Tips and Techniques for 28-nm Design Optimization

WP-01172-1.0 White Paper

Timing closure is of critical importance in high-speed FPGA designs. This white paper focuses on the challenges that affect timing closure and discusses how simple HDL changes can help resolve timing issues and reduce the time needed to achieve timing closure.

Introduction

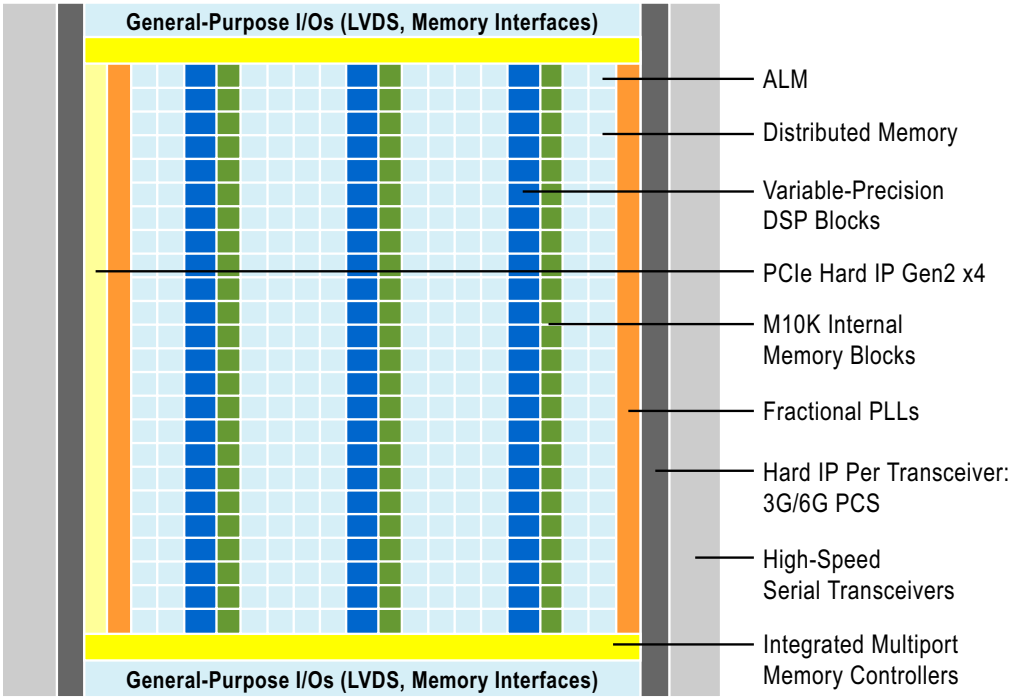
The tools available in Altera's Quartus® II development software are designed to help address the challenges that effect timing closure—the availability of critical resources, the amount of routing congestion both local and global, and the ability to accurately time the logic to avoid timing volitions that could otherwise be caused by skews within the clock network—but often simple HDL changes will go a long way towards resolving timing issues and reducing the time it takes to achieve timing closure. This white paper addresses these possible HDL changes and their relationship to the architecture and layout of the FPGA. This paper is not intended to be a comprehensive look at all timing issues but a guideline to good coding practices with a specific emphasize on how these apply to designs implemented in Altera's 28-nm portfolio of devices.

Optimizing for Timing, or Finding the Shortest Path

Figure 1 shows an abstract top-level view of Altera's Arria® V FPGA architecture. All of the resources for the FPGA are laid out in a matrix of columns and rows with I/Os framing the top and bottom of the device and serializer/deserializer (SERDES) transceivers covering the left and right sides. Fractional phase-locked loops (PLLs) are collocated on the sides with the transceivers as well as in the center of the FPGA. Digital signal processors, adaptive logic modules (ALMs), and memory are distributed in regular columns throughout the FPGA. This column mapping of resources applies to the entire range of Altera's 28-nm products with the exact mix of resources (columns) dependent upon the features of that FPGA.

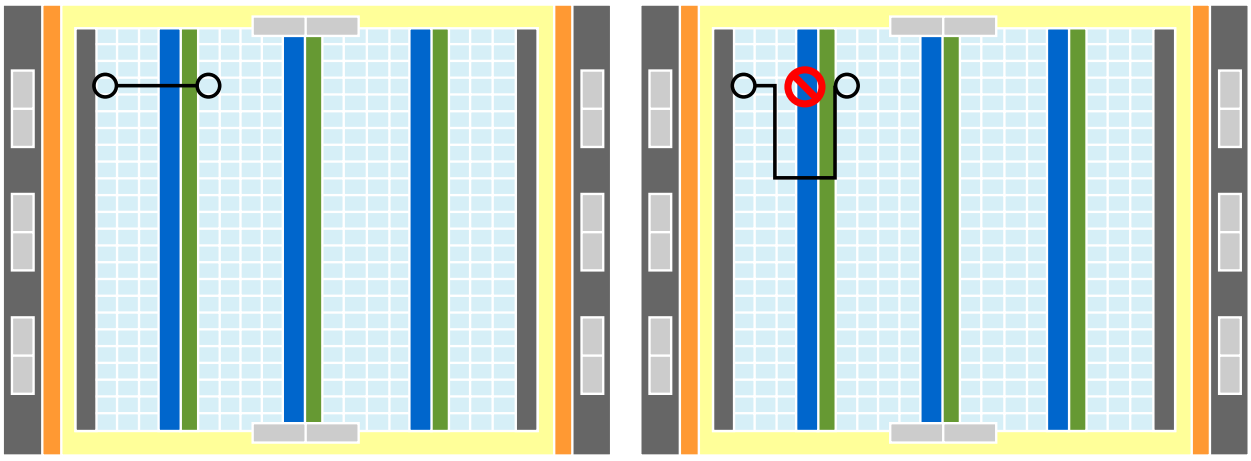
101 Innovation Drive
San Jose, CA 95134
www.altera.com

Figure 1. Top-Level Device View of Arria V Architecture



The layout of the FPGA is a regularly repeating matrix of resources, which are accessed via a mesh of interconnected fabrics. In other words, they are connected together through a series of horizontal and vertical pathways. As shown in Figure 2, the fitter tries to always choose the shortest available path through the device but often that path is blocked because it is used by other pieces in the design.

Figure 2. Path Analysis in an FPGA



The fitter therefore acts much like the GPS used in most cars today. It is constantly trying to find the shortest path through the network by choosing between different paths based on available resources and based on the constraints placed on it. The path it finds will vary depending on how many different paths it must choose from and the distance it must go.

Tip 1—The Highway is Not Always the Fastest Way (Retiming the Logic)

In 28-nm FPGAs, the amount of resources that can be placed within the mesh has grown dramatically. Gone are the days of devices with only 100K logic elements (LEs). Devices are now approaching 1M LEs, but the basic underlying interconnect mesh remains the same. This means that the cell delays have gotten faster while the interconnect delays have gotten slower. Using the GPS analogy, the main highways (interconnects) are often congested because there are so many more devices competing for access to those pathways in larger devices that sometimes using local streets (local cell connections) can be much faster. Anyone who has driven in rush-hour traffic of any major city can understand the desire not to take main highways during peak times.

One way to avoid congestion is to collocate critical resources on critical pathways preferably within the same cell. The shorter the distance the signal must travel, the less likely the fitter must use the longer interconnect lines that may be congested and thereby cause delays on the path. This is often known as retiming the logic. The best way to do this is to evaluate the HDL code and break up large chunks of logic into smaller pieces and limit the number of dependencies between the different pieces. A good indicator of whether there will be issues is the numbers of layers of logic that exist between points in the timing paths. The more levels of logic there are, the more likely the resources available to complete the function are not collocated. The more resources are spread out across the device, the heavier use of interconnect paths is needed to reach them.

Tip 2—Don't Drive During Rush Hour (Pipeline Aggressively)

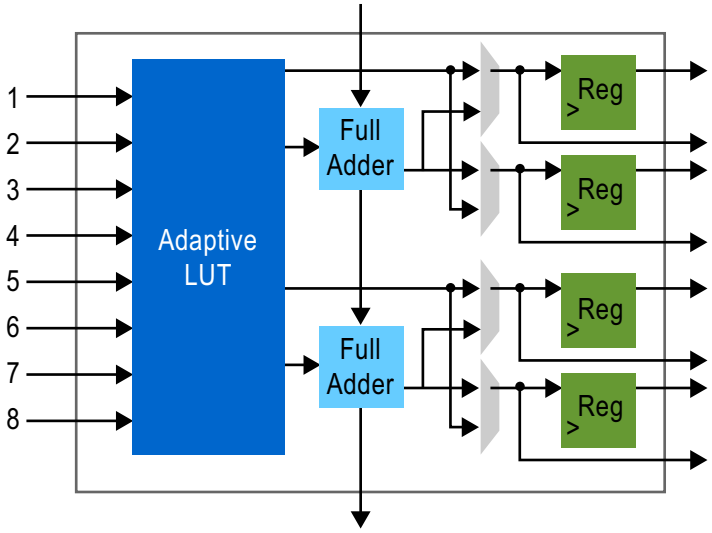
Again using the GPS analogy, the time it takes to get from point A to point B in a major city is highly dependent upon the time of day. If someone attempts to drive during the end of a workday, it is more likely that person will encounter congestion because many people are trying to go home from work. On the other hand, that same path is completely free of traffic at midnight when most people in the city are asleep. In FPGA designs, the best way to manage congestion is to pipeline the design.

Pipelining a design adds additional latency in the system but it avoids timing problems by in effect changing the relationship between the clock and the data to ensure that they arrive at the destination at the same time. Pipelining can reduce the effect of longer interconnect delays that are not seen by the clock. Clocks travel on a separate network of paths than their data or logic counterparts, so timing problems can occur if the data comes before or after the clock expects it. (These are generally considered set-up and hold violations.) By inserting pipeline registers into the path, the signal is retimed so that both the clock and the data appear to the next piece of logic at the same time. This is the preferred way to handle long interconnect delays and is an easy way to avoid timing violations.

Altera recognized the need for more pipelining in its larger FPGAs and has redesigned its LEs and adaptive look-up tables (ALUTs) (Figure 3) to include an additional two registers. There are two reasons for this change. First, the additional registers provide more register resources so that normal logic functions are not impacted by the need for an additional pipeline register. Previously, an ALUT was wasted when a pipeline register was used to retime a design. Second, this change allows for collocation of the pipeline register with the logic it is intended for. Collocation is important because the pipeline registers are used to remove long

interconnect delays. If the pipeline registers were not collocated with logic, then that would introduce yet another interconnect delay as the data must go to a register—which may be located at the other end of the device—and come back to the ALUT. The additional registers ensure that the pipeline register can be placed close to the logic, thus minimizing any delays in the system.

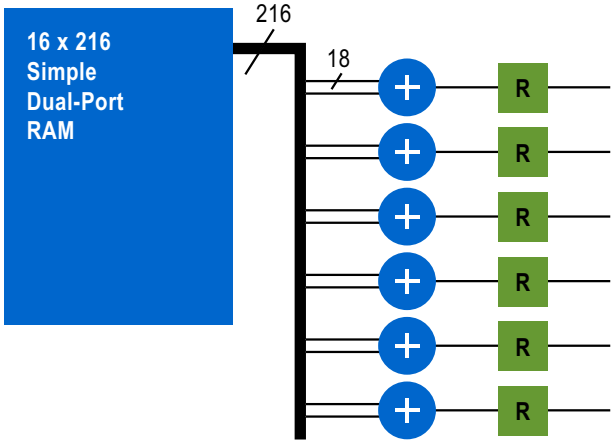
Figure 3. Altera's 28-nm ALUT



How Pipelining Helps Designs

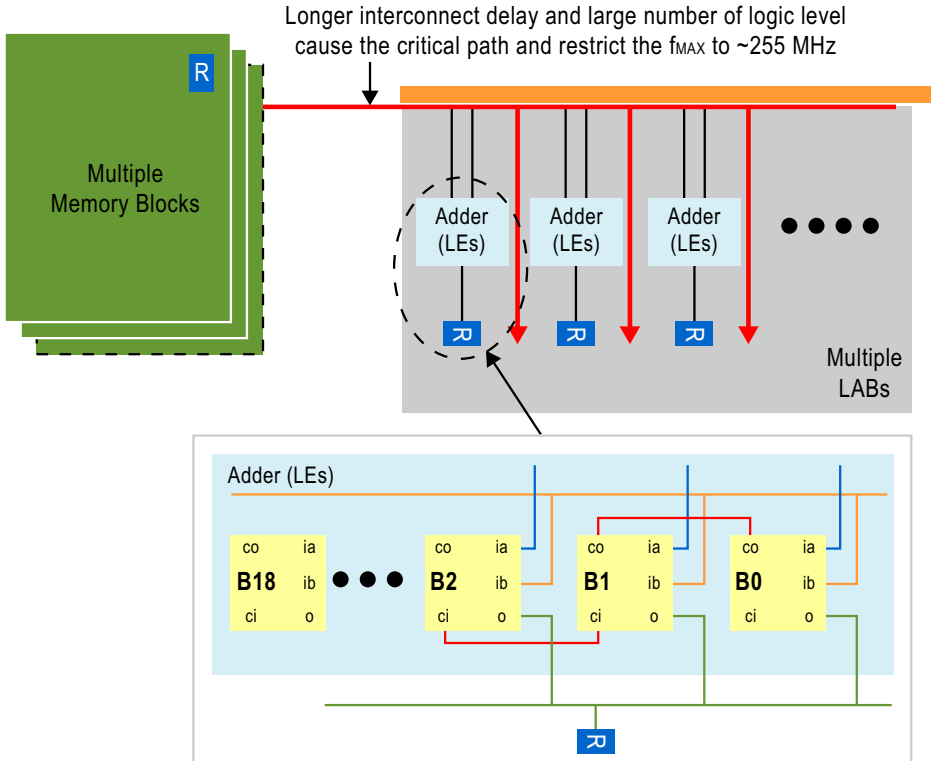
This example addresses a 16-bit x 256-bit simple dual-port RAM (Figure 4) instantiated in an Arria V 360K-LE device. Due to high fanout of the RAM to multiple LEs, the speed on this path was restricted to 255 MHz.

Figure 4. 16x216 Simple Dual-Port RAM



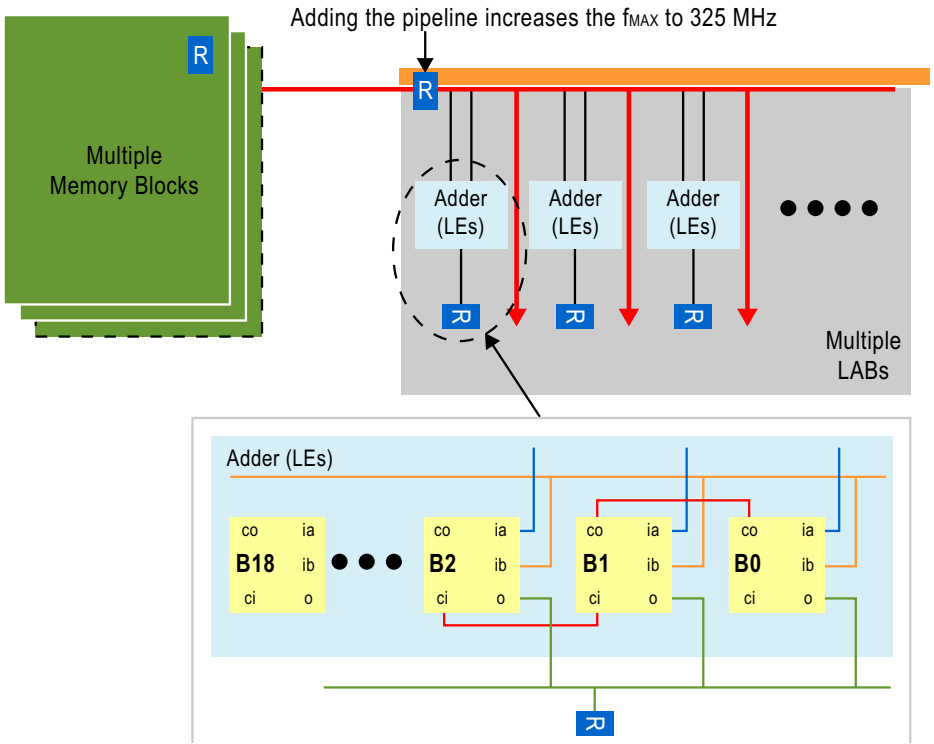
Further analysis of the design, illustrated in Figure 5, shows that the interconnect delays between the RAM and the adder account for 54% of the total delay in the system.

Figure 5. Path Analysis of Simple Dual Port Design



As shown in Figure 6, the insertion of a single pipeline register before the logic eliminates much of the interconnect delay, allowing the design to run at 325 MHz. This difference of 70 MHz is a 27% increase in performance provided just by inserting a single pipeline register in the design.

Figure 6. Simple Dual-Port Design with Addition of a Pipeline Register



Tip 3—Carpool If Possible (Avoid High Fan-Out Nodes)

In logic, often the same signal must arrive at multiple locations at the same time. Since the signal is launched from the same resource, the fitter must find a path for all the copies of the signal and ensure that they all arrive at their respective locations at the same time. This type of situation is called a high fan-out node, and can be seen in Figure 4. The challenge is that it may become difficult to ensure all the signals arrive at their destination at the same time because all five copies cannot use the same routing resources to get to their respective destination and since the destination may be located in different areas of the device. The best way to avoid this problem is to use node replication.

Node replication can be done manually or automatically in the tool. As an example, 20 copies of the same signal need to appear on the same logic at the same time. It may be extremely difficult to ensure that all 20 paths exactly match and meet timing, especially if the design is very full. Using node replication, the design is broken into two stages. The first stage is a 1-to-4 fan-out, and the second stage is a 1-to-5 fan-out. Adding the two stages together still provides the equivalent of a 1-to-20 fan-out of the signal, but it breaks it into two separate and more manageable problems. It is easier to match four or five paths through the FPGA than it is to match 20 paths.

Reducing Congestion

The previous sections address ways to retime the logic to avoid timing issues in the design. This section and the next address structural things to consider before even starting the design. The first of these is congestion in the FPGA, which is caused when too many LEs compete for the same set of resources, whether they are clock resources or interconnect lines. Because the fitter must trade off the needs of the different LEs, the more LEs that are used, the fewer available options the fitter has for choosing optimal paths. Designs more than 85% full will have a harder time closing timing than designs that are only 50% full. Altera's fabric allows for designs up to 90% full but care must be taken to ensure timing closure on those designs. There are a couple of ways to improve timing in very congested designs. They are the use of partitioning and the management of clock networks.

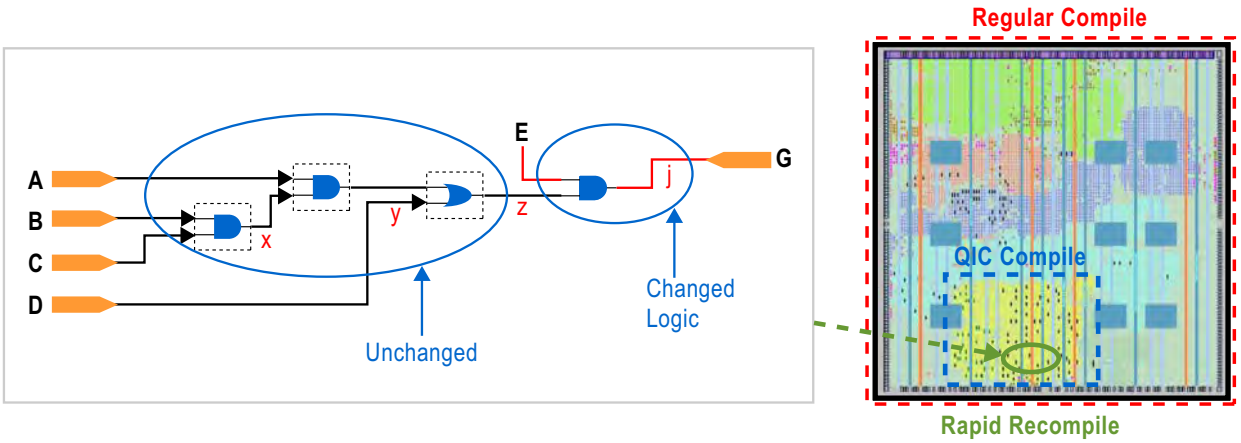
Tip 4—Think Globally but Act Locally (Use Partitioning and Hierarchical Designs)

A hierarchical design approach has many advantages in FPGA design. First, it allows for partitioning of the design into smaller blocks. Partitioning allows for the use of incremental rapid recompile, a tool offered in Quartus II software that allows users to recompile only a portion of the design. This tool is a huge benefit because it helps the fitter focus only on the section of the design that may have changed, thus simplifying the routing problem and ensuring that the rest of the design remains unchanged.

By focusing the fitter, timing closure becomes easier because the fitter does not need to take into account all the paths in the design, as opposed to when a user complies an entire design, the fitter has to take into account all dependencies. This method can lead to different routes every time the design is compiled and result in different timing results. Since the fitter does not know what was and was not changed in the design, it will assume everything has changed and will use its algorithm to try to refit the entire design.

In addition, the overall compile time of a design will decrease when using rapid recompile because only the section that was changed will be rerouted. Figure 7 shows a portion of a design and how rapid recompile is used on it. A partition can be as small as a single ALUT and as big as the entire device.

Figure 7. Rapid Recompile vs. Regular Compile



Traffic Management

A second structural method is the use of hierarchical design to allow for the use of system-test bedding tools such as Altera's Qsys system-integration tool. Qsys allows the user to quickly and easily connect different blocks of intellectual properties (IPs) and integrate them into Altera's suite of debugging IP. This integration allows for virtual testing of the entire system and the ability to gauge performance as well as functionality of the design. Partitioning is a great way to break IPs into different blocks for integration into Qsys and an overall hierarchical design flow. Other benefits of partitioning include design reuse, because the IP can be preserved in blocks and can be preserved with the same routing, performance, and multisite design activities as different locations work on different parts of the overall design. Figure 8 shows the advantages of hierarchical design and Qsys.

Figure 8. Qsys Vision

Tip 5—Make Sure Trains Run on Time (Manage Clock Skews and Clock Networks)

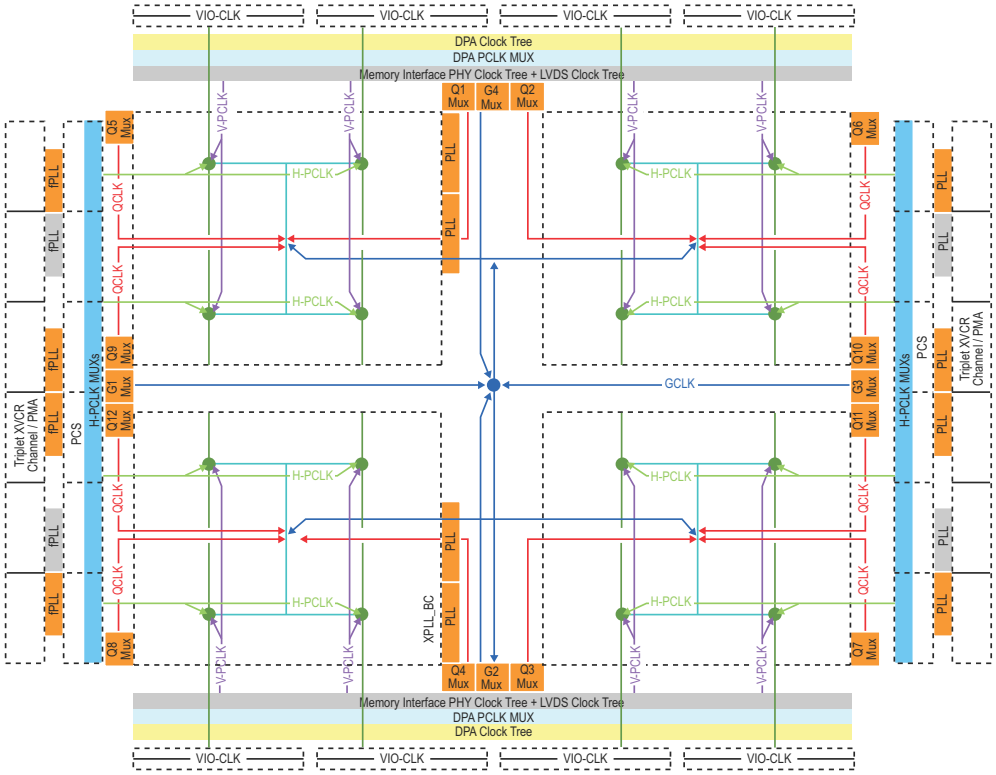
There are six different types of clocks in Altera's FPGAs: GCLK, QCLK, PCLK, SCLK, ROWCLK, and VIOCLK. Each clock from ROWCLK to GCLK covers a progressively wider area of the device. Unlike the interconnects where the suggestion was to avoid longer lanes (or highways) due to the amount of resources contending for access to those lanes, clocks are few in number. The larger clock lines such as GCLK allow quicker propagation of the clocks, which limits clock skew and improves both set-up and hold timings.

It is best to get the clock signal from PLL to the internal resources as quickly as possible. This means immediately transitioning from a local clock network such as a ROWCLK to a higher clock network such as QCLK or GCLK. This is called clock promotion, which the fitter will attempt to do automatically. In cases where the clocks are restricted due to a software design constraint (SDC) file, the user must ensure that the clocks are promoted where ever possible. Table 1 shows the different clocks and the regions they cover, while Figure 9 shows the layout of the clock network for Arria V FPGAs.

Table 1. Different Clocks and Clock Regions in Altera 28-nm Devices

Network Name	Clock Region Coverage/Usage
GCLK	Device-wide network
QCLK	Quadrant Side-wide Spine-segment
PCLK	Spine-segment Vertical spine segment Horizontal spine segment
SCLK	Access 1/16 of core (4-spine segment device) Access 1/8 of core (2-spine segment device)
ROWCLK	Access core resource: LAB/M10K/DSP and HIO register
VIOCLK	Access VIO register

Figure 9. Graphical View of Altera's Clock Regions, PLLs, and Clock Multiplexers



Timing Closure is Complicated

Even with the best planning and the best HDL code, sometimes it is still difficult to close timing. This is especially a problem with very large designs where it is time consuming to understand every route and identify which ones can benefit from timing enhancements. Also simply knowing which enhancement would help the best in any given situation would be difficult to understand.

Tip 6—Ask for Directions (Use Altera's Automated Timing Closure Analysis Tool)

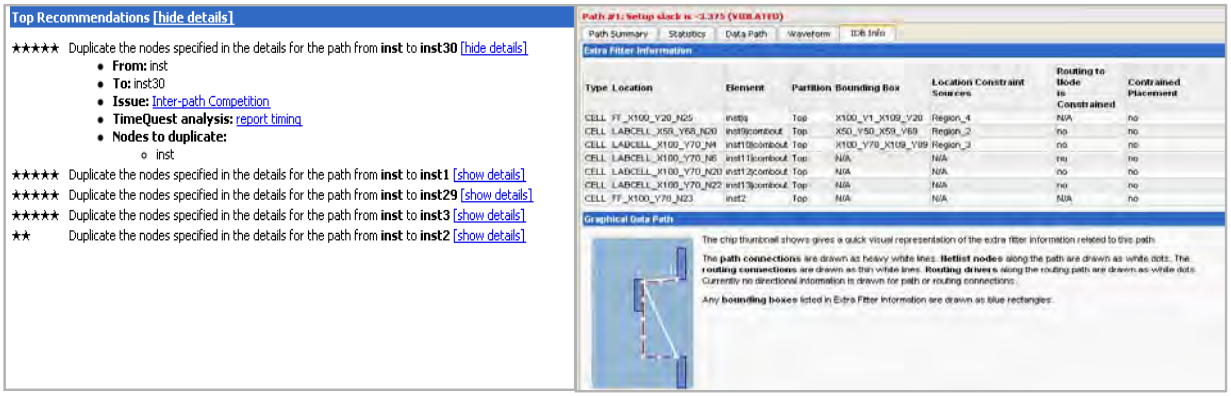
Within Quartus II software, Altera's automated Timing Closure Analysis tool that can be launched via the TimeQuest timing analyzer. The tool is designed to help guide the designer by providing a set of recommendations that the user can evaluate and decide if it is worth trying. The recommendations will not be perfect because the tool does not know if the changes are feasible or easy to try. The tool does help determine what portions of design a user must revisit and explains what changes would most likely help with that circuit.

The overall goal is to provide intelligent advice to the user because it is often unclear where to start looking at timing failures. The tool looks for the following timing issues:

- Large clock skews
- Restricted optimizations where Quartus II software was not allowed to retime or duplicate
- Unbalanced logic that would benefit from retiming the path
- Region constraints where nodes on the path are locked to non-overlapping regions
- Partition constraints that occur when a path crosses partition boundaries
- Too much logic for the given timing constraint
- Use of control signal paths on critical logic
- Reduced optimization focus by Quartus II software because the fitter did not see this path as being critical
- Interpath competition

The tool has a link that explains each problem and the corresponding recommendation. It also ranks recommendations using a star system so that the user can focus on those that will have the biggest impact on the timing of the system. Figure 10 shows an example of the tool and its results.

Figure 10. Timing Closure Analysis Tool



Conclusion

Today's larger FPGAs are quickly approaching 1M LEs, which can make timing closure a difficult process in extremely large designs. Good coding practices such as retiming logic, pipelining where practical, and avoiding high fan-out nodes can make the problem much easier. In addition, partitioning the design and good clock management at the start will go a long ways towards avoiding issues once the design is complete. However, these suggestions are not all inclusive and may not solve all timing issues, Altera has developed advanced tools and has highly trained support personnel who are ready and committed to help the user close timing in their designs.

Further Information

- Arria V FPGAs: Balance of Cost, Performance, and Power: www.altera.com/devices/fpga/arria-fpgas/arria-v/arrv-index.jsp
- Cyclone V FPGAs: Lowest System Cost and Power: www.altera.com/devices/fpga/cyclone-v-fpgas/cyv-index.jsp
- Quartus II Subscription Edition Software: www.altera.com/products/software/quartus-ii/subscription-edition/qts-se-index.html
- Webcast: "Optimize Your 28-nm FPGA Design for Maximum Performance": www.altera.com/education/webcasts/all/wc-2011-optimize-28nm-fpga-max-performance.html

Acknowledgements

- Trung Tran, Staff Product Marketing Manager, High-Density Products, Altera Corporation

Document Revision History

Table 2 shows the revision history for this document.

Table 2. Document Revision History

Date	Version	Changes
November 2011	1.0	Initial release.