WHITE PAPER

FPGA

Intel® Arria® 10 FPGA Performance Benchmarking Methodology and Results

Intel Arria 10 FPGAs deliver more than a speed grade faster core performance and up to a 20% f_{MAX} advantage for publicly available OpenCore designs.†

Authors

Martin S. Won
Senior Member of Technical Staff
Intel Programmable Solutions Group

Madhu Monga
Applications Engineer
Intel Programmable Solutions Group

Table of Contents

Introduction

This paper presents a rigorous methodology for evaluating and benchmarking the core performance of the Intel® Arria® 10 FPGA programmable logic product family, with the goal of transparently presenting the methods and data such that any interested party can reproduce and analyze the results.† To this end, ten publicly-available designs from OpenCores representing a variety of functions were implemented in a device from the Intel Arria 10 FPGA family and a device from the closest competitor: the Xilinx* UltraScale* family. The benchmark results show that Intel Arria 10 FPGAs deliver up to 20% higher performance than Xilinx UltraScale devices, as measured by the maximum clock frequencies achieved in the example designs across a range of device utilization.†

Background: evaluating the performance of Intel FPGA and SoC products

The programmable logic industry does not have a standard benchmarking methodology. Therefore, Intel employs rigorous internal analysis using a broad combination of customer and internally-generated designs to understand and quantify the performance of its programmable logic products relative to prior-generation Intel products and competing products. The designs are collected from a variety of market segments, such as high-performance computing, image and video processing, wired and wireless communications, and consumer products. Additionally, the designs use a variety of implementation technologies including ASICs and FPGAs from other vendors. By using a broad suite of designs, Intel ensures that the results are accurate and representative of the complex interaction of customer designs and FPGA design tools such as the Intel® Quartus® Prime Software Suite. To use customer designs, Intel invests significant resources in converting designs to work with various synthesis tools and electronic design automation (EDA) vendors. Intel also ensures that functionality is preserved and appropriate code optimizations for the specific FPGA vendor are made (which is necessary because designs are often developed such that they are optimized for a specific FPGA).

For performance comparisons, Intel employs at least two comparison types: timing-constrained, and best effort. The effort level is a reflection of both the user's effort (how much work the user must do) and the tool's effort (how much total compile time is required) to obtain the results. The purpose of the timing-constrained comparison is to use default EDA tool settings, but ensure that the EDA tool is optimizing for performance. The purpose of the best effort comparison is to give an indication of the best possible result achievable. The experiments for this kind of comparison require longer individual compilation times than in a default push-button compile, and more than one compile per design.

Using this methodology, Intel has determined that Intel Arria 10 FPGAs and SoCs deliver a core performance advantage over competitive 20 nm FPGA products, as measured by the maximum f_{MAX} achievable for the speed-critical clock paths in each of the designs in the design suite. This performance advantage ranges from around 10% - 20% depending on the design, which equates roughly to an advantage of one or two speed grades, where speed grades are typically defined as a performance difference of 10% - 15%.† These results help validate the position of the Intel Arria 10 FPGAs as the highest performance 20 nm FPGA family. However, because these results were obtained using customer and Intel proprietary designs, Intel can share only limited details of the analysis, which ultimately limits the usefulness of this information to programmable logic users.

Increasing transparency via OpenCore-based performance comparisons

To address this challenge to understanding programmable logic performance, Intel has undertaken a benchmarking effort using publicly-available designs from OpenCores (www.opencores.org), an organization that offers open-source hardware intellectual property (IP)cores. The goal of this benchmarking effort is to help programmable logic users:

- Understand the exact designs used in the performance evaluation, including the specific details of those designs down to the register transfer level (RTL) description
- Reproduce the results of the analysis themselves
- Scrutinize the results of the analysis to better understand the applicability of the Intel Arria 10 FPGA performance advantage to their specific design

The scope of this OpenCores-based performance analysis is narrower than the internal analysis that Intel employs because it focuses specifically on timing-constrained compilations. This analysis is not a comprehensive treatment of the topic, but the results and conclusions provide insight into the relative performance of Intel Arria 10 FPGAs compared to competitive devices when implementing similar designs or designs that are composed of functions similar to the ones used in the design example suite.

Target device families for performance comparison

Intel chose its Intel Arria 10 FPGA family and the Xilinx UltraScale family for the performance analysis. These products are both built on a 20 nm TSMC process. Intel chose the largest, comparably-sized devices using the fastest speed grades available:

- Intel Arria 10 Device: 10AX115U1F45E1SG
- Xilinx UltraScale Device: XCKU115FLVF19243E

Note: Smaller devices within the families obtain similar performance results. According to Intel internal tests, Kintex* UltraScale and Virtex* UltraScale devices of the same speed grade exhibit similar performance levels, therefore, the fastest speed grade device from either family can represent the entire UltraScale offering.

OpenCore designs used in the analysis

Intel selected OpenCores based on design size and complexity, with the intent of representing a wide variety of function types that use a mix of different device resource types (i.e., logic, RAM, and DSP). Table 1 lists the OpenCore designs used in the performance comparison and links to the www.opencores.org web page for each design where users can learn more about the design and download it. The table also shows the average amount of logic utilized by each of the OpenCore designs, measured using:

- Adaptive logic modules (ALMs) for Intel Arria 10 FPGAs
- Configurable logic block look-up tables (CLB LUTs) for UltraScale FPGAs

Note: ALMs and CLB LUTs have architectural differences (ALMs use look-up tables with eight inputs and CLBs use look-up tables with six inputs), therefore, the devices are expected to have different utilization numbers for a given design.

OpenCores stamping methodology

The OpenCores designs use only a small fraction of the resources in the target devices. Utilizing only a small fraction of the total device resources is not a common practice or desired goal among programmable logic users. Also, increasing utilization often has a negative impact on the highest achievable f_{MAX} as device resources become exhausted and the design becomes harder to place and route. To simulate the impact of device utilization on programmable logic performance, Intel performed a large number of compilations, each one incrementally adding one OpenCore instance compared to the prior compilation. To increase the design size in the programmable logic device, each OpenCore design was instantiated repeatedly (multiple stamps of the same core) in the FPGA such that:

- Each stamp was implemented in parallel
- An I/O wrapper logic was added to reduce the number of I/O pins required for the larger design
- No timing critical paths between the cores and the wrapper logic existed
- The wrapper logic provided as little overhead as possible

Figure 1 illustrates the stamping process.

OpenCore stamping and benchmarking methodology

As the number of instantiations of the OpenCore design increases (and thus design size increases), resources such as I/O pins and global clocks become limited. To avoid running out of pins, each OpenCore was wrapped in a shift register, such that one physical pin would feed all input pins of a core and all output pins of a core would feed into a loadable shift register. Figures 2 and 3 show the input and output shift registers, respectively. The shift register size is dependent on the number of I/O pins, and the number of shift registers is dependent on the number of OpenCores implemented in the FPGA.

#	OpenCore Design Name	Design Function	Web Page	Device Utilization (Single Instance)	
				Intel Arria 10 FPGA ALMs Used	UltraScale CLB LUTs Used
1	oc_avr_hp_cm4	AVR Processor	https://opencores.org/project,avr_hp	1,269	1,706
2	oc_warp_tmu	Image Processing	https://opencores.org/project,warp	2,100	2,417
3	oc_reed_ solomon_decoder	Error Correction Code	https://opencores.org/project,reed_ solomon_decoder	2,208	3,299
4	oc_usbhostslave	USB 1.1 Controller	http://opencores.org/ project,usbhostslave	1,228	1,701
5	oc_dma_axi64	Single-channel 64 bit AXI* Master DMA	https://opencores.org/project,dma_axi	1,817	2,582
6	oc_256_aes	Advanced Encryption Standard (AES)	Note 1	1,101	2,220
7	oc_m1_core	32 bit RISC Processor	http://opencores.org/project,m1_ core,overview	1,797	2,617
8	oc_aquarius	RISC SuperH Processor	https://opencores.org/project,aquarius	2,703	3,457
9	oc_des_des3perf	Triple Data Encryption Standard (DES)	https://opencores.org/project,des	3,574	5,618
10	oc_fpu100	32 bit Floating Point Unit	https://opencores.org/project,fpu100	1,827	2,700

Table 1. Ten OpenCore Designs Used in the Performance Comparison

[1] The oc_256_aes design was available from OpenCores when the performance comparison project was started, however, it is now unavailable. The design is still available under its open source license from Intel at the links included at the end of this paper.

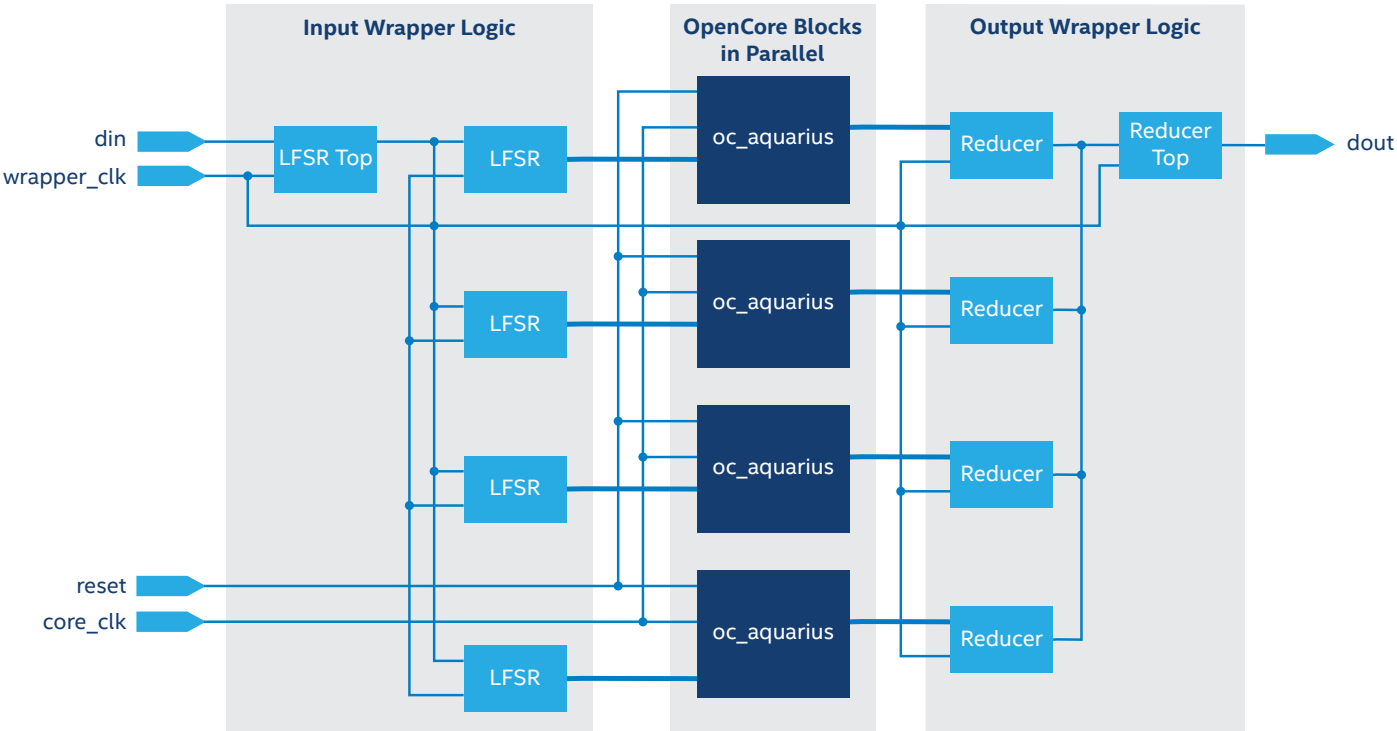

Figure 1. oc_aquarius Design Instantiated Four Times in the FPGA

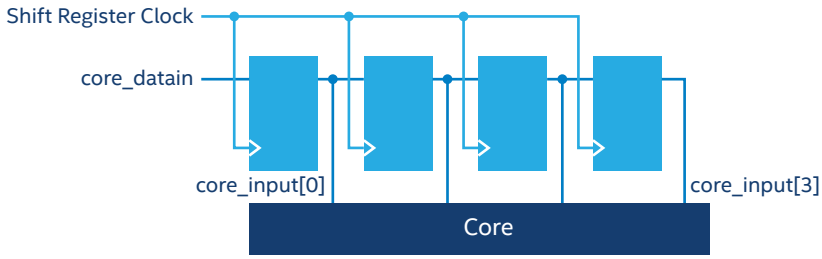

Figure 2. Input Shift Register Implementation

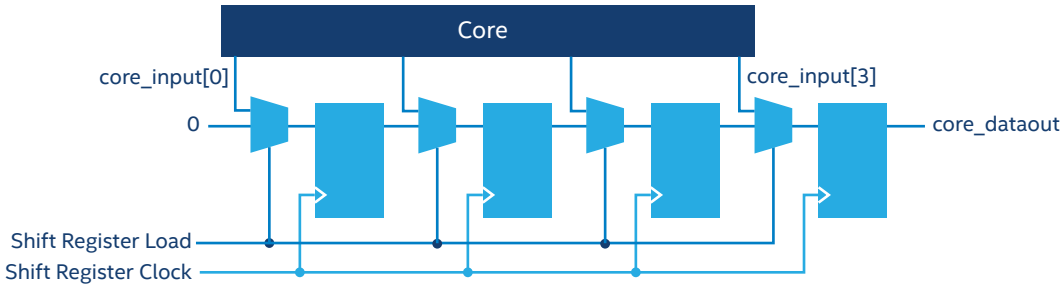

Figure 3. Output Shift Register Implementation

To avoid running out of global clock resources, a pin directly fed the global clock and reset signals for all OpenCores. For example, if a core required two clocks (core clock 1 and core clock 2) and one reset signal, all instances of core clock 1 were fed by one pin, all instances of core clock 2 were fed by a different pin, and all instances of the reset signal were fed by a third pin. With this method, all OpenCores were fed by the same clock and reset signals (see Figure 4).

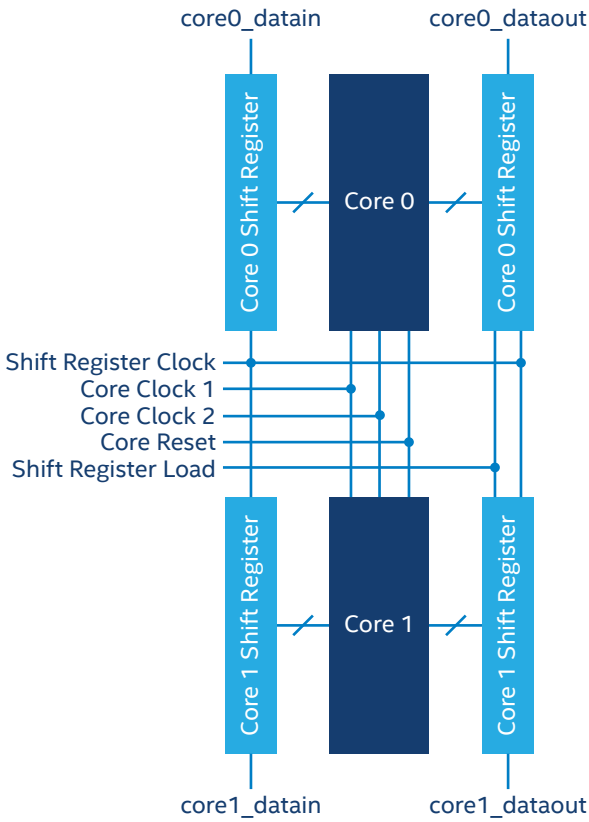

Figure 4. Two-core implementation with shared clock and reset signals

Once the wrapper logic tied up all of the OpenCores in the FPGA, Intel ensured that no critical paths existed between the wrapper logic (shift registers) and the OpenCore. To achieve this goal, false paths were created and, by making the core clock(s) and wrapper logic clock on different unrelated clock domains, no timing paths existed. The design tools could then optimize the cores separately from the shift registers. Intel instantiated the OpenCores as many times as the device and design tools would allow without compilation errors.

Software tools, settings, and constraints

To perform this study, Intel used the latest version of the required FPGA development tools that were available at the time of the analysis:

- Intel Quartus Prime Software Suite version 16.1 B189
- Xilinx Vivado* software version 2016.3

Both tools were installed and operated on Linux64 machines.

These programmable logic tools offer settings that provide a trade-off among design performance, logic resource consumption, compile time, and memory usage. The settings that produce the best results for one design are likely not the best for another. Additionally, user constraints that guide the EDA tool can improve the results. Even with a design set that is representative of customer designs, the benchmarking outcome varies significantly with software settings and applied constraints. For the performance comparisons presented in this paper, Intel used the default EDA settings, but set aggressive timing constraints. To determine aggressive timing constraints for each design, Intel applied a frequency (f_{MAX}) constraint to each OpenCore design clock such that the constraint is just beyond what is achievable for each clock. Intel determined a base constraint value by increasing the constraint until it could not be met. Then, Intel determined the aggressive constraint by multiplying the base constraint value by a factor of at least 1.3. The following sections describe the constraints applied to each design.

Individual OpenCore design compilation results

This section provides the detailed compilation results for each OpenCore design. In each case, a graph is provided of the f_{MAX} achieved for each design compilation.

- The vertical axis measures the f_{MAX} of the compilations
- The horizontal axis measures the device utilization, as measured by logic utilization (Intel Arria 10 FPGA ALMs and Ultrascale CLBs)
- Blue dots show the f_{MAX} values for Intel Arria 10 FPGA compilations
- Red dots show the f_{MAX} values for UltraScale compilations
- The geometric mean value of each set of data points is also shown (blue line for Intel Arria 10 FPGA, red line for UltraScale), providing a numerical value for the overall trend of the data points

The data points at the leftmost edge are for the compilations corresponding to device utilization starting at 40%,as measured by logic usage. The analysis uses 40% because most programmable logic users seek to utilize around half or more of their device resources, or select a smaller device to achieve lower cost. Also, as indicated in most of the graphs, the f_{MAX} values are quite stable for the majority of the device utilization and only tends to fall off when one of the resources (logic, memory, DSP, routing, etc.) becomes limited. The last data point for each set (the last successful compilation) is not shown or included in the calculation of the geometric mean. This value is not considered in the analysis because the last data point usually represents a compilation that has completely exhausted one of the device resources and indicates a design situation that would be considered unusable or unacceptable by many users.

OC_AVR_HP_CM4 core

This graph shows the f_{MAX} results for the compilations of the OC_AVR, an AVR processor core. The Intel Arria 10 FPGA f_{MAX} values fall off a little starting at around 90% utilization, producing a geometric mean f_{MAX} of 320 MHz. The Intel Arria 10 FPGA compilations fail at about 97% logic utilization, due to routing congestion. The UltraScale f_{MAX} values also fall off at about 90% utilization, producing a geometric mean f_{MAX} of 267 MHz. The last compilation is at about 97% logic utilization, and stops afterwards due SSI partitioning failure.

OC_Warp_TMU core

This graph shows the f_{MAX} results for the OC_Warp_TMU image processing design compilations. The Intel Arria 10 FPGA f_{MAX} values are stable at about 250 MHz, and only fall off slightly after 90% logic utilization, producing a geometric mean f_{MAX} of 248 MHz. The Intel Arria 10 FPGA compilations are successful until about 97% utilization, and fail afterwards due to routing congestion. The UltraScale f_{MAX} values begin to fall off at about 75% device utilization, and the design ceases to compile successfully at about 87% utilization due to routing congestion, producing a geometric mean f_{MAX} of 225 MHz.

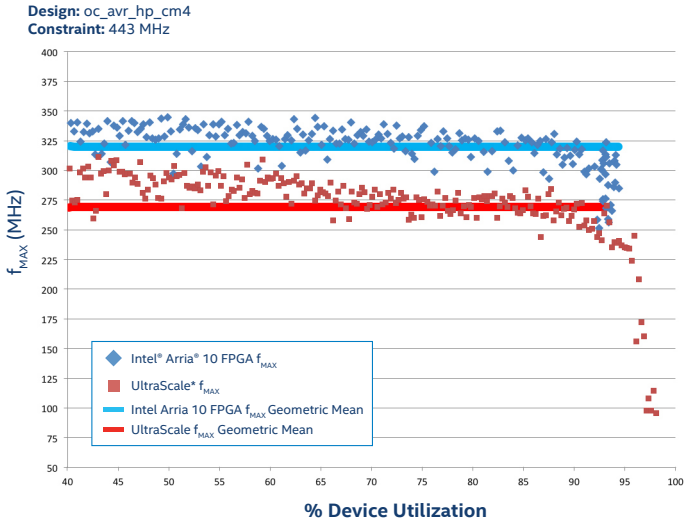

Figure 5. OC_AVR_HP_CM4 Results

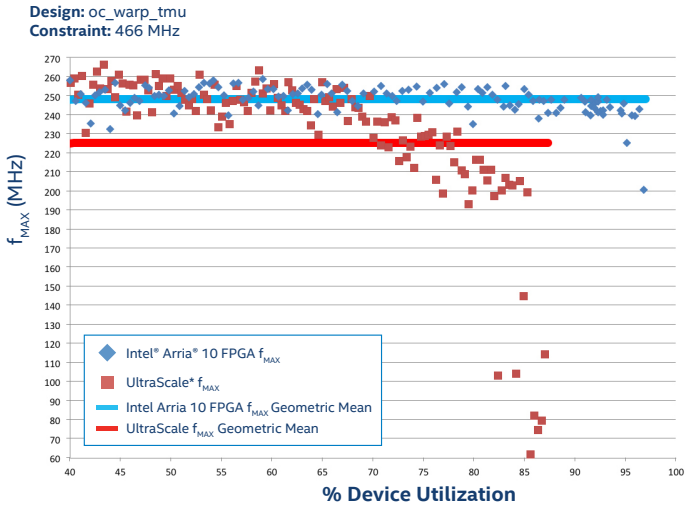

Figure 6. OC_Warp_TMU Results

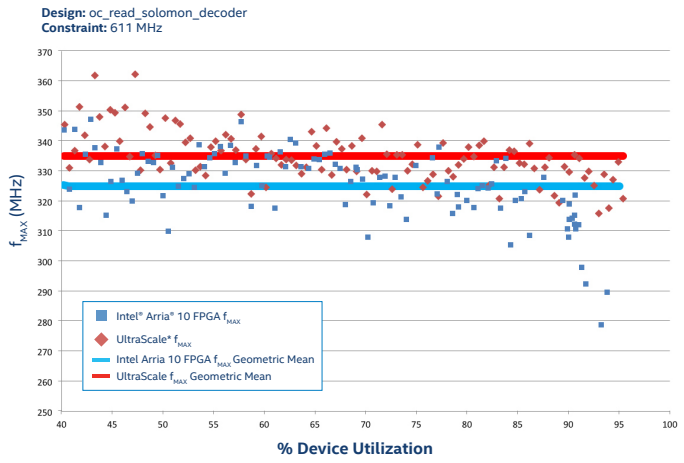

Figure 7. Reed-Solomon Decoder Results

OC_Reed-Solomon-Decoder core

This graph shows the f_{MAX} results for the compilations of the OC_Reed-Solomon_Decoder, a function commonly used for error correction. The Intel Arria 10 FPGA f_{MAX} values are quite

stable at about 330 MHz, and falls off starting at roughly 90% logic utilization due to exhausting the M20K memory resources, producing a geometric mean f_{MAX} of 325 MHz. The f_{MAX} fall off occurs due to exhaustion of the M20K memory resources, and is expected due to the Intel Quartus Prime Software Suite default compilation settings, which do not enable MLAB memory resources to be inferred. In this mode, the M20K resources are used for all memory needs, even when MLAB resources might suffice, and the M20K resources are completely exhausted at about 94% logic utilization. The UltraScale f_{MAX} values are stable at about 340 MHz, producing a geometric mean f_{MAX} of 335 MHz. The last successful compilations are at about 96% utilization, and fail afterwards due to insufficient logic.

OC_USBHostSlave core

This graph shows the f_{MAX} results for the compilations of the OC_USBHostSlave, a USB 1.1 controller. The Intel Arria 10 FPGA f_{MAX} values are quite stable and produce a geometric mean f_{MAX} of 480 MHz. The Intel Arria 10 FPGA compilations fail at about 78% logic utilization, again due to exhausting the M20K resources, similar to the Reed-Solomon case. The UltraScale f_{MAX} values are stable at about 425 MHz, producing a geometric mean f_{MAX} of 425 MHz. The last compilation is at about 96% logic utilization, and stops afterwards due to SSI partitioning failure.

OC_DMA_AXI64 core

This graph shows the f_{MAX} results for the compilations of the OC_DMA_AXI64, a single-channel 64 bit AXI master direct-memory access function. The Intel Arria 10 FPGA f_{MAX} values are quite stable start to fall off a little at 90%+ utilization, producing a geometric mean f_{MAX} of 331 MHz. The Intel Arria 10 FPGA compilations fail at nearly 100% logic utilization, due to routing congestion. The UltraScale f_{MAX} values are also quite stable and only fall off at 90%+ utilization, producing a geometric mean f_{MAX} of 283 MHz. The last compilation is at about 95% logic utilization, and stops afterwards due to SSI partitioning failure.

OC_256_AES core

This graph shows the f_{MAX} results for the compilations of the OC_256_AES, a 256 bit Advanced Encryption Standard (AES) function. The Intel Arria 10 FPGA f_{MAX} values fall off a little starting at 70% utilization, producing a geometric mean f_{MAX} of 242 MHz. The Intel Arria 10 FPGA compilations fail at nearly 100% logic utilization, due to routing congestion. The UltraScale f_{MAX} values are quite stable, producing a geometric mean f_{MAX} of 209 MHz. The last compilation is at nearly 100% logic utilization, and stops afterwards due insufficient LUTs.

OC_M1 core

This graph shows the f_{MAX} results for the compilations of OC_M1, a 32 bit processor core. The Intel Arria 10 FPGA f_{MAX} values are quite stable and begin to fall off a little starting at around 90% utilization, producing a geometric mean f_{MAX} of 277 MHz. The Intel Arria 10 FPGA compilations fail at about 99% logic utilization, due to insufficient LAB resources. The UltraScale f_{MAX} values also fall off at about 90% utilization, producing a geometric mean f_{MAX} of 258 MHz. The last compilation is at about 98% logic utilization, and stops afterwards due to insufficient CLBs.

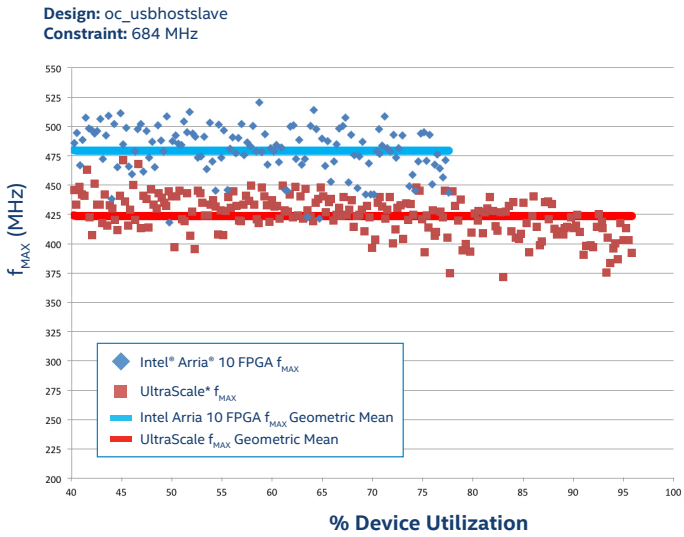

Figure 8. USB Host Slave Results

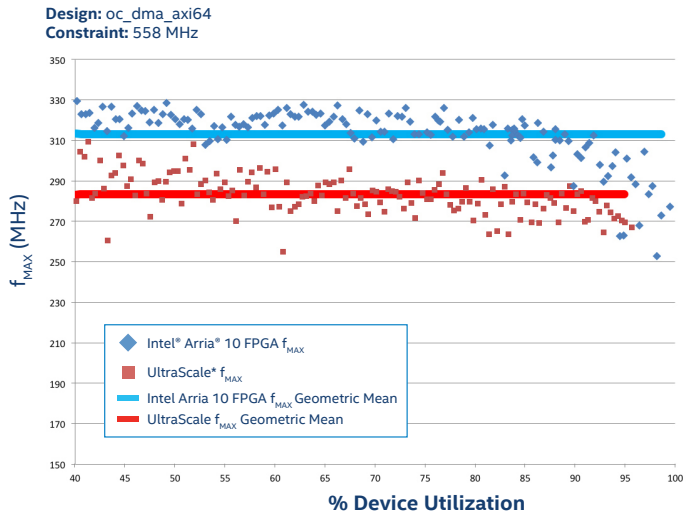

Figure 9. DMA AXI64 Results

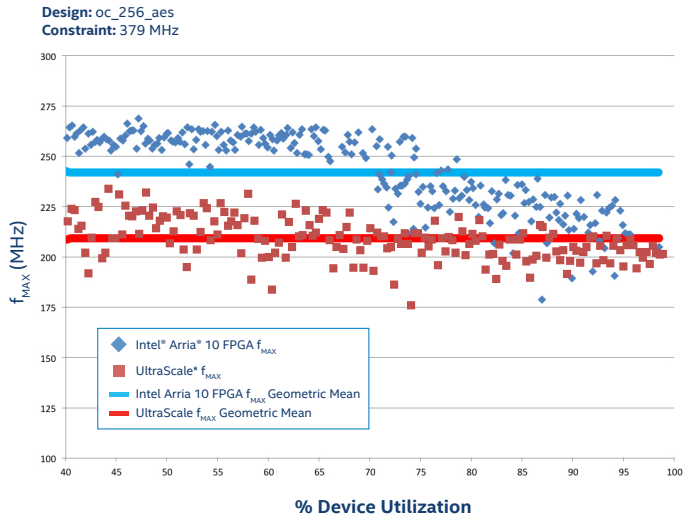

Figure 10. 256 AES Results

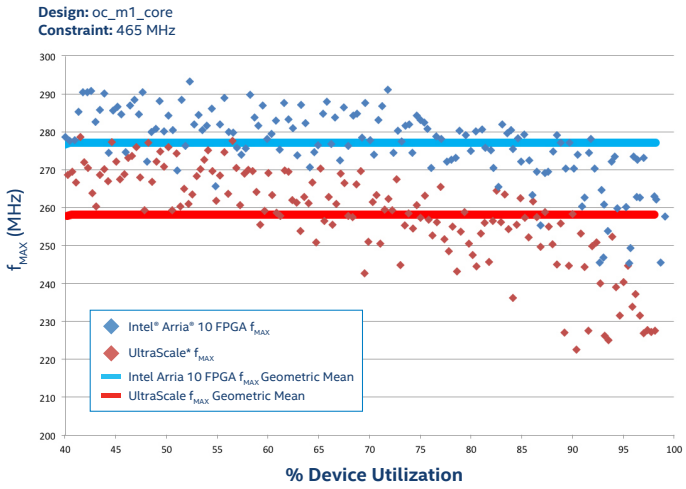

Figure 11. M1 (32 bit Processor) Results

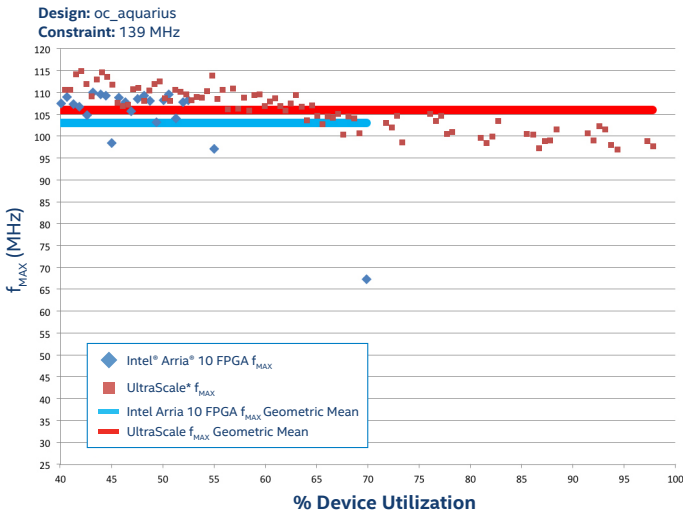

Figure 12. Aquarius Results

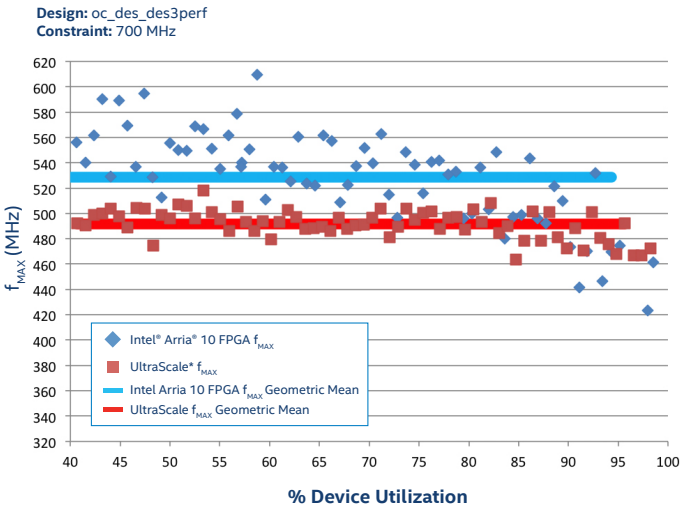

Figure 13. DES DES3Perf Results

OC_Aquarius core

This graph shows the f_{MAX} results for the compilations of OC_Aquarius, a RISC SuperH processor core. The Intel Arria 10 FPGA f_{MAX} values fall off at over 50%, producing a geometric mean f_{MAX} of 103 MHz. The Intel Arria 10 FPGA compilations fall off due to the default compiler settings exhausting the M20K resources, similar to the Reed-Solomon case, and all of the M20K resources are exhausted at the last indicated compile, which is at about 70% logic utilization. The UltraScale f_{MAX} values fall off slightly, producing a geometric mean f_{MAX} of 106 MHz. The last compilation is at about 99% logic utilization, and stops afterwards due to insufficient CLBs.

OC_DES_DES3Perf core

This graph shows the f_{MAX} results for the compilations of the OC_DES_DES3Perf, a Triple Data Encryption Standard (DES) function. The Intel Arria 10 FPGA f_{MAX} values are stable and start to fall off at 80%+ utilization, producing a geometric mean f_{MAX} of 528 MHz. The Intel Arria 10 FPGA compilations fail at about 98% logic utilization, due to routing congestion. The UltraScale f_{MAX} values are quite stable and only fall off a little at 90%+ utilization, producing a geometric mean f_{MAX} of 492 MHz. The last compilation is at about 98% logic utilization, and stops afterwards due to insufficient CLBs.

OC_FPU100 core

This graph shows the f_{MAX} results for the compilations of the OC_FPU100, a floating point unit function. The Intel Arria 10 FPGA f_{MAX} values are stable and start to fall off a little at 92%+ utilization, producing a geometric mean f_{MAX} of 254 MHz. The Intel Arria 10 FPGA compilations fail at nearly 100% logic utilization, due to insufficient LAB resources. The UltraScale f_{MAX} values are quite stable, producing a geometric mean f_{MAX} of 232 MHz. The last compilation is at about 94% logic utilization, and stops afterwards due to SSI partitioning failure.

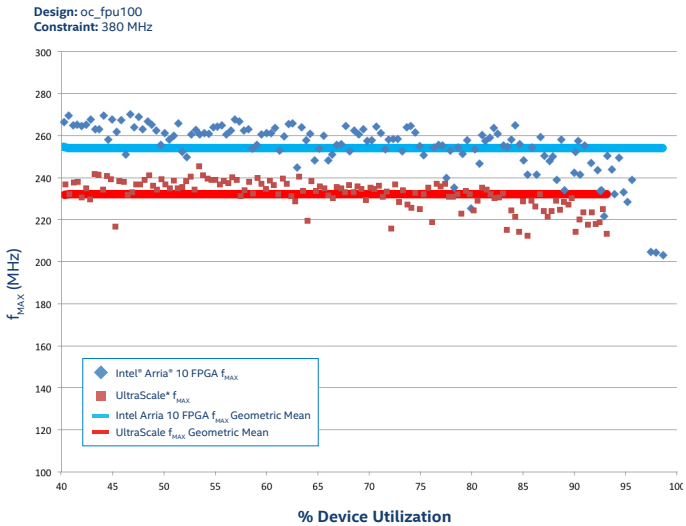

Figure 14. FPU100 Results

#	OpenCore Design Name	Design Function	Intel Arria 10 FPGA f_{MAX} (geomean)	UltraScale f_{MAX} (geomean)	Intel Arria 10 FPGA f_{MAX}/ UltraScale f_{MAX}
1	oc_avr_hp_cm4	AVR Processor	320 MHz	267 MHz	+20%
2	oc_warp_tmu	Image Processing	248 MHz	225 MHz	+10%
3	oc_reed_solomon_ decoder	Error Correction Code	325 MHz	335 MHz	-3%
4	oc_usbhostslave	USB 1.1 Controller	480 MHz	424 MHz	+13%
5	oc_dma_axi64	Single-channel 64-bit AXI Master DMA	331 MHz	283 MHz	+17%
6	oc_256_aes	Advanced Encryption Standard (AES)	242 MHz	209 MHz	+16%
7	oc_m1_core	32-bit RISC Processor	277 MHz	258 MHz	+7%
8	oc_aquarius	RISC SuperH Processor	103 MHz	106 MHz	-3%
9	oc_des_des3perf	Triple DES (Data Encryption Standard)	528 MHz	492 MHz	+7%
10	oc_fpu100	32-bit Floating Point Unit	254 MHz	232 MHz	+10%

Table 2. OpenCore Results Summary

Conclusion

Table 2 summarizes the f_{MAX} values reported for each of the OpenCore comparisons, and the relative performance of the Intel Arria 10 FPGA family relative to the UltraScale family.

Across the ten benchmark designs, eight out of ten achieve higher f_{MAX} values in the Intel Arria 10 FPGA family, in the range of 7% - 20%. As measured by device speed grades, which are typically defined as a difference of 10% - 15%, this performance advantage represents one to two speed grades.† Two of the benchmark designs achieve slightly higher performance (3%) in the competing family. In these two designs, the amount of M20K memory blocks in the Intel Arria 10 FPGA was exhausted ahead of the logic resources as device utilization increased, which negatively impacted f_{MAX}. As a result of using default compilation settings, the M20K memory blocks were used exclusively for the memory resources required by the design. Under different compilation settings allowing use of the MLAB memory resources, it is likely that the reported f_{MAX} for the Intel Arria 10 FPGA compilations would have improved. Future versions of the Intel Quartus Prime Software Suite may operate with different default compilation settings if it is determined that they will improve f_{MAX} performance while maintaining other compilation performance metrics.

Intel provides this data and the designs upon which the analysis is based with the intent of increasing transparency and understanding among programmable logic users of the performance capabilities of Intel Arria 10 FPGAs. Intel Arria 10 FPGAs and SoCs were designed to be the highest performance products in their class, and the comparisons described in this analysis using publicly-available designs help to underscore and reinforce their position as the programmable logic industry's highest-performance 20 nm FPGAs and SoCs, as measured by OpenCores designs.

References

[1] http://www.opencores.org
[2] http://www.altera.com/arriaperformance

Where to get more information

For more information about Intel and Intel Arria 10 FPGAs, visit **https://www.altera.com/products/fpga/arria-series/arria-10/overview.html**

For more information about Intel and Intel Stratix 10 FPGAs, visit **https://www.altera.com/products/fpga/stratix-series/stratix-10/overview.html**

For more information about the the high-performance architecture of Intel Arria 10 devices, consult the paper: **https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/wp/wp-01200-power-performance-zettabyte-generation-10.pdf**

To download the design files for each of the benchmarks, visit **http://www.altera.com/arriaperformance**

† Tests measure performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.