WORKING SMARTER, NOT HARDER: NVIDIA CLOSES DESIGN COMPLEXITY GAP WITH HIGH-LEVEL SYNTHESIS

FRANS SIJSTERMANS AND JC LI, NVIDIA COMPANY

HIGH-LEVEL SYNTHESIS

WHITEPAPER

INTRODUCTION

By adopting a C++ High-Level synthesis (HLS) flow using Catapult® from Mentor Graphics®, NVIDIA® was able to simplify their code by 5X, reduce the number of CPUs required for regression testing by 1000X, and run 1000X more tests to achieve higher functional coverage of their designs. HLS decreased design time by 50 percent and overall development time, including verification, by 40 percent, closing the gap between design complexity and the capacity to design.

Specifically, shortly after adoption, HLS was instrumental in slashing the schedule of a JPEG encoder/decoder by five months and in helping the NVIDIA video team upgrade two 8-bit video decoders to 4K 10-bit color for Netflix and YouTube applications within two months. Without HLS, they would have had to scrap these designs due to schedules that would have been impossible to meet with an RTL flow. The success of these projects removed any skepticism about HLS within NVIDIA and led to its use in all future NVIDIA video and imaging designs that include new or re-designed components or that target different standards or process technologies.

Quality of results was not only maintained but actually improved by using HLS micro-architectural exploration, which enabled NVIDIA to quickly produce higher quality designs through continuous refinement. Finally, in the increasingly important area of lower power usage, early evaluations show that using Catapult with the PowerPro® Low-Power platform yields a 40 percent reduction in power.

This paper will discuss the challenges NVIDIA faces in the ever evolving world of video, camera, and display standards and the reasons an HLS/C-level flow make it possible for them to succeed in this context. In particular, it will discuss the advantages of designing and verifying at the C level and how the HLS flow enabled them to meet impossible schedule demands.

AT THE FOREFRONT OF VISUAL COMPUTING

Visual computing is more important than ever, with consumer expectation for rich graphics rising amid a massive proliferation of mobile devices. NVIDIA's products span the entire spectrum of visual computing — from fundamental inventions, to processors incorporating graphics processing unit (GPU), to system components, to fully integrated systems. NVIDIA sells components and licensing IP to leading OEMs who wish to create devices differentiated by rich graphics.

The GPU, invented by NVIDIA in 1999, is the engine of modern visual computing. GPU-accelerated computing combines a GPU and CPU to accelerate scientific, analytics, engineering, consumer, and enterprise applications. The GPU has propelled computer graphics from a feature into an ever-expanding industry and is now driving new fields like computer vision, image processing, machine learning, and augmented reality.

The NVIDIA Tegra® SoCs serve the smart device markets where visual computing matters — from superphones and tablets to auto infotainment and driver assistance systems to gaming devices. The new Tegra X1, which features HLS generated blocks, is the most advanced mobile processor NVIDIA has created thus far. Tegra X1 is essentially a supercomputer featured already in the flagship video game controller for the Android platform as well as slated to be used in self-driving cars. It features a powerful NVIDIA Maxwell™ architecture, 256 GPU cores, a 64-bit CPU, unbeatable 4K video capabilities, and more power-efficient performance than its predecessor, making it ideal for the most challenging mobile and automotive applications.

The NVIDIA video team successfully adopted and used the Catapult HLS flow to design and verify several multimedia hardware blocks, including a video decoder and JPEG encoder/decoder for the Tegra X1 and the GPU GM2OX.

THE DESIGN CAPACITY GAP

The tremendous increase in video applications and devices that use them has resulted in a succession of ever more complex video compression standards. The latest standard, H.265, delivers around 50 percent better compression ratios than its predecessor, but the hardware is several times more complex.

Yet the number of designers on a project has remained relatively constant so that the gap between design complexity and the capacity to design is becoming wider and wider, and this will become even more so with each new generation. The NVIDIA video team has seen design complexity (in terms of the number of transistors per chip) grow by 1.4X per generation, with design capacity lagging behind at a 1.2X growth rate.

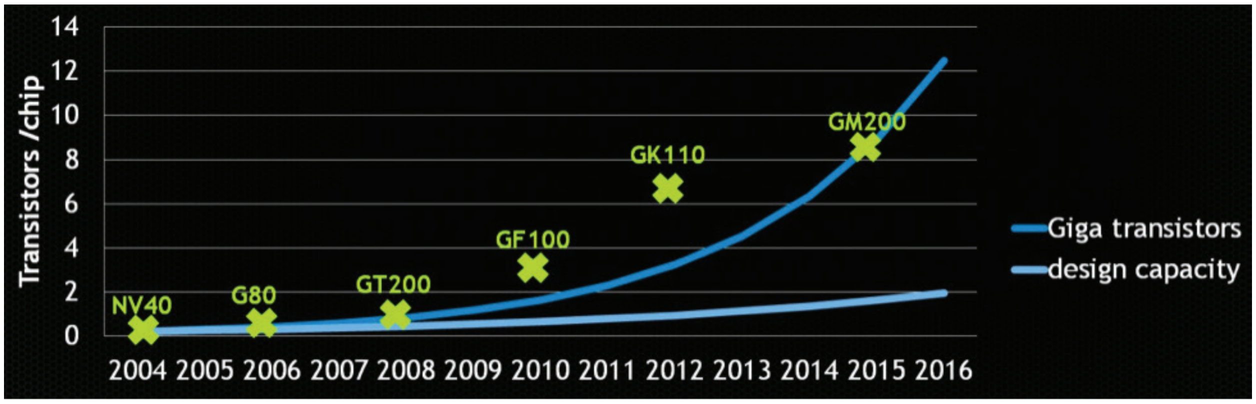


Figure 1: The growing gap between complexity and design capacity

This trend is creating a design crisis for traditional RTL flows — a crisis that will become even more acute and widespread in future design generations. The design complexity of multimedia hardware has become so high that if designers continue to work at lower levels of abstraction, they simply cannot design the products they need to design. This is true for not only video, but for all algorithmically complex SoCs.

Thus far, companies have had to continue to pour more money, engineers, and processors into the RTL flow, and they have had to increasingly rely on third-party IP. However, companies cannot profitably follow the design complexity curve by throwing money at the problem. Continually increasing head count and buying and running a warehouse of power-hungry CPUs is not an option for even companies with deep pockets. Even more troubling, third-party IP weakens a company like NVIDIA's core competency; that is, creating original, cutting edge designs that differentiate their products from the competition. If forced to use IP on more than just the generic parts of a design, a company loses control over the direction in which they want to take their products.

Further, these complex designs need to be verified. Not surprisingly, the cost and time of verification continues to rise along with design complexity, constituting a corresponding verification crisis. Designs of this complexity consume millions or billions of cycles. This many cycles requires thousands of CPUs to run the number of regressions required at the RTL, and this battery of CPUs requires virtually a power plant to fuel. Thus, it is also very important to cut costs and improve efficiency by reducing the number of CPUs.

Clearly, unless design teams do something different, they will fail. They need to do something to become more productive and energy efficient. They need a strategy that not only helps them overcome the immediate challenge of getting the current project completed on time, but also enables them to deal with growing complexity going forward and, thus, reverse the trend of exploding costs and inadequate capacity. This strategy must enable them to close the design capacity gap for those key blocks that they want to maintain control of in order to enhance their core competencies, leaving third party IP for the less essential parts. For the parts of the design they want to control, they need high-level synthesis (HLS).

Initially based on the experience of Google and their evaluations, the NVIDIA video team determined that moving up to the C level for design and verification and employing an HLS methodology was the strategy they needed to pull in their design schedules and close the design capacity gap.

ADVANTAGES OF HLS/C LEVEL FLOW

NVIDA sees two main advantages in using HLS:

1. Higher productivity and reduced effort so more design can be done faster
2. 1000X reduction in compute resources required for verification

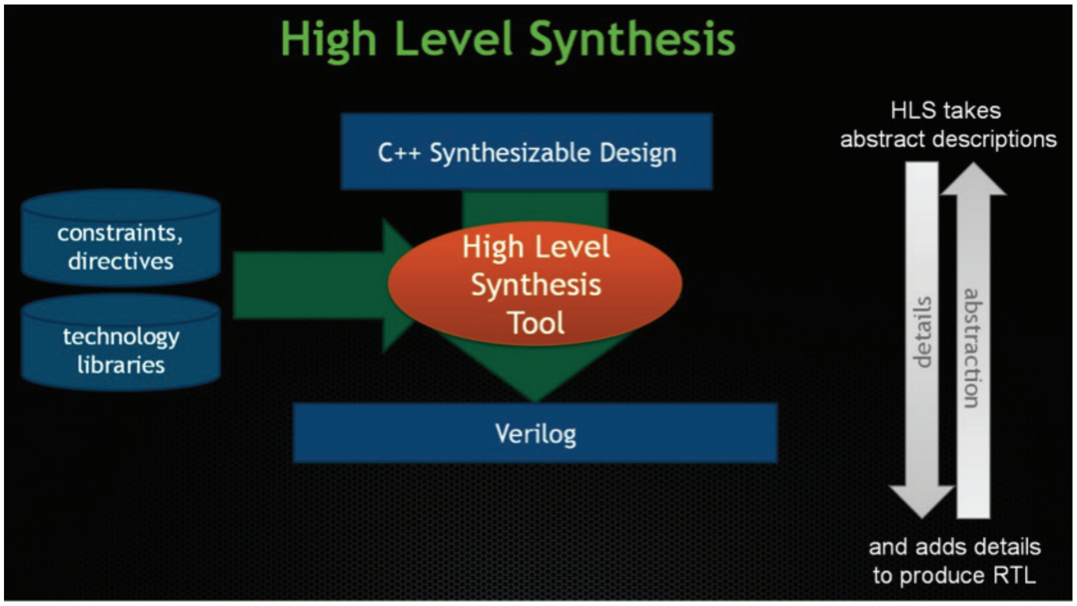


Figure 2: The HLS flow

Coding at a higher level of abstraction reduces the number of lines of code and produces simpler models. In C++, design code is typically 5X smaller than in Verilog. C++ does not include timing and does not require the finer granularity of RTL. For example, a designer can model a large function in C++ to determine the number of pipeline steps that needs to be synthesized (from C to Verilog/VHDL) without worrying about internal timing; whereas, at the RTL, this has to be done manually. In other words, pipelining changes that are painful in RTL are easy in HLS.

Another significant advantage of moving to a HLS/C flow is that it establishes a continuous process of refinement that leads to better end results. This process is made possible through a powerful HLS capability called micro-architectural exploration, which supports performing what-if analysis by changing particular design parameters and running a quick synthesis to immediately see how these changes impact the design – often in terms of area, performance, and power tradeoffs. It is also very useful for quickly retargeting a design to different technology nodes or new standards.

This is a feature that companies evaluating HLS for the first time do not generally know exists; yet it is never overlooked once a team experiences it's utility in creating better designs. Micro-architectural exploration simply opens up a design to improvements that would not even be considered in a traditional RTL flow.

Continuous stepwise refinement not only saves a great deal of time but also allows significant design improvements that could not be done otherwise. In traditional non-HLS flows, the architectural designer writes a C model then hands it off to the RTL engineer who has to implement that functionality in a completely new Verilog model. The RTL engineer must make an initial guess as to what this independently created model is going to be like and then designs it. In a typical case, RTL designers work for about three months before they have the first design, and only then do they get feedback on whether the timing and area are actually what they needed to hit. Even when it is not, they can make only small changes around that initial RTL. They cannot rewrite the entire Verilog again. It's too much work. With RTL, they are stuck with whatever they decided to design, even if it's not optimal; it's too late to make major changes. Further, this flow requires proving that the two distinct models are functionally the same.

When using HLS, only one model is created — which is a C model — and then, to hit things like timing and area budgets, the design is continuously refined from the high-level spec to final implementation guided by immediate feedback on what is working and what is not working. Once the C model is created, it is immediately run through the HLS tool. Within half a day the designer can make a change and get feedback as to whether changes made the design better or not.

This is a much easier process, refining the design in small, very fast steps to get increasingly better quality of results. NVIDIA uses this micro-architectural refinement process to explore how to best adapt a design to different configurations or performance requirements.

For example, when moving to a new process technology, designers can check to see how much bigger a dual port RAM would be compared to a single port. If HLS shows that there is not much of a difference, it is better to use a dual port because it makes the design more flexible with a negligible area hit. This would be impossible to do with handwritten RTL; in that case, to play it safe, the designer would have to go with a single port RAM if area was a concern. But with HLS, all the designer has to do is change their C code and remap it. If the scheduling process finishes successfully, they can check out other things further down the flow — synthesis, timing, evaluation, memory access efficiency — and then make the decision whether to go with single or dual ports.

HLS also allows designers to do things they could not do in the traditional RTL flow when the design has hit the timing limits for a specific process. HLS enables designers to tell whether a particular design will work at a given frequency or not very early in the design stage. Whereas, with an RTL flow, they would have to either leverage the DC re-timing capability or move a flip-flop forward to try to tune timing for a different process.

The primary emphasis of the NVIDIA verification flow is on comparing two different versions of the C model against each other. One is a high-level, purely functional C model, which is the golden model. Through the continuous refinement process, they create the optimized HLS C model. They run these two C models next to each other using the same test sequences and make sure they yield the same results. A significant benefit of HLS for functional verification is that because C models are so much simpler than Verilog models, it enables verification engineers to run 1000x more tests on the C code compared to what they could run on the RTL code. This provides better coverage and allows bugs to be fixed early on when they are easier to fix. It also produces bug free RTL and vastly reduces the time and effort required for RTL verification.

Once C-level verification is completed, they still run a restricted set of tests on the generated RTL—both as a sanity check and to verify things that cannot be modelled in C++; such as timing related functionality. If the HLS tool is doing everything right the RTL model will pass, and this has been NVIDIA's experience when using Catapult. For similar reasons, they also run some tests at the gate level; in this case, to verify that the RTL to gates synthesis was done correctly and check things that can not be seen at the RTL; such as x-propagation.

USING HLS

For these reasons and to resolve the design capacity crisis, NVIDIA adopted HLS, with Catapult as their tool of choice. They have completed a video encoder/decoder as well as a JPEG encoder/decoder for the Tegra X1 using Catapult generated code. These are large, complex designs with the HLS generated code used in, for example, seven to eight blocks in the Google video encoder and two or three blocks in the Tegra X1 JPEG encoder/decoder.

TABLE 1 - HLS PORTION OF THE VIDEO DECODER

	Gate count
All	10M
HLS Logic	6.8M
RAM	3.2M

Table 1 shows HLS portion of the video decoder (using 16ff AND2D1 cell as the unit gate in terms of gate count)

NVIDIA has used HLS chiefly at the block level. They are using Catapult generated FIFO code between blocks and are evaluating it for sub-system design and verification. HLS using C++ is most appropriate for signal processing blocks. It is especially effective for complex algorithms because these are naturally expressed in C++, HLS automates the time consuming manual pipelining and parallelization process, and HLS is very good for exploring options such as timing and resource sharing.

If the main function of the block is to take an input, then transform the data and generate an output, C++ works very well. For control heavy blocks and designs with strict cycle-by-cycle requirements, where timing becomes more important, then System C is the better language. In addition to Catapult HLS being well proven in silicon and Catapult's history as a pioneer and continued leader in HLS, Catapult's support of both C++ and SystemC was one of the reasons NVIDIA chose it over competitive solutions.

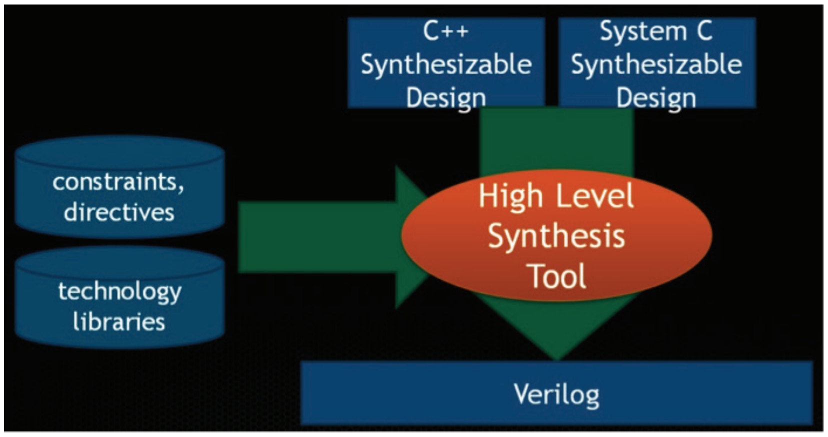


Figure 3: Catapult supports integrated or separate C++ and SystemC flows

Its integration with equivalence checking and power optimization tools are also attractive features. NVIDIA is currently evaluating those technologies. Two big advances in recent Catapult releases are very promising. The first improved the HLS generated code with conditions that enable clock gating assertions during synthesis. The other involves integration with PowerPro. Early evaluations on a JPEG design comparing the integrated Catapult-PowerPro solution to NVIDIA's internal power optimization flow shows an almost 40 percent power reduction when using the former.

SAVING THEIR SKIN

The NVIDIA video team used HLS to "rescue" them from two otherwise impossible situations: doing 14 months' work in 9 months and retargeting a 4K HEVC video decoder from 8-bit to 10-bit color.

In 2013, the NVIDIA video team was informed they had 9 months to produce a JPEG encoder/decoder for a Google 600 Mpixel video application. This new chip was full of new, "must-have" technologies, including HEVC, VP9, deep color, 4K resolution, and HDR. Their analysis of the project revealed that this chip would require 14 months to design in RTL — five more than they had. Google had been using HLS successfully for video and talking publicly about it so NVIDIA contacted them and then they decided to use HLS to pull in the schedule. Using HLS they were able to go from design-complete to fully verified RTL within 30 days, which is equivalent to a 50 percent reduction in design time, and a 40 percent reduction in the overall development effort.

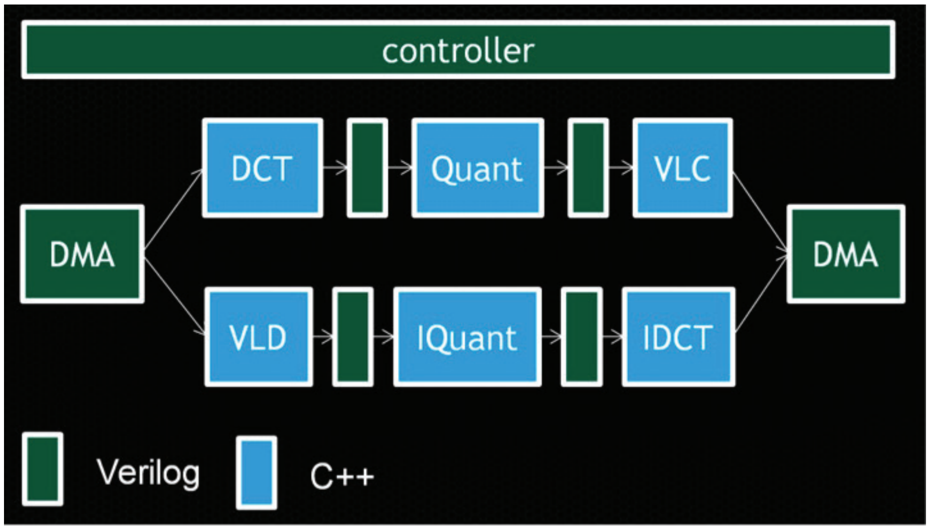


Figure 4: JPEG block diagram

When Netflix® and other multimedia stakeholders announced they were going to start supporting 4K HEVC 10-bit color, NVIDIA was already within two months of RTL freeze on a 4K HEVC 8-bit video decoder chip. The change to 10 bit would impact every register and operator of every datapath in the design, so they knew there was no way to fix this without HLS. Once they had recoded the C++ for 10 bit and ran it through HLS, they still had to verify it would work after change. They figured out that running the 100,000 tests that the HEVC standard provided in RTL would take 1000 CPUs three months. Instead, by running all of the verification at the C-level, NVIDIA was able to complete verification within three days using only one CPU.

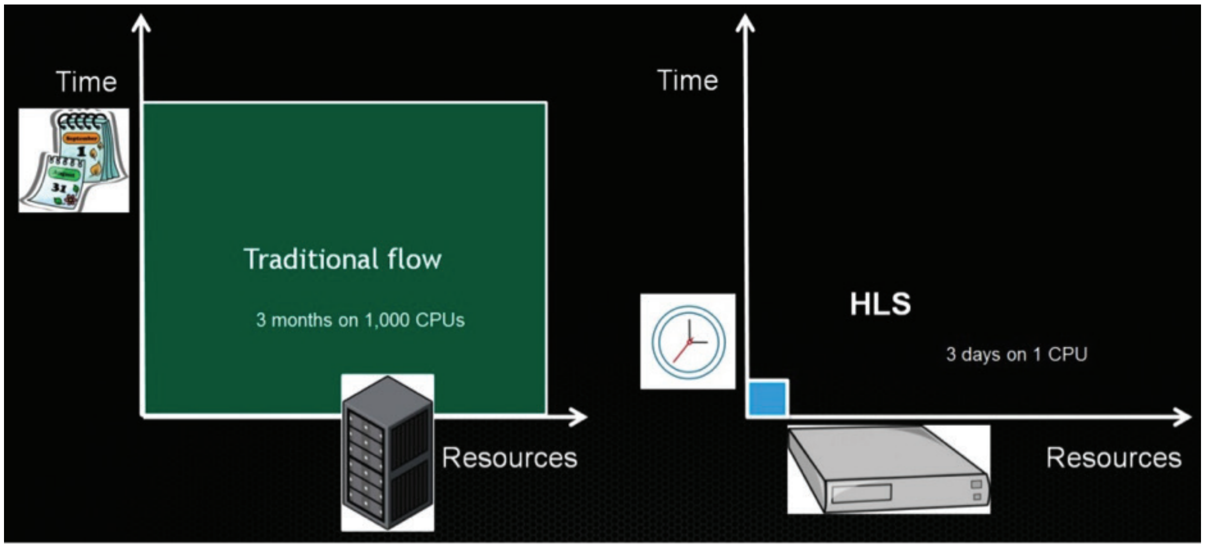


Figure 5: Video decoder compatibility regression

Not only did HLS save this video decoder from the dust bin, but it enabled NVIDIA to tape out two 10-bit versions, a 20 nm block for a mobile GPU at 510 MHz and a 28 HP discrete GPU at 800 MHz. Because of the ability to make rapid modifications and receive immediate feedback, NVIDIA was able to re-design the decoders from 8-bit to 10-bit within a few weeks and still meet the tape out and final netlist deadlines on both of these projects.

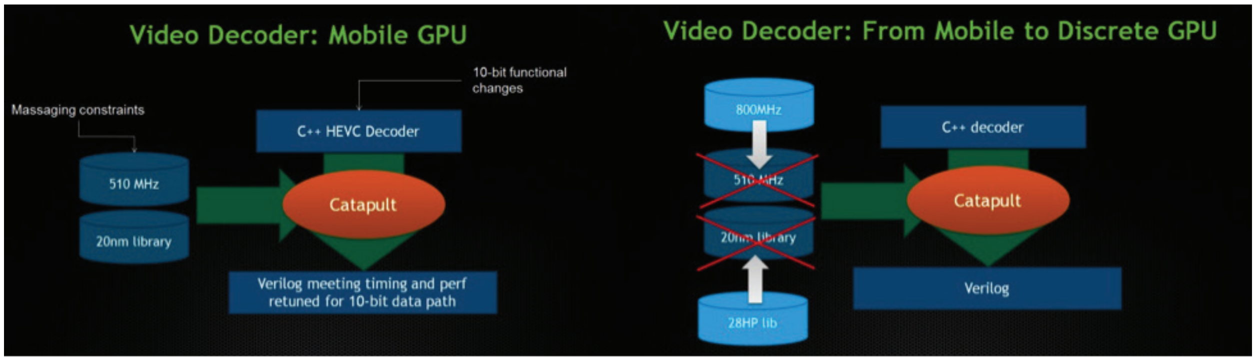
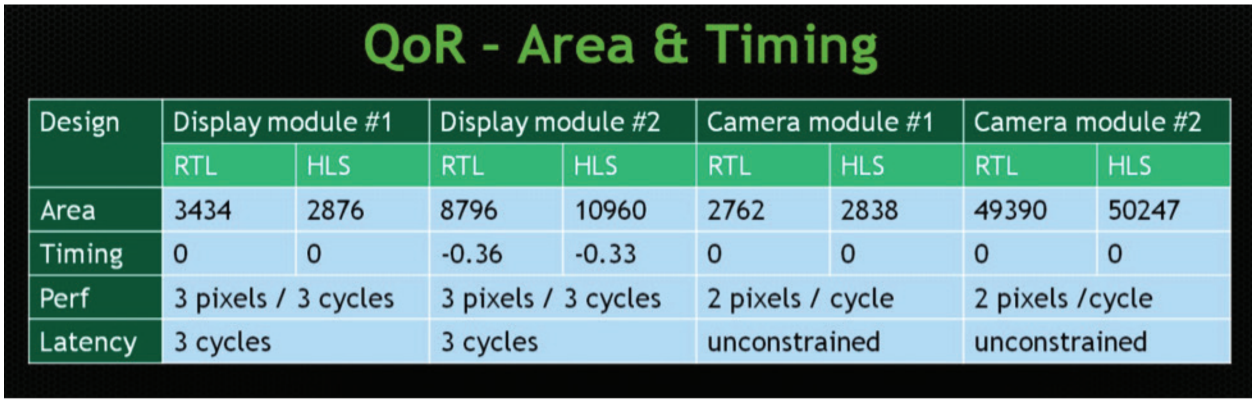

Video Decoder: Mobile GPU
10-bit functional changes
Massaging constraints
C++ HEVC Decoder
510 MHz
20nm library
Catapult
Verilog meeting timing and perf retuned for 10-bit data path
Video Decoder: From Mobile to Discrete GPU
800MHz
510 MHz
20nm library
28HP lib
C++ decoder
Catapult
Verilog

Figure 6: The HLS micro-architectural refinement approach delivers a two for one design deal

This not only enabled them to deliver products under otherwise impossible circumstances, but also eliminated any remaining skepticism regarding HLS at NVIDIA, resulting in its adoption throughout NVIDIA on all multimedia hardware designs that use a new standard, target a new process technology, or require major revisions. In other words, any new design or component in every NVIDIA product will have High-Level Synthesis blocks inside.



QoR - Area & Timing

Design	Display module #1		Display module #2		Camera module #1		Camera module #2	
	RTL	HLS	RTL	HLS	RTL	HLS	RTL	HLS
Area	3434	2876	8796	10960	2762	2838	49390	50247
Timing	0	0	-0.36	-0.33	0	0	0	0
Perf	3 pixels / 3 cycles		3 pixels / 3 cycles		2 pixels / cycle		2 pixels /cycle	
Latency	3 cycles		3 cycles		unconstrained		unconstrained	

Figure 7: Comparison of HLS versus production RTL (timing goal 600–700 MHz range, 16nm process)

The success of the NVIDIA HLS team was highlighted with the recent announcement of a four star product used for SHIELD TV. HLS enabled video features headlined the release. The NVIDIA Tegra X1 chip powers the first 4K set top box on the market, supporting both Netflix and YouTube 4K streaming. This means that the NVIDIA team produced a highly successful, cutting-edge consumer product two years after first Catapult trial license.

CONCLUSION

One of the intrinsic aspects of HLS is the abstraction of as much design and verification work as appropriate to the C level. Working at a higher level of abstraction significantly reduces the lines of code and slashes the number of CPUs and time required for regression runs. The Catapult HLS C++ flow gave NVIDIA the increased design and verification productivity they needed to close the gap between design complexity and design capacity.

By enabling verification teams to achieve complete functional coverage of the C code, offering designers a continuous refinement process through micro-architectural exploration, and providing immediate results through very fast synthesis of RTL code, HLS generates production worthy RTL and yields better quality of results in a much shorter time than it would take to create and verify designs in a traditional RTL flow. The continuous refinement process also makes it easier to port designs to different technologies.

HLS enabled the NVIDIA team to make rapid design adjustments and complete projects within very tight schedules even as design complexity increased and the number of designers remained roughly the same. Shortly after adoption, HLS made it possible for the NVIDIA video team to meet deadlines and deal with late specification changes that would have been impossible with a traditional RTL flow.

HLS and C-level design and verification reduces the overall development effort by 40 percent and design time by 50 percent. This level of efficiency made it possible for an NVIDIA video team of set size to successfully complete projects that they could not have done if they stuck with an RTL flow. The success of the video team's application of HLS resulted in its adoption company-wide for all new NVIDIA designs.

When it comes to working smarter, not harder, HLS is the way to go.

For the latest product information, call us or visit: www.mentor.com

©2015 Mentor Graphics Corporation, all rights reserved. This document contains information that is proprietary to Mentor Graphics Corporation and may be duplicated in whole or in part by the original recipient for internal business purposes only, provided that this entire notice appears in all copies. In accepting this document, the recipient agrees to make every reasonable effort to prevent unauthorized use of this information. All trademarks mentioned in this document are the trademarks of their respective owners.

Corporate Headquarters
Mentor Graphics Corporation
8005 SW Boeckman Road
Wilsonville, OR 97070-7777
Phone: 503.685.7000
Fax: 503.685.1204

Sales and Product Information
Phone: 800.547.3000
sales_info@mentor.com

Silicon Valley
Mentor Graphics Corporation
46871 Bayside Parkway
Fremont, CA 94538 USA
Phone: 510.354.7400
Fax: 510.354.7467

North American Support Center
Phone: 800.547.4303

Europe
Mentor Graphics
Deutschland GmbH
Arnulfstrasse 201
80634 Munich
Germany
Phone: +49.89.57096.0
Fax: +49.89.57096.400

Pacific Rim
Mentor Graphics (Taiwan)
11F, No. 120, Section 2,
Gongdao 5th Road
HsinChu City 300,
Taiwan, ROC
Phone: 886.3.513.1000
Fax: 886.3.573.4734

Japan
Mentor Graphics Japan Co., Ltd.
Gotenyama Garden
7-35, Kita-Shinagawa 4-chome
Shinagawa-Ku, Tokyo 140-0001
Japan
Phone: +81.3.5488.3033
Fax: +81.3.5488.3004

Mentor Graphics®

MGC 01-16 1033790-w