Figure 3: To measure routing delays over various distances, two registers are configured as a circular shift register. Note that the maximum frequency is dependent on the delay through the slower of the two interconnects between the logic blocks.

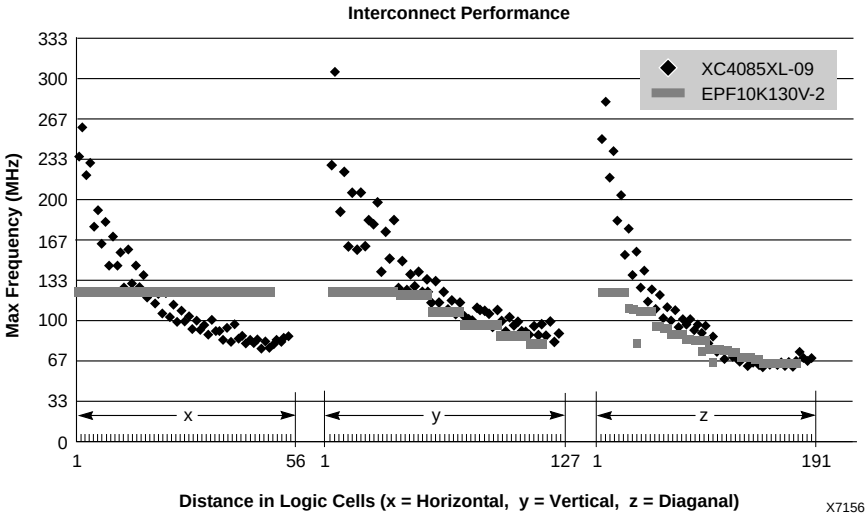

Figure 4: Results of Routing Delay metric implementation for the XC4085XL-09 and FLEX10K130V-2 devices.

The results of these trials are shown in the graphs in Figure 4. Since the interconnect architecture of the FLEX device includes fewer levels of hierarchy than the XC4000XL architecture, the “routing distance vs. frequency” curves on this graph show descreet levels for the FLEX device. However, with the exception of long horizontal paths, its segmented routing scheme allows the XC4000XL architecture to consistently out-perform the FLEX architecture in terms of routing delays over a given distance.

3. N-Level Combinatorial Logic

This metric measures the maximum frequency for a N-level chain of LUTs, each with M inputs. 4-input LUTs were examined, with chains ranging from 1 to 6 logic levels. In each case, the LUTs were fully-loaded and fully-interconnected. For example, Figure 5 shows the circuit used to test a 2-level chain of 4-input LUTs. The entire suite of LUT chains was implemented in a single design. With its emphasis on LUT and local routing performance, this metric is a good indicator of state machine performance within the FPGA.

For both the XC4000XL and FLEX design, the LUTs were instantiated as native LUTs. The Xilinx implementation used .xnf macros for 4-input LUTs instantiated within a VHDL file. The Altera implementation, written in AHDL, used LCELL primitives.

The results when implementing the LUT-chains are shown in Figure 6. The Xilinx device outperformed the FLEX FPGA in each case.

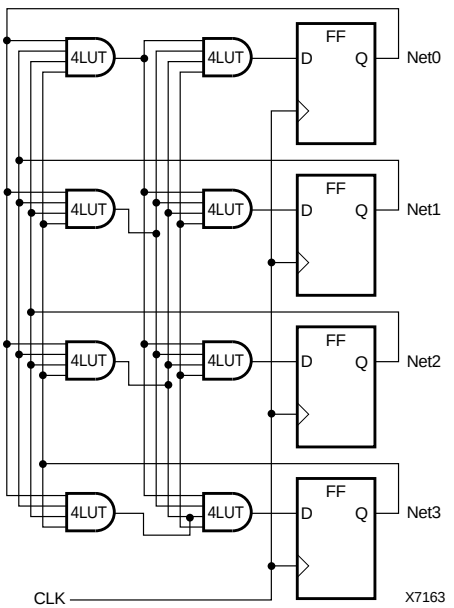

Figure 5: A 2-level chain of 4-input look-up tables

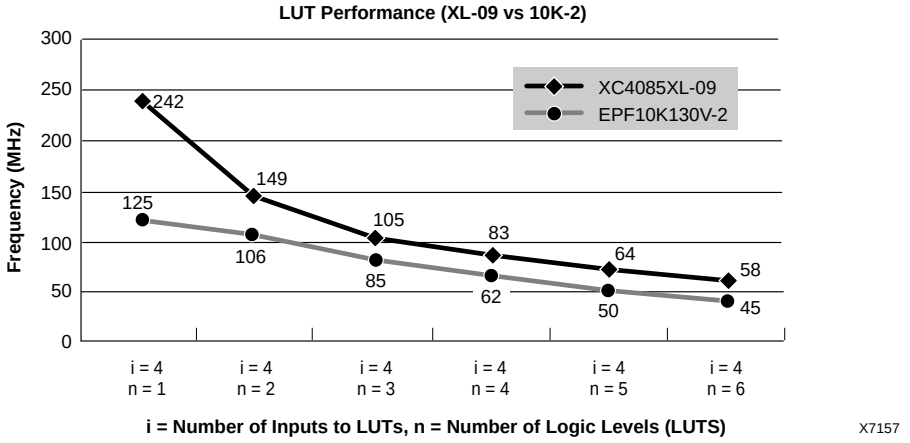

Figure 6: Maximum frequencies for chains of N-level look-up tables implemented in the XC4085XL-09 and FLEX10K130V-2 devices.

4. N-to-1 Multiplexer

Multiplexers represent a common class of circuit elements in FPGA designs, and are typically needed in bus interfaces, register files, memory buffers, and similar applications. In this test, a suite of 2:1, 4:1, 8:1, 16:1, 32:1 and 64:1 multiplexers were implemented, and placed between registers. Each set of multiplexers had 64 inputs; thus, 32 of the 2:1 multiplexers were implemented, 16 of the 4:1 multiplexers, and so on. Each set of multiplexers shared the same 64 inputs, introducing a significant amount of routing delay. All the multiplexers in the suite were implemented at the same time in a single device. As with the previous metric, this test focuses on logic block and routing performance.

The design was entered as trees of 2:1 multiplexers, using VHDL for the XC4000XL design and AHDL for the FLEX 10K design. The automatic placement algorithms were allowed to place the multiplexers and registers.

As shown in Figure 7, the XC4085XL-09 was up to 70% faster than the EPF10K130V-2 device for this metric. Again, part of the Xilinx performance advantage in this metric stems from the logic block architectures. In the FLEX architecture, implementing each 2:1 multiplexer requires an entire 4-input

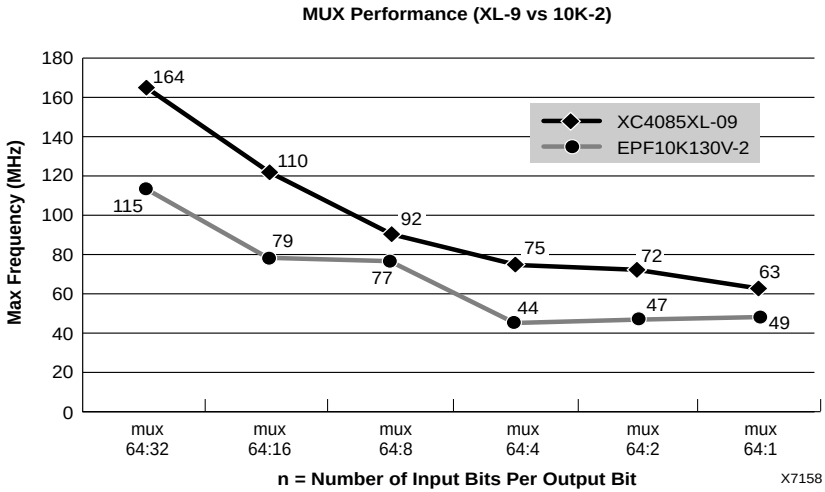

Figure 7: Maximum frequencies for N:1 multiplexers implemented in the XC4085XL-09 and FLEX10K130V-2 devices.

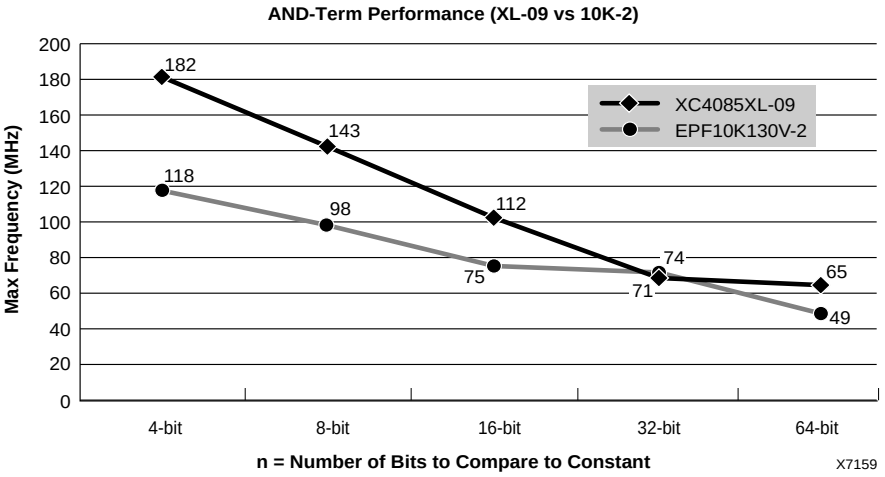

Figure 8: Maximum frequencies for sets of N-bit wide AND-terms (constant comparators) implemented in the XC4085XL-09 and FLEX10K130V-2 devices.

LUT. With its cascaded LUTs, The XC4000XL can implement a 4:1 multiplexer in each CLB. The synthesizer and mapper automatically took advantage of this feature, thereby allowing the larger multiplexers to be implemented in a more efficient manner than a simple tree of 2:1 multiplexers.

5. N-Bit Wide AND Term

This metric tests the ability of the FPGA architecture to implement wide combinatorial functions. Sets of AND gates (constant comparators) with 4-, 8-, 16-, 32-, and 64-bit wide inputs shared a common set of 64 input bits. Thus, 16 of the 4-bit AND-terms were implemented, 8 of the 8-bit AND-terms, and so on. All the constant comparators in the suite were implemented at the same time in a single device. As with the previous two metrics, this test focuses on logic block and routing performance, with an emphasis on the ability to cascade multiple LUTs. In system implementations, similar wide functions may be required to implement decoders (as commonly generated from high-level “case” statements) and large state machines.

Figure 8 shows the results of implementing this metric. With its ability to cascade LUTs within a single CLB, the Xilinx architecture allows for more efficient implementations of these cascaded circuits. As a result, the XC4085-09 again significantly outperforms the FLEX 10K-2 device.

6. Chained Adders

This metric measures the maximum frequency for a chain of adders placed between registers. 8-, 16-, 24-and 32-bit adders were each placed in chains of length 1, 2, and 4. For example, Figure 9 shows the structure for chaining two 32-bit adders. The performance of adder chains is a good indicator of the relative performance of these FPGAs in more complex mathematical operations such as multipliers, dividers, magnitude comparators, maximum, minimum, and even transcendental transforms. The high performance in these types of circuits has led to the widespread use of high-density FPGAs in digital signal processing applications.

Modern FPGAs such as the XC4000XL and FLEX 10K families include dedicated ripple carry logic, providing for the implementation of carry propagate functions without using the more general-purpose LUTs. The performance of this metric is heavily dependent on the speed of these dedicated carry generation and propagation resources.

For the Xilinx implementation, the adders were described with “+” operators in VHDL. For the Altera implementation, the adders were instantiated as LPMs (Library of Parameterized Modules) within the AHDL file. The entire suite of adders was implemented in a single FPGA.

As shown in Figure 10, the FLEX device was slower in all instances except for a double 8-bit adder. The Xilinx performance advantage was especially significant for larger adders, where the XC4000XL-09 FPGA is up to 50% faster

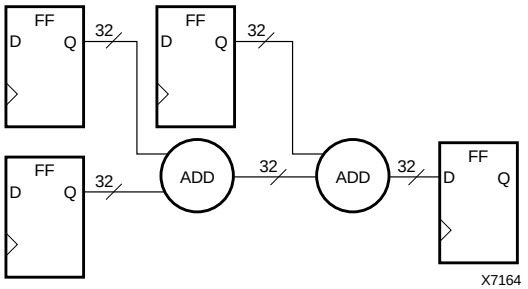

Figure 9: Chain of two 32-bit adders

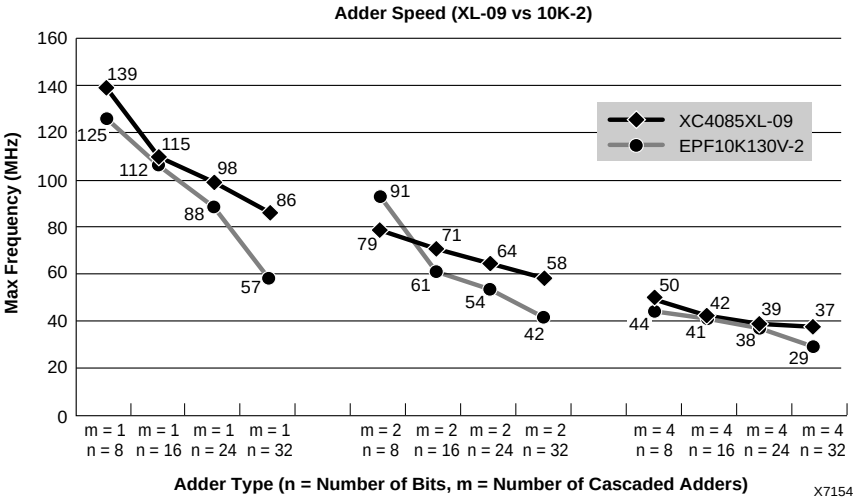

Figure 10: Maximum frequencies for chains of multi-bit adders implemented in the XC4085XL-09 and FLEX10K130V-2 devices.

when implementing 32-bit adders. Again, part of the Xilinx performance advantage is due to architectural differences; the XC4000XL architecture has fast, dedicated routing resources for propagating the carry signal between CLBs, while the FLEX 10K architecture requires the use of general-purpose interconnect to propagate the carry signal between LABs.

7. Dual-Port Memory

The final metric is a measure of internal memory bandwidth. By replicating the circuit shown in Figure 11, 16-bit dual-port memories were tested at depths of 16, 32, 64, and 128 words. As a peak performance indicator, the write cycle time is examined. (In general, all other delays can be reduced below the write cycle time through the judicious use of pipelining, register duplication, and RAM placement.) Among the most useful and powerful memory structures, dual-port memories are key to a wide variety of interface and storage applications (e.g., PCI bus interfaces).

The XC4000XL design was entered in VHDL; the RAM elements were instantiated with DPRAM primitives. The memory structures for the FLEX FPGA were implemented as per Altera Application Note 65, “Implementing Dual Port RAM in FLEX 10K Devices.”[4] The entire suite of dual-port memories was implemented in a single device.

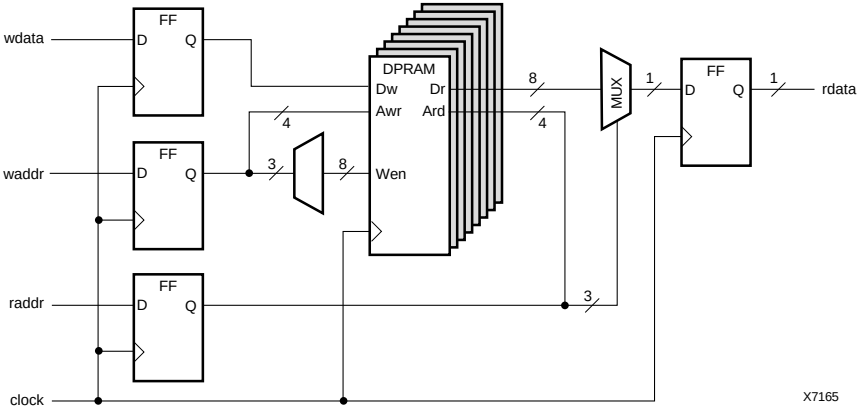

Figure 11: Circuit used to test dual-port memory bandwidth.

While both the XC4000XL and FLEX 10K architectures include on-chip memory resources, their structures are quite different. In the Xilinx architecture, any of the 4-input LUTs in the CLBs can optionally be used as a 16-by-1 memory. The Altera architecture includes discrete blocks of memory located in fixed positions in the array and called Extended Array Blocks (EABs).

Due to architectural limitations, the FLEX 10K memories can operate at a maximum of only 23 MHz in dual-port mode (Figure 12). Since FLEX EABs are always 256 words deep, this result is the same for all four memory sizes tested. The flexible distributed memory of the XC4000XL architecture can operate internally at frequencies up to 128 MHz for 16-word deep buffers, and up to 52 MHz at a depth of 128 words.

Designing for Performance

Taken together, these metrics provide a basis for estimating the performance of system-level applications in FPGAs, as well as documenting the high performance levels that can be reached with the XC4000XL family devices. They also provide insight into the design methodologies and amount of effort that are likely to be required to reach a given performance level.

Experienced FPGA designers are aware that the design of the circuit itself can be altered to increase FPGA performance - for example, through the addition of pipeline registers or the duplication of heavily-loaded logic. Further performance gains may result from the proper use of tools such as timing-driven place and route algorithms and floorplanners to achieve more-optimal physical implementations. The designer's performance goals often dictate when such methods are needed.

The performance estimates below are derived from both the metrics explained above and the actual implementation of various system-level designs. This information applies to the XC4000XL-09 FPGA family devices, and is intended only as general guidelines; again, varying applications may experience varying results.

Low Frequency Designs: < 40 MHz

As indicated by the performance metrics, performance levels below about 40 MHz are easy to obtain with the XC4000XL-09 FPGAs. Low frequency designs typically have few registers and a high "LUT-to-register" ratio; time delays from one register to another may include multiple levels of logic and long interconnect delays. A low frequency design often is the result of using logic synthesis when the high-level source code is not written with FPGA architectures in mind; pipelining is not automatically implemented by most synthesis tools.

Low frequency designs tend to have the following characteristics:

- Placement is not critical, since extremely long routing delays are tolerable.
- Pipelining is not needed since the logic depth is not critical; multiple levels of logic - up to as many as 8 levels - can be placed between registers.
- Wide (32-bit) arithmetic functions can be directly implemented and chained together.
- Large dual-port memories can be accessed in a single clock.
- Typically, timing-driven tools will not be needed; however, if timing problems do appear, the simple application of global timing constraints usually is all that is needed to rectify the problem.

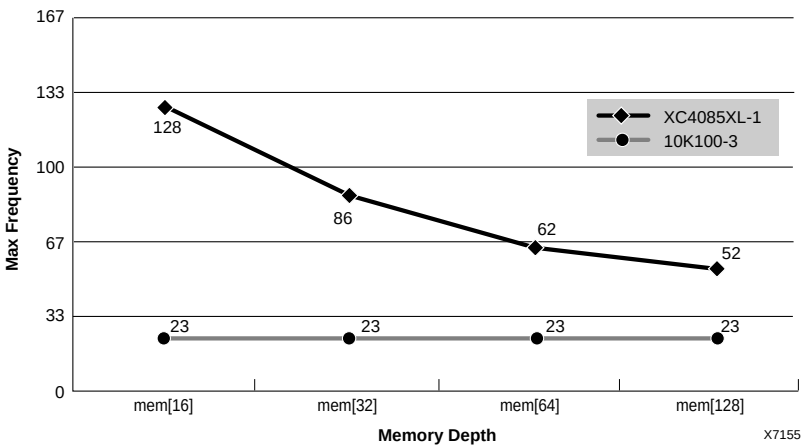

Figure 12: Maximum frequency for 16-bit dual-port memories implemented in the XC4085XL-1 and FLEX10K100-3 device.

Mid Frequency Designs: 40 MHz - 80 MHz

Mid frequency performance is typical of designs that target an FPGA architecture. The LUT-to-register ratio for these designs often approaches unity, as registers are included to pipeline wide, cascaded logic operations.

Mid frequency designs tend to have the following characteristics:

- The FPGA is able to communicate with the system via registered inputs and outputs.
- Routing can span long distances horizontally, vertically or diagonally.
- A reasonable number of logic levels (4) can reside between pipeline registers.
- Single 32-bit wide words can be added between registers.
- Dual-port memories up to 128 words deep can be accessed in a single clock.
- The use of timing constraints and timing-driven place and route tools is necessary to set appropriate performance goals along the design's critical paths.

High Frequency Designs: 80 MHz to 130 MHz

High frequency designs attempt to deliver the maximum possible performance from the FPGA architecture. As a result, they often require careful floorplanning, extensive pipelining, and the use of timing-driven tools. Pipeline registers are used liberally - possibly even at each end of a long interconnect path - and wide operations are broken down into smaller pieces. For example, a 32-bit adder may be implemented as 4 pipelined 8-bit adders. Inputs to LUTs are registered as close as possible to the LUT to decrease interconnect delays. Logic capacity may need to be sacrificed in order to insure that enough short, fast local interconnect lines are available to handle all the routing needs.

Fortunately, it is often the case that only a small portion of a larger design needs to run at these peak rates. Thus, the designer's effort can be directed to a small and manageable portion of the design.

High frequency designs tend to have the following characteristics:

- Only registered I/O-to-I/O operations referenced to the same internal clock can keep pace.
- Interconnect can span only local distances.
- Only one or two logic levels can reside between pipeline registers.
- Wide mathematical operations must be broken into smaller pieces, such as 8-bit pipelined operations.
- Dual-port memories up to 16 words deep can be accessed in a single clock.
- The use of timing constraints and timing-driven place and route tools will be required; user intervention in the placement process using floorplanning tools also may be necessary.

Regardless of the performance goals, the first implementation attempt may not always produce the desired results. The design process, of course, is an iterative process. Designers can experiment with different approaches, change circuits, alter options within the implementation tools, examine the results, and use the knowledge gained as a starting point for the next iteration.

Summary

It is often difficult to predict FPGA performance prior to the actual physical implementation of the desired design. To aid in this process, a set of performance metrics was developed for the XC4000XL FPGA family that examines the individual "components of delay" within an FPGA architecture. The metrics provide valuable insight into the design techniques and methodologies required to reach a given level of performance.

These metrics show that the XC4000XL-09 FPGA family significantly outperforms the Altera FLEX10K-2 FPGA family.

References:

1. Xilinx Application Brief XBRF 011, *"An Alternative Capacity Metric for LUT-Based FPGAs."*, Feb. 1, 1997
2. Xilinx Core Solutions Products Catalog, May, 1997
3. Xilinx Product Description, *"CORE Generator Tool for PCI"*, April, 1997
4. Altera Application Note 65, *"Implementing Dual Port RAM in FLEX 10K Devices"*, Feb., 1996