WP496 (v1.0.1) August 8, 2017

Measuring Device Performance and Utilization: A Competitive Overview

By: Frederic Rivoallon

Across numerous applications and markets, the UltraScale architecture offers a portfolio of devices with fully connectable independent logic and flexible interconnect, delivering both high performance and compact utilization.

ABSTRACT

Coupled with the Vivado® Design Suite, the UltraScale™ architecture delivers high clock-rate designs, even at nearly full device utilization.

This white paper compares actual Kintex® UltraScale FPGA results to Intel's (formerly Altera) Arria 10, based on publicly available OpenCores designs. For each of the designs, detailed graphs that show performance and utilization are provided. The data confirms that the Xilinx device is two speed grades faster while enabling 20% more design content.

Introduction

Verifiable results based on OpenCores designs demonstrate that the Xilinx® UltraScale architecture delivers a two-speed-grade performance boost over competing devices while implementing 20% more design content. This boost equates to a generation leap over the closest competitive offering.

UltraScale Architecture and High-Level Design Software Tools

Xilinx's 16nm and 20nm families are based on the first All Programmable UltraScale architecture to span multiple nodes from planar through FinFET technologies and beyond, while also scaling from monolithic through 3D ICs. At 20nm, Xilinx pioneered the first ASIC-class All Programmable architecture to enable multi-hundred gigabit-per-second levels of system performance. At 16nm, UltraScale+™ families combine new memory, 3D-on-3D, and multi-processing SoC (MPSoC) technologies.

The Vivado Design Suite, on its own, offers an SoC-strength, IP- and system-centric development environment. Coupling this powerful design platform with the flexibility and targeted power of the UltraScale architecture identifies, readily addresses, and removes complex productivity bottlenecks in system-level integration and implementation.

Most Efficient Device Density

Compared to competing devices, the UltraScale architecture, in conjunction with the Vivado tool suite, makes it possible to leverage the device logic to its fullest potential. This directly translates into significant cost and power savings.

Architecture and software tools need to work symbiotically to yield the most efficient results. Failing to adhere to this principle can result in great losses in design efficiency. This is illustrated in Figure 1, which shows the competitor's device ceasing to accept more logic, *even though its LUT utilization is still quite low*.

When comparing devices, the UltraScale architecture can pack more logic through the Vivado tool suite's advanced algorithms. On average and across all designs, the UltraScale device is able to utilize 86% of LUTs at maximum device utilization—versus just 65% of LUTs for the competing device.

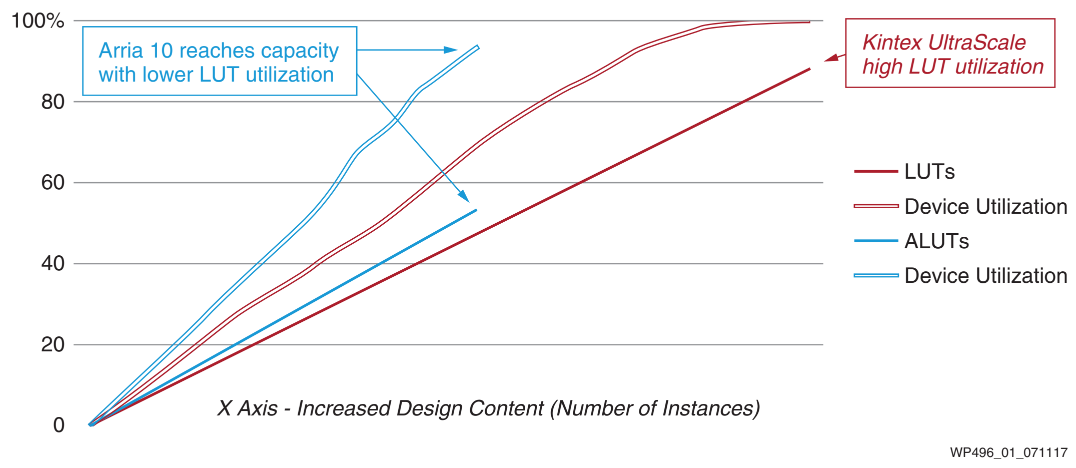

Figure 1: **Typical Device and LUT Percent Utilization vs. Increased Design Content**

Vivado Tools Push Device Utilization Higher

The UltraScale architecture offers truly independent LUTs, which can be routed at very high rates of utilization using Vivado tools. In a benchmark where design content is iteratively added, the Vivado tool can achieve 99% LUT utilization, place and route the design, and meet timing! By contrast, the competing device systematically falls short of full device utilization; the place-and-route tool fails to implement a design long before being able to use all LUTs in the device.

Better LUT Utilization through Better Architecture

It is not surprising that the competitor's LUTs can rarely be used at a satisfactory level of utilization, because their physical cluster (adaptive logic module (ALM)) has several connectivity restrictions within it that often result in using only one LUT in the cluster, leaving the other one unusable.

In Arria 10, the physical cluster (ALM) is able to host two 6-input LUTs (LUT6). However, these can only be packed together in a single ALM under connectivity restrictions that are often not realizable, and thus often require an additional ALM. With the Kintex UltraScale device, the LUT6 has its own inputs and can implement logic independently of any other LUT within the physical cluster. See Figure 2.

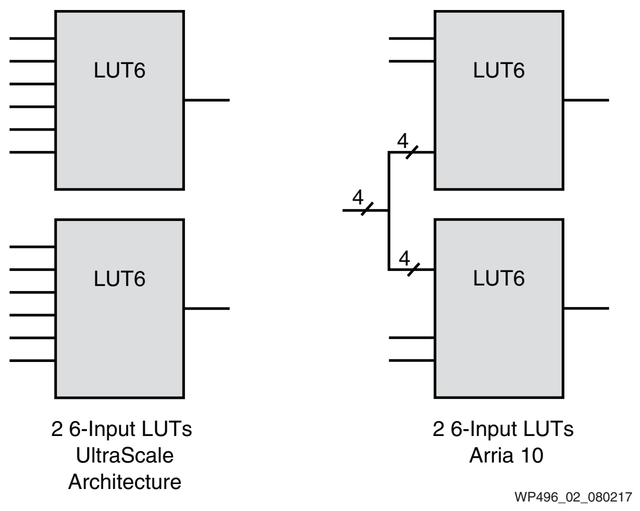

Figure 2: **LUT Architecture Differences**

In Figure 2, any two 6-input LUTs in the Kintex UltraScale device are truly independent. In Arria 10, two 6-input LUTs can only coexist in an ALM if they share four inputs.

The Vivado place-and-route tool technology and UltraScale architecture have been designed to handle dense and challenging designs and can reach high levels of LUT utilization, enabling the user to put more logic into the device. For large devices with high logic utilization, the Vivado tool's implementation engine ensures that the results can be as good as with a lower utilized device. The Vivado tools can better maintain performance and provide more consistent results from one run to the next when more instances are added to the design.

Robust Performance with Predictable Results

The Vivado Design Suite provides state of the art place-and-route algorithms for push-button performance. In current and future technologies (20nm and below), interconnect tends to be the primary bottleneck. The Vivado place-and-route tool delivers more predictable design closure by concurrently optimizing for multiple variables: timing, interconnect usage, and wire length. Considering a set of OpenCores designs, the performance advantage of UltraScale architecture is measured to be, on average, 25% faster (F_{MAX}) than an Arria 10 device (see individual data points in Figure 3).

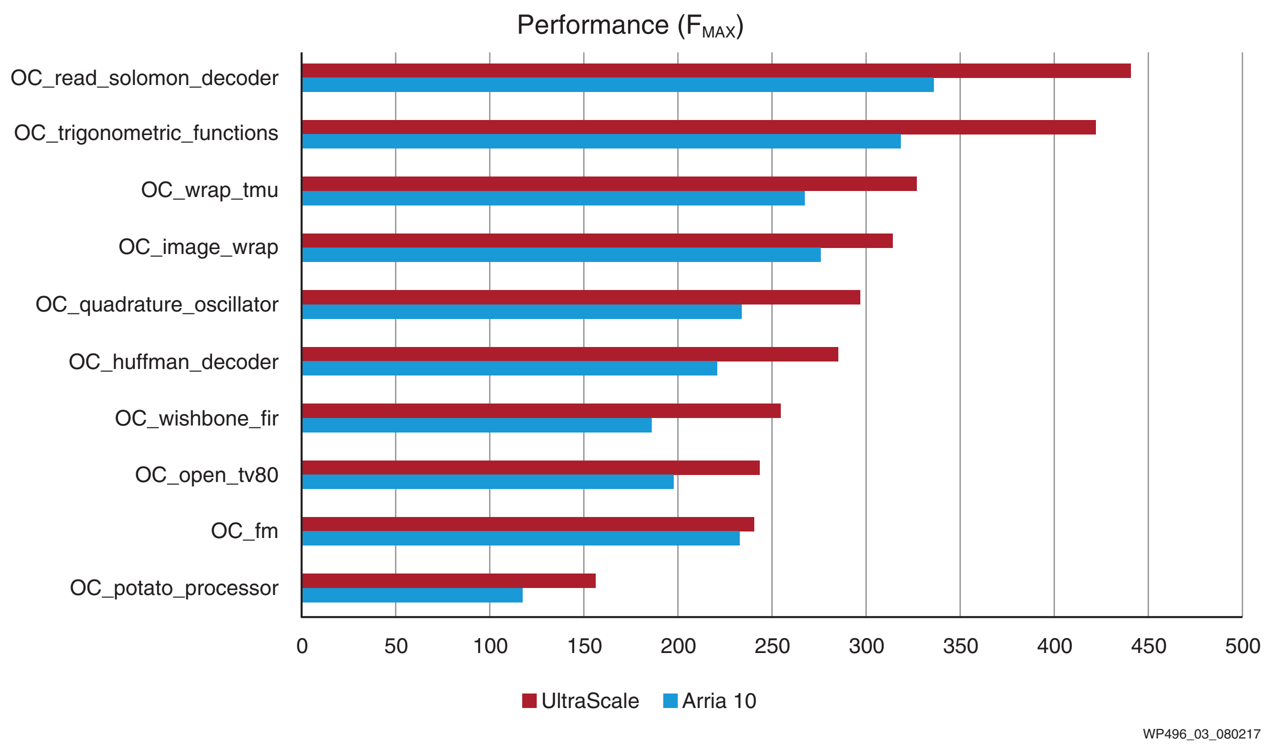

Figure 3: **F_{MAX} for the OpenCores Designs as Measured on One Instance**

OpenCores Designs

Benchmark Setup

The data was generated for Xilinx’s 20nm devices to align with the most advanced node publicly available in the competitor's software tools (as of June 2017). Xilinx selected OpenCores designs that represent current market needs and exercise a variety of device resource types (i.e., logic, RAM, and DSP). The selected devices were the Kintex UltraScale XCKU115-FLVF1924-3 FPGA and the Arria 10 10AX115U1F45E1SG, both the fastest speed grades available in their respective software packages: Vivado Design Suite 2017.1 and Quartus® Prime v16.1. The designs were then stamped repeatedly to gradually fill up the devices until the software could no longer implement the increased amount of logic. The experiments were conducted using the software tools’ default options and tight timing constraints. A top wrapper with shift registers was used to reduce I/O count to just one top-level input port and one top-level output port per stamp. All instances were connected such that each core's clock and reset nets were mapped to global buffers shared across all the stamped cores, ensuring that all wrapper logic remained free from performance bottlenecks (see Figure 4).

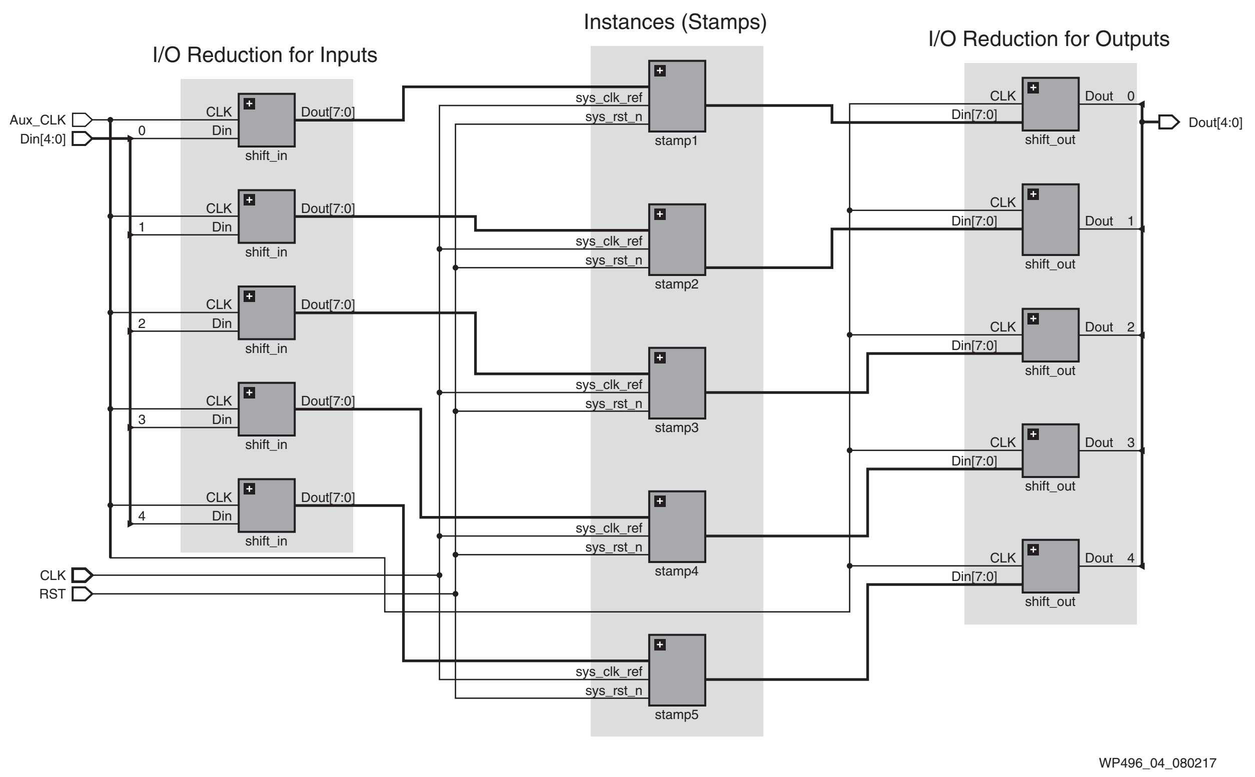

Figure 4: **Mechanism to Reduce I/O Count (Example with Five Stamps Shown)**

Utilization and Performance Tables for Each Design

Table 1 lists the OpenCores designs used in this comparison and links to the www.opencores.org website where users can learn more about each design and download the HDL source code. Table 1 shows the maximum percentage LUT utilization and the actual stamped instance count that was reached. A higher stamped instance count reflects a greater amount of design content being successfully fit into the device. The rightmost column displays the ratio for the most stamped instances implemented between the two devices.

The results show that the Vivado tools implement most designs with a high LUT utilization percentage, up to 99%. For Quartus Prime, the average LUT utilization is 65%—which helps explain how the UltraScale device packs more stamps for all these designs.

Table 1: **Open Cores Designs Used in Benchmark Comparison**

#	OpenCores Design Name	Design Function	Max LUT Utilization / Max # of Instances		UltraScale Area Advantage
			UltraScale	Arria 10	
1	OC_reed_solomon_decoder	Error correction	95% / 200	75% / 183	+9%
	URL: https://opencores.org/project,reed_solomon_decoder				
2	OC_wishbone_fir	Finite impulse response filter	99% / 80	62% / 45	+63%
	URL: https://opencores.org				
3	OC_huffman_decoder	Data compression	69% / 360	44% / 120	+200%
	URL: https://opencores.org/project,huffmandecoder				
4	OC_quadrature_oscillator	Telecom	90% / 565	76% / 85	+564%
	URL: See Note 1				
5	OC_image_warp	Image processing	83% / 230	65% / 190	+8%
	URL: https://opencores.org/project,warp				
6	OC_tmu	Video synthesis	86% / 230	66% / 197	+17%
	URL: See Note 1				
7	OC_potato_processor	Processor	88% / 240	53% / 130	+84%
	URL: https://opencores.org/project,potato https://github.com/skordal/potato				
8	OC_open_tv80	Microcontroller	97% / 125	72% / 120	+4%
	URL: https://opencores.org/project,tv80				
9	OC_trigonometric_functions	Math functions	69% / 360	80% / 190	+89%
	URL: https://opencores.org/project,trigonometric_functions_in_double_fpu				
10	OC_fm	Radio design	95% / 250	70% / 140	+78%
	URL: https://opencores.org/project,simple_fm_receiver				
Average LUT % (geometric mean):			**86%**	**65%**	

Notes:

1. This design is no longer linked from the OpenCores website, but can be provided upon request.

Table 2 displays the performance information for each of these OpenCores designs. The rightmost column computes the delta of the average F_{MAX} performance for the UltraScale device vs. the competitor's, based on all the runs that successfully produced a result for both devices (see Detailed OpenCores Compilation Results to view a plot of all the individual performance numbers).

Table 2: **Performance Information**

#	OpenCores Design Name	Design Function	F_{MAX} for One Core		UltraScale F_{MAX} Advantage (Geomean across Stamped Runs)
			UltraScale	Arria 10	
1	OC_reed_solomon_decoder	Error correction	441MHz	336MHz	+13%
2	OC_wishbone_fir	Finite impulse response filter	254MHz	186MHz	+35%
3	OC_huffman_decoder	Data compression	285MHz	221MHz	+18%
4	OC_quadrature_oscillator	Telecom / modulation	297MHz	234MHz	+35%
5	OC_image_warp	Image processing	314MHz	276MHz	+8%
6	OC_tmu	Video synthesis	327MHz	227MHz	+8%
7	OC_potato_processor	Processor	156MHz	117MHz	+23%
8	OC_open_tv80	Microcontroller	243MHz	198MHz	+6%
9	OC_trigonometric_functions	Math functions	422MHz	318MHz	+61%
10	OC_fm	Radio design	241MHz	233MHz	+0%

Detailed OpenCores Compilation Results

This section shows all the individual data points measured in the benchmark. For each design, two graphs are provided: one for performance and the other for device and LUT utilization, with actual design content measured by the number of stamped instances on the X axis. For the latter, two lines are drawn, one for the device utilization (percentage of configurable logic blocks (CLBs) or ALMs) and a second line for LUT percentage utilization. The LUT utilization line is always below device utilization, since it is a finer granularity metric.

OC_reed_solomon_decoder

Analysis: The design fills up to 200 instances in the UltraScale device, with a final LUT utilization of 95%. As shown on the utilization graph (bottom right in Figure 5), the Vivado tools start packing the logic more tightly into the UltraScale device's CLBs until the LUT count gets critically high.

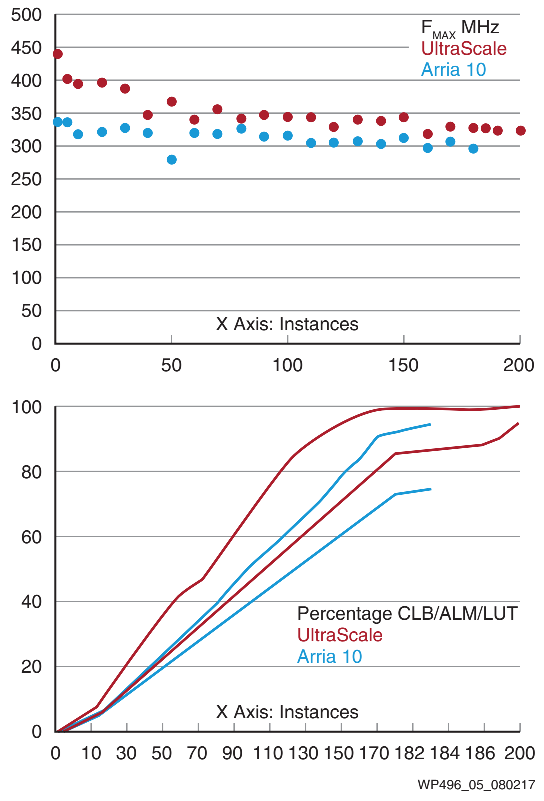

Figure 5: **OC_reed_solomon_decoder Results**

The LUT utilization in Arria 10 peaks at 75%; Quartus can no longer accept more logic beyond that point.

For all instances with possible direct comparisons, the F_{MAX} is higher for the UltraScale device compared to Arria 10. The performance in the UltraScale device remains above 300MHz, even at the 95% LUT utilization design.

OC_wishbone_fir

Analysis: A very significant difference is revealed in the number of instances that can be packed into the devices. Quartus reports early on that it is running out of ALMs with which to implement the logic. On the other hand, LUT utilization in the UltraScale device reaches 99.5%, compared to 62% for the competition. See Figure 6.

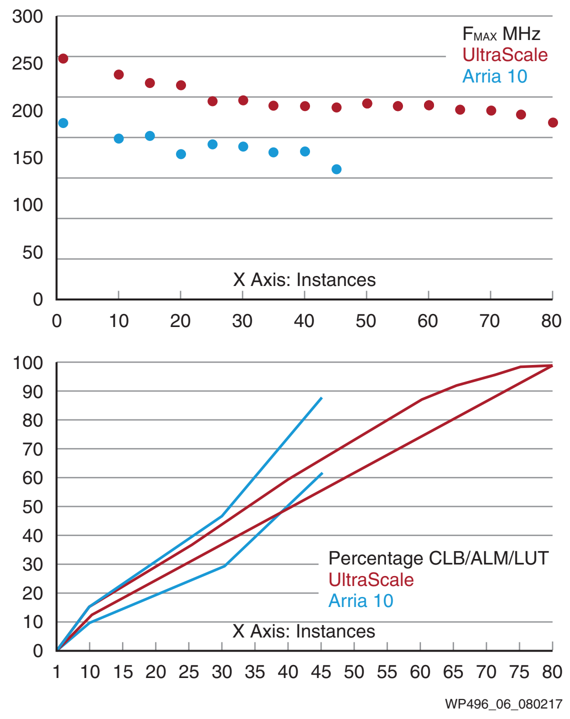

Figure 6: **OC_wishbone_fir Results**

The UltraScale device also shows better performance. Even when the UltraScale device is filled (eighty instances), it runs at a better F_{MAX} than the Arria 10 with only one stamp.

OC_huffman_decoder

Analysis: There is a very significant difference in the number of instances that can be placed and routed in each device. The UltraScale device can hold 3X more instances than Arria 10 (360 versus 120). On the last run that can fit into the Arria 10 device, its ALM utilization is 90% but with only 44% of LUTs used, which constitutes the largest measured gap in this series of runs. See Figure 7.

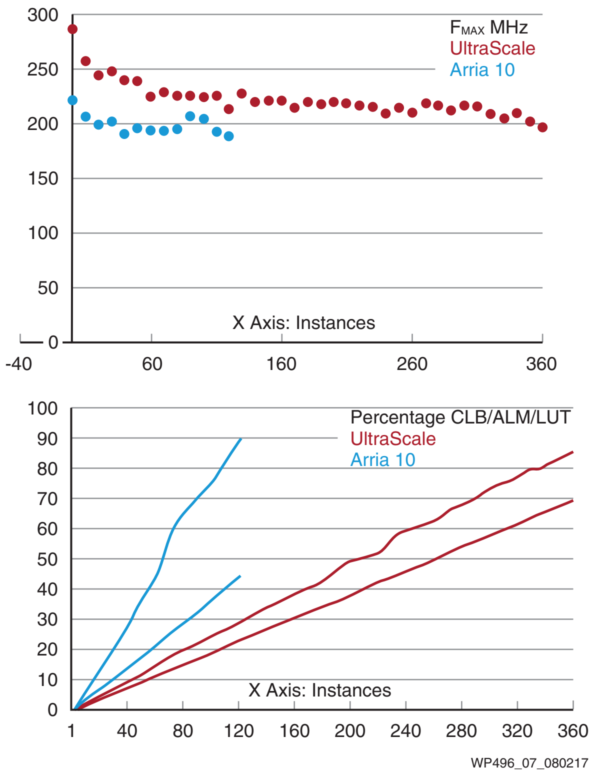

Figure 7: **OC_huffman_decoder Results**

In terms of performance, the UltraScale device has the advantage and performs better on each individual run for which the competing device can produce a result.

OC_quadrature_oscillator

Analysis: This VHDL design shows both better F_{MAX} and better logic capacity for the UltraScale device. The design uses only one block of memory per instance. It does not affect the timing-critical path, which is pure logic interconnect with a mix of LUTs and carry chains (used for arithmetic operations). See Figure 8.

These results show a dramatic difference between the two devices because one of the memories described in VHDL is implemented as a single block of RAM for the Kintex UltraScale device. The competitive device maps that memory onto LUTs and flip-flops, which explains why each stamp for Arria 10 device uses so many resources.

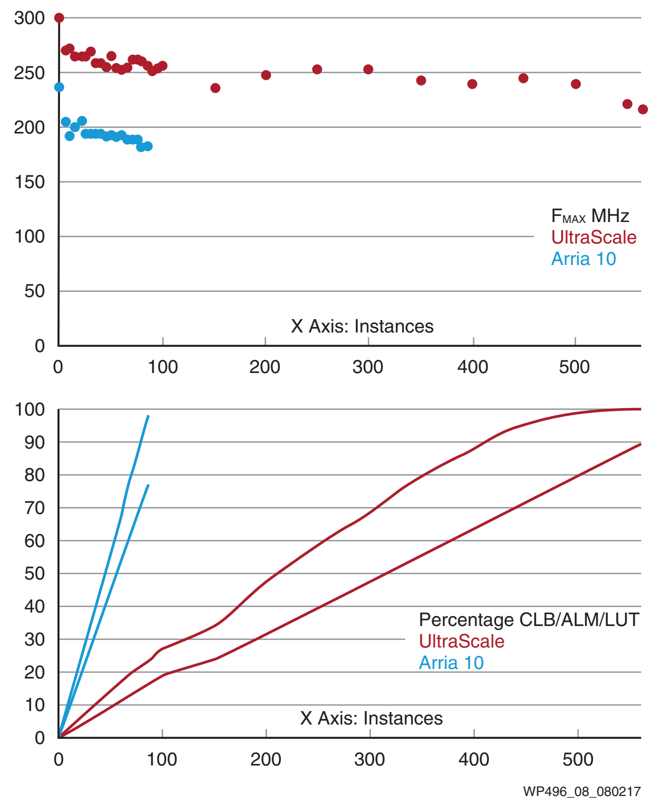

Figure 8: **OC_quadrature_oscillator Results**

OC_image_warp

Analysis: Maximum LUT utilization is 83% for the UltraScale device versus 65% for Arria 10. The design also uses integrated DSP and block RAM blocks. See Figure 9.

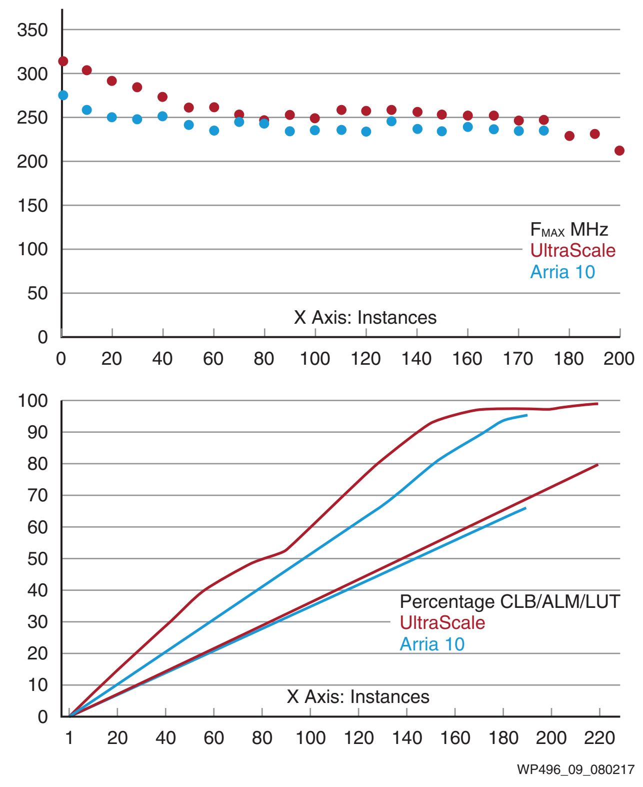

Figure 9: **OC_image_warp Results**

The timing-critical paths span from a block RAM to a DSP48 cell and across several levels of LUTs in between. In terms of results, here again the UltraScale device produces a better F_{MAX} for all possible matchups.

OC_tmu

Analysis: The UltraScale device has the clear advantage in terms of percentage of LUT utilization and instances fitted with 86% (230 stamps) for the UltraScale device versus 66% (197 stamps) for Arria 10. Note that this design also involves integrated block RAM and DSP. See Figure 10.

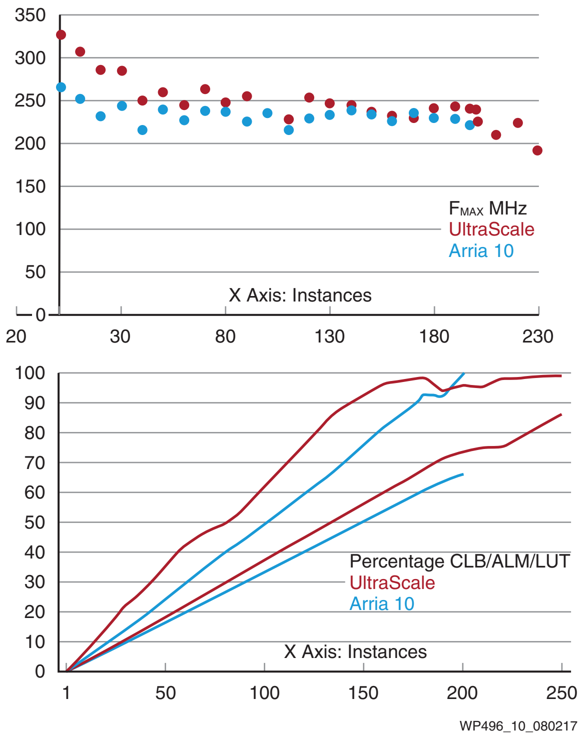

Figure 10: **OC_tmu Results**

The most critical paths are based on interconnect and LUTs and are contained within the “edge trace” level of hierarchy.

OC_potato_processor

Analysis: The maximum number of instances that can be placed and routed for the UltraScale device is 240 versus 130 for Arria 10. The highest percentage LUT count reached for UltraScale is 88% versus only 53% for Arria 10. The design leverages both the block RAM and the distributed RAM (CLB based) in the UltraScale device. See Figure 11.

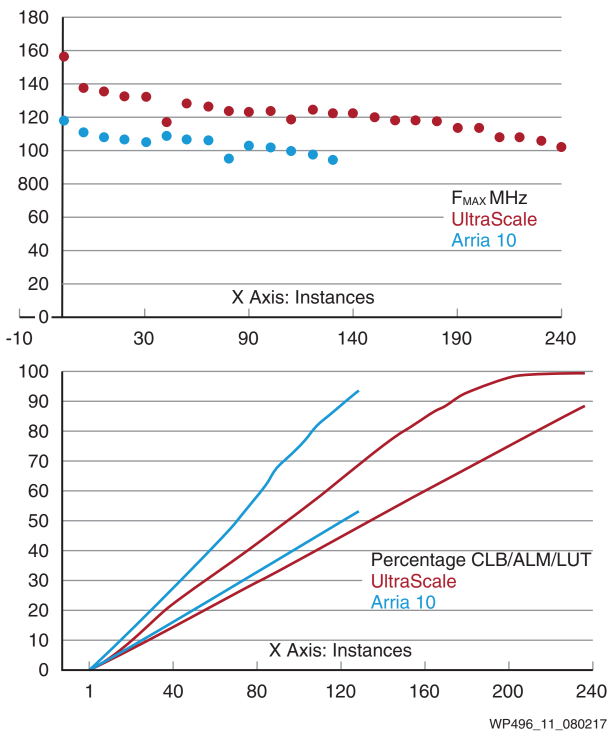

Figure 11: **OC_potato_processor Results**

As commonly seen in processor designs, the critical path has many levels of logic (21), which yield relatively slow F_{MAX} numbers. However, the UltraScale device is always above 100MHz across all runs and is always faster than Arria 10 for all possible match-ups.

OC_tv80

Analysis: Both performance and area are closer between the two devices, with still an advantage for the UltraScale devices in both. The design does not include block RAM or DSP blocks, and the critical path is about 13 levels of logic. See Figure 12.

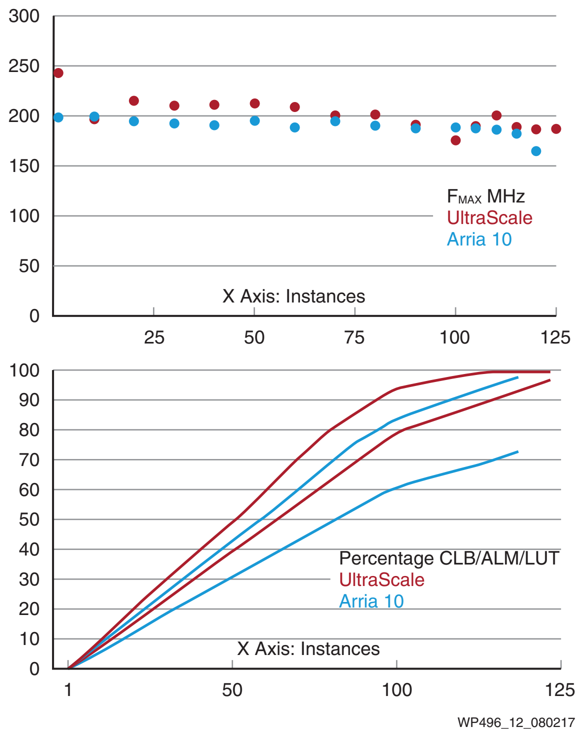

Figure 12: **OC_tv80 Results**

The utilization graph shows that the CLB count reaches close to 100% for 110 instances, but the Vivado tool manages to control the clustering of logic to implement more instances.

OC_trigonometric_functions

Analysis: This design is pure interconnect with a lot of fully utilized LUT6 in the UltraScale devices. This explains how the utilization can be so much better than for Arria 10, since the architecture of the latter cannot efficiently use 6-input LUT functions without running out of clusters quickly. It is visible on the area utilization graph that Arria 10 reaches 100% ALM very early and cannot implement 200 instances, while UltraScale easily passes 300 instances. See Figure 13.

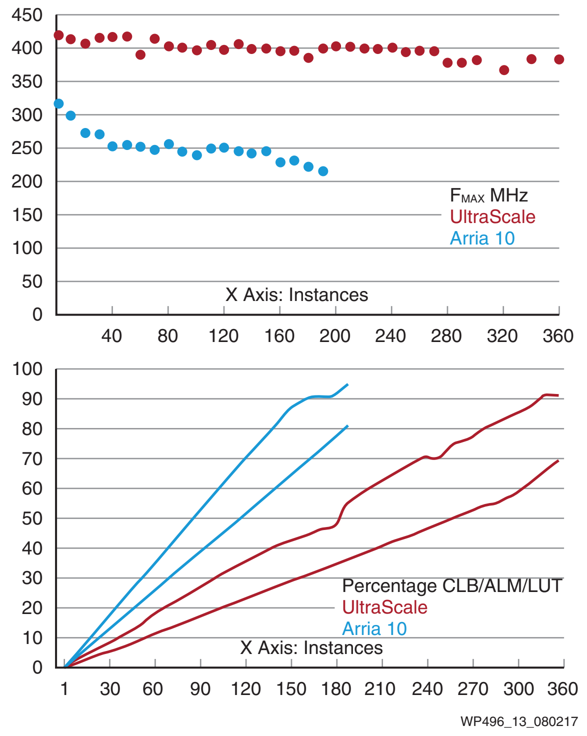

Figure 13: **OC_trigonometric_functions Results**

In terms of performance, there is also a clear difference: In a design requiring five levels of logic (like this one), the UltraScale device can reach over 400MHz—and is at times about 150MHz over the competition's result.

OC_fm

Analysis: This design has many adder trees that the Vivado tool implements very efficiently, leveraging the two outputs of the LUT6 in the UltraScale device. This results in a high-performance design with very compact logic. See Figure 14.

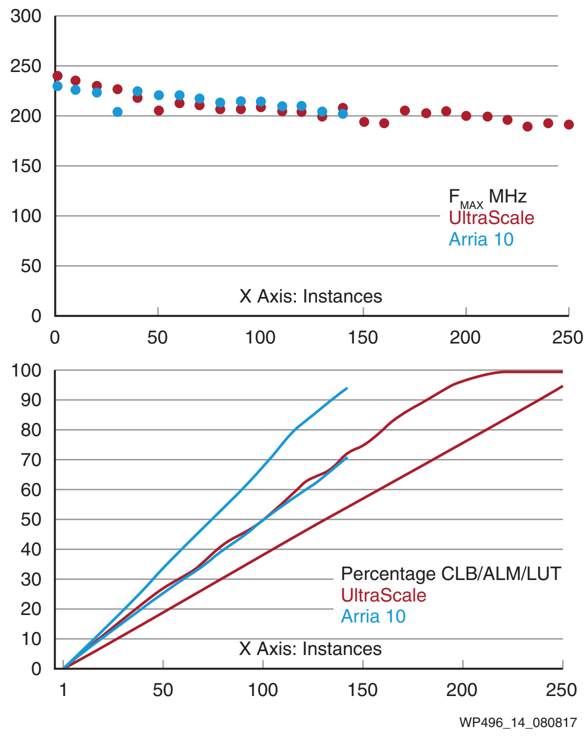

Figure 14: **OC_fm Results**

The UltraScale device implements over 100 more stamps than Arria 10.

Conclusion

Across OpenCores designs gradually stamped to fill devices to their maximum, the Kintex UltraScale FPGA's LUT utilization ranges from 69% to 99% compared to 44% to 80% for Arria 10. The higher LUT utilization allows more design content to fit into the Xilinx device. Conversely, the Arria 10 limited cluster connectivity yields lower LUT utilization and fits less content.

The Kintex UltraScale device also shows higher performance (F_{MAX}), which in some cases exceeds two speedgrades, while better utilizing the available resources with 20% more actual design content: OC_wishbone_fir, OC_quadrature_oscillator, and OC_trigonometric_functions (see Table 2).

Through a superior circuit architecture with unbounded LUT connectivity, coupled with advanced software tools, Xilinx offers a winning unparalleled combination to craft highly efficient designs.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
08/08/2017	1.0.1	Typographical edits.
08/07/2017	1.0	Initial Xilinx release.

Disclaimer

Automotive Applications Disclaimer