MOVE TO A HIGH-LEVEL SYNTHESIS FLOW TO REMAIN COMPETITIVE

BADRU AGARWALA, GENERAL MANAGER, MENTOR, A SIEMENS BUSINESS

Mentor®
A Siemens Business

WHITEPAPER

HIGH-LEVEL SYNTHESIS

www.mentor.com

INTRODUCTION

In a matter of decades, the IC design community has seen the progression of automation move from hand-drawn geometries, to schematics, and to RTL design abstraction. But today's competitive market for state-of-the-art image processing, high-bandwidth communication, and computer vision and neural computing solutions demand another level of abstraction. The prevailing design flow centered on RTL design and verification does not allow companies in these markets to be competitive as this flow takes too long to get to market. Successful companies in these markets are nimble, can target many potential implementation solutions, and can make last-minute specification changes while staying on schedule (Figure 1). The only way that they can achieve this success is by moving up to a high-level synthesis (HLS) flow using C++ and/or SystemC.

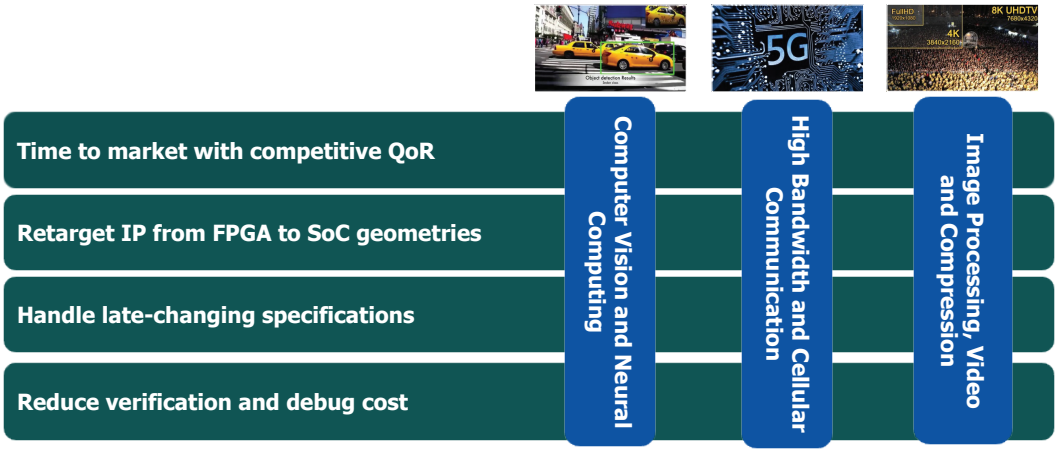

Figure 1: Moving to an HLS flow provides competitive advantages

In a traditional RTL design and verification flow (Figure 2) the RTL signoff is performed in parallel with the RTL debug. The verification consists of constrained random and directed tests and coverage must be closed for both functional and code coverage. In this flow the verification team is in the critical path. Functional RTL is not available until later in the design flow and because of this, there is no way to start the verification process earlier. It is not unreasonable to see verification and debug taking more than 50% of the design flow time. Another drawback to this traditional flow is that there is little chance for design exploration because the RTL is not available till late in the process. Making late-stage algorithmic or architectural changes is almost impossible.

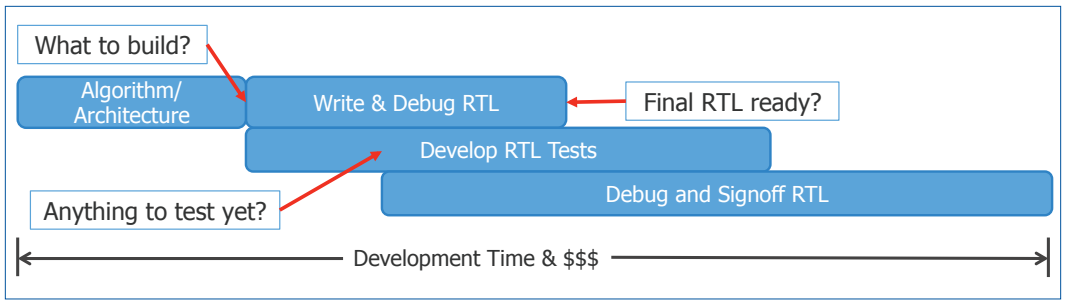

Figure 2: The RTL design and verification flow takes too long

INTRODUCING THE CATAPULT HLS PLATFORM

The Catapult HLS Platform provides a hardware design solution for designers that generates high-quality RTL from C++ and/or SystemC descriptions that target ASIC or FPGA implementation. The platform delivers the ability to check the design for errors before simulation, provides a seamless and reusable testing environment, and supports formal equivalence checking between the generated RTL and the original source. The platform flow (Figure 3) ensures fast design and verification and delivers power-optimized RTL ready for simulation and RTL synthesis.

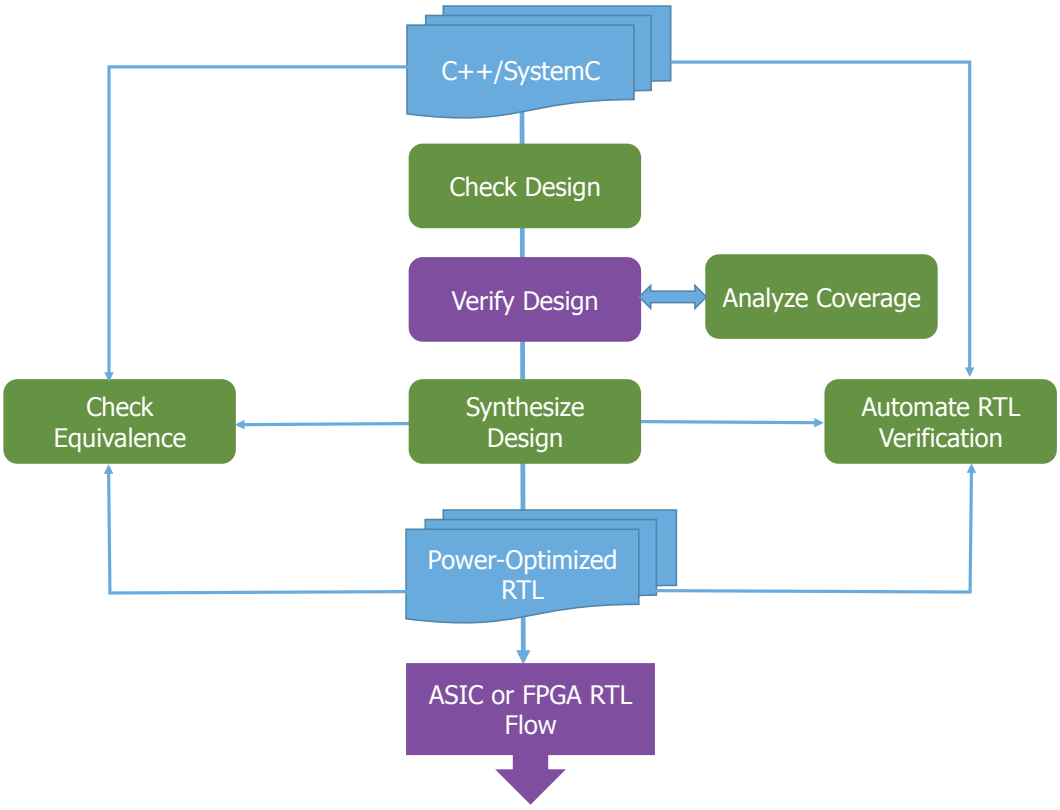

Figure 3: The HLS design and verification flow

Recognizing that C++ and SystemC synthesis alone does not provide a complete solution, Mentor created the Catapult HLS platform to bridge the high-level language flow to the standard ASIC and FPGA RTL flow. This solution includes the following elements that should be very familiar to the hardware designer:

- C++/SystemC: choose either C++, SystemC, or both to implement your design. The Catapult HLS Platform supports all the popular SystemC abstractions, including untimed, loosely-timed, and cycle-accurate descriptions.
- Check design: Catapult Design Checker (CDesignChecker) provides linting and formal analysis to find language and coding bugs before simulation and synthesis.
- Analyze coverage: Catapult Coverage (CCOV) provides hardware-aware verification coverage to C++ and SystemC with integration to the Questa Unified Coverage Database (UCDB) for RTL coverage planning. Traditional software coverage tools on the market do not understand hardware so you cannot rely on them for coverage analysis of your design.

- Synthesize design: Catapult generates optimized RTL using your design directives and performs power optimization using the PowerPro technology.
- Automate RTL verification: For a dynamic method of verifying C++/SystemC to RTL, SCVerify automatically generates the framework that reuses the tests from your C++/SystemC testbench and compares results of running them through both the RTL and C++ DUT. It also auto-generates a complete UVMf infrastructure for further RTL verification. The tool propagates the Assert and Cover statements from your C++/SystemC into Open Verification Library (OVL) or Property Specification Language (PSL) in the generated RTL, ensuring continuity of functional checking throughout the flow.
- Check equivalence: SLEC-HLS provides formal C++ to RTL equivalence checking to ensure that your high-level description matches the generated RTL. (SystemC to RTL equivalence checking will be available in a future release).
- Power-optimized RTL: PowerPro provides early RTL power estimation and power optimization to rapidly converge on a power-optimized solution.
- ASIC or FPGA RTL flow: target ASIC or FPGA implementations with the same design descriptions, allowing you the flexibility to move between technologies over the lifecycle of a product.

By employing these elements of the Catapult HLS Platform solution you take your product to market faster and at a lower cost. Imagine getting to market months ahead of your competitors who are still using a traditional RTL flow.

REDUCING CODING ERRORS AND MAINTENANCE

Obviously, automatically generating RTL code that meets your specifications reduces the errors that can occur when hand-coding your design. An RTL expert will likely state, "I can code for power, performance, and area better than any tool." However, users of the Catapult HLS Platform will respond with, "The HLS design flow results in RTL code that matches power, performance, and area of hand-coded RTL in ½ to ¼ the time." Sure, if you had extra months to tune the RTL by hand you might squeeze out slightly better design metrics, but you cannot afford that extra time. During evaluations, the question of hand-coded versus generated RTL quality is the first question answered in order to move forward with a HLS solution. If the solution could not generate RTL code equivalent to the quality of hand-coded RTL, no one would purchase the product.

The difference between the line count of RTL code versus C++ code, for example, is significant for both debugging and for maintenance purposes. For instance, STMicroelectronics® designed a JPEG encoder that has 20K lines of RTL code but only 2K lines of C++ code. In another example, they designed a 2D rendering IP block that has 60K lines of RTL code versus 13K lines of C++ code.

To further improve maintenance and time-to-market, teams can create a library of reusable, parameterized C++ or SystemC templates for design elements that they repeatedly deploy in designs. This library provides benefits that include:

- The designers can focus on improving algorithms instead of coding common elements.
- The team can create designs faster by including the templates.
- Any changes to the templates are propagated to the designs that use them.
- Eliminate hand-coding errors for code that the templates implement.

Because of these strong benefits, Mentor makes a common set of these available to the design community.

REDUCING VERIFICATION TIME AND DEBUG COST

Synthesizing high-level descriptions to RTL provides incredible improvements in productivity. But, that is only one aspect of the HLS solution. Productivity improvements for verification begin with the fact that the simulation time at the C++/SystemC level is 50X to 1000X faster than RTL simulation. This means that the team can afford many more test runs within a defined schedule to ensure quality. In order to improve verification time even more, the Catapult HLS platform seamlessly integrates into the Questa verification flow. This flow starts with coverage- driven analysis where the tool translates assertions and coverage statements from the HLS level into the RTL for coverage closure (Figure 4).

```
int alu(int a, int b, uint2 opcode) {
  cover(opcode==ADD);
  cover(opcode==SUB);
  cover(opcode==MUL);
  cover(opcode==DIV);

  short r;

  switch(opcode) {
    case ADD: r = a+b;
              break;
    case SUB: r = a-b;
              break;
    case MUL: r = a*b;
              break;
    case DIV: assert(b!=0);
              r = a/b;
              break;
  }
  return r;
}
```

TYPE cvg	52.7%	100	52.7%
CVP cvg::cvp_lz	100.0%	100	100.0%
CVP cvg::cvp_vz	25.0%	100	25.0%
CROSS cvg::cross_lz_vz	33.3%	100	33.3%
/scverify_top/rtl/top_vip_inst/cvg_b_rsc			
TYPE cvg	88.8%	100	88.8%
CVP cvg::cvp_lz	100.0%	100	100.0%
CVP cvg::cvp_vz	100.0%	100	100.0%
CROSS cvg::cross_lz_vz	66.6%	100	66.6%

Figure 4: Supporting cover points and assertions in C++

The SCVerify flow automatically generates the verification infrastructure for verifying the functionality of the generated RTL against the original source code and reuses the original C++/SystemC testbench. This saves the team significant time. Catapult can also automatically generate a UVM test environment (Figure 5) that includes the source HLS C++ model as a predictor connected in parallel with the RTL so that SystemVerilog constrained-random sequences can be pushed through both with the outputs and compared using UVM scoreboards.

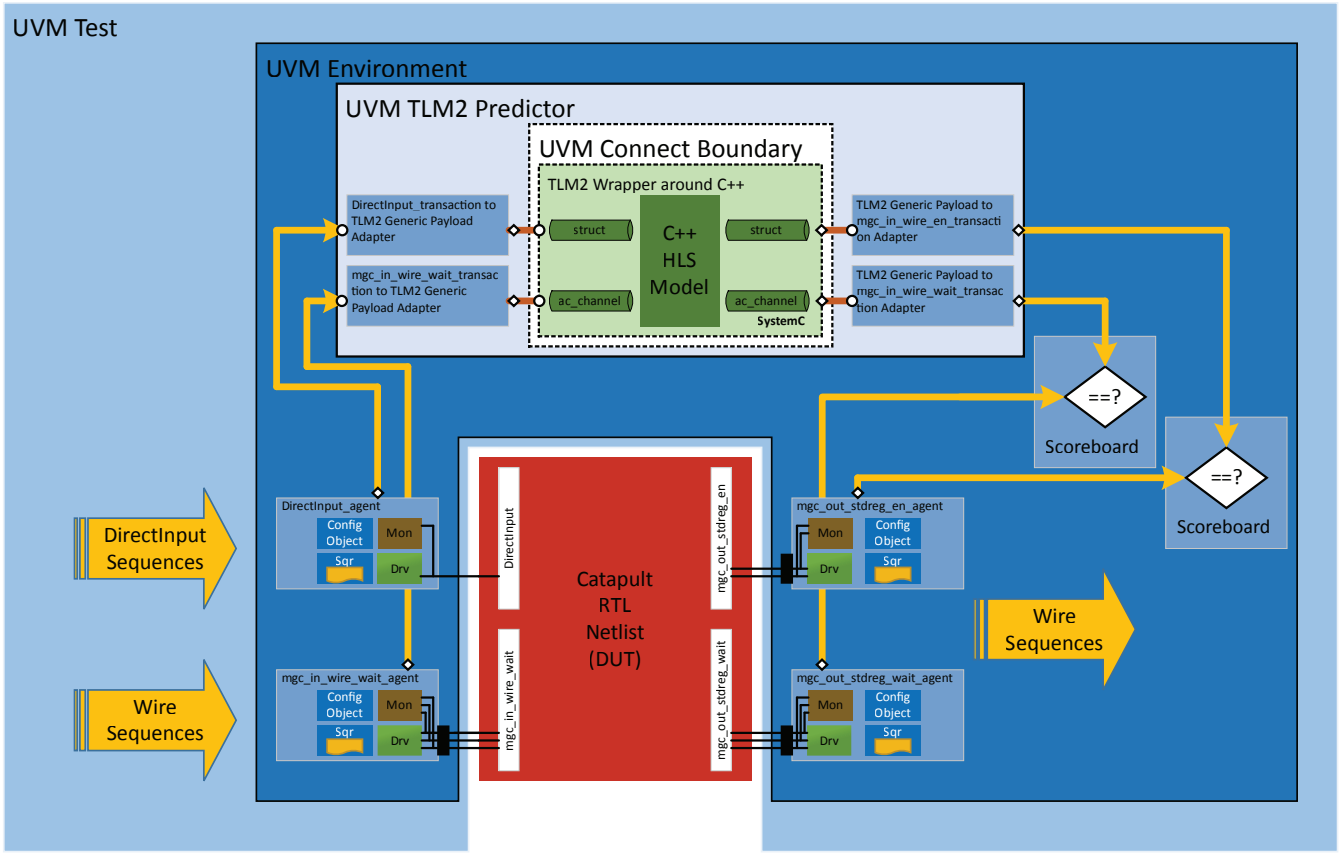

Figure 5: Automatically-generated UVM structure

The team saves additional time using the Catapult HLS Platform solution because they can eliminate iterative RTL functional verification during the development cycle. Verifying the generated RTL is a three step process (Figure 6):

1. C++/SystemC verification: the team creates functional tests in order to approach 100% C++/SystemC coverage. This helps to improve RTL coverage.
2. RTL functional verification: the original functional RTL tests are used in order to achieve high coverage metrics.
3. RTL structural coverage: High-Level Synthesis adds structural elements such as interfaces, buffers, and control that were not present in the original source code, so the team needs to add reset and stall tests to complete RTL coverage. The SCVerify flow can automatically insert stall behavior in the test environment. Through the use of CoverCheck, the flow automates formal waiver generation for unreachable expression combinations in the RTL.

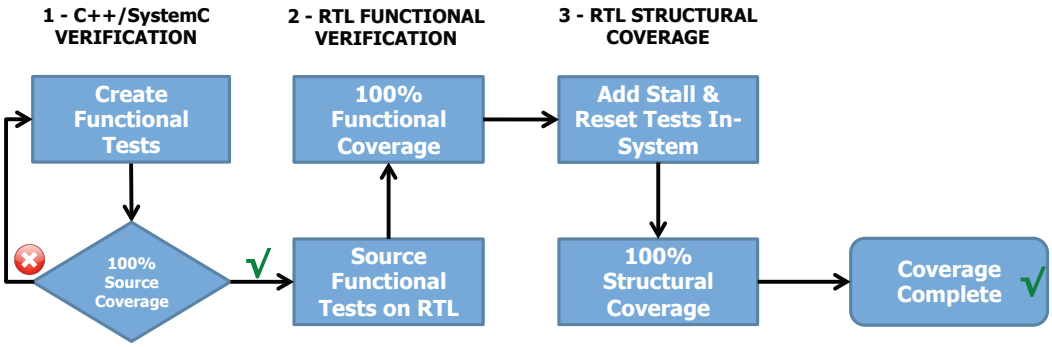

Figure 6: The verification process

Because verification time is reduced, the team has the freedom to do more tests within schedule.

PROVIDING THE FREEDOM TO EXPERIMENT

The Catapult HLS Platform provides teams with unparalleled flexibility to experiment with their designs. This flexibility starts with allowing the source code to remain unchanged but guiding solutions by applying constraints. Teams can experiment with micro-architectures by simply changing constraints and analyzing the effects on the design (Figure 7). You can explore your ideas and analyze the impact on parallelism, throughput, area, latency and try out memory implementations like DPRAM, SPRAM versus registers. This flexibility is impossible when coding with RTL.

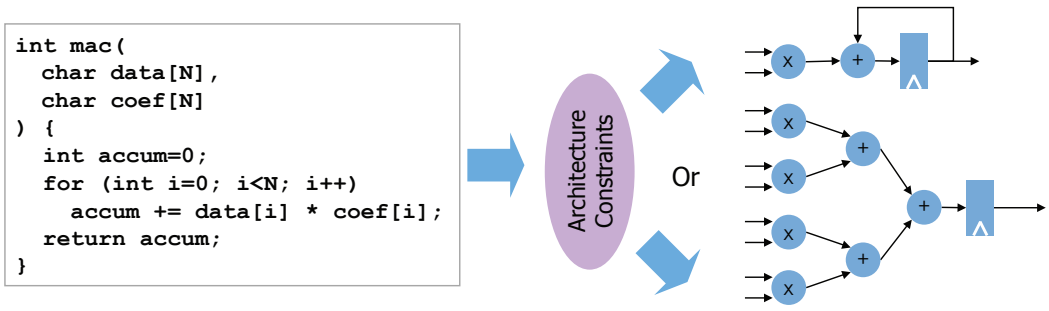

Figure 7: Change constraints instead of source code

The same source code can be targeted to different ASIC or FPGA libraries to assess the impact of technology on the design. This is a powerful solution for teams that start their product line implemented in FPGA technology and then want to move to an ASIC later in the product lifecycle, without rewriting their source code. As teams strive to look at different implementation architectures for solutions such as artificial intelligence, the Catapult HLS Platform eases experimentation of targeting any technology without modifying the source code.

The Platform offers many techniques to help visualize the impact of experimentation. For example, you can change the clock frequency for your design, re-target to a new technology, and then visualize the impact on scheduling. The Catapult HLS Platform always attempts to produce the lowest area solution that meets timing for the target technology. As Figure 8 shows, different pipelining is necessary to meet timing in a slow process versus a faster process without changing the source code.

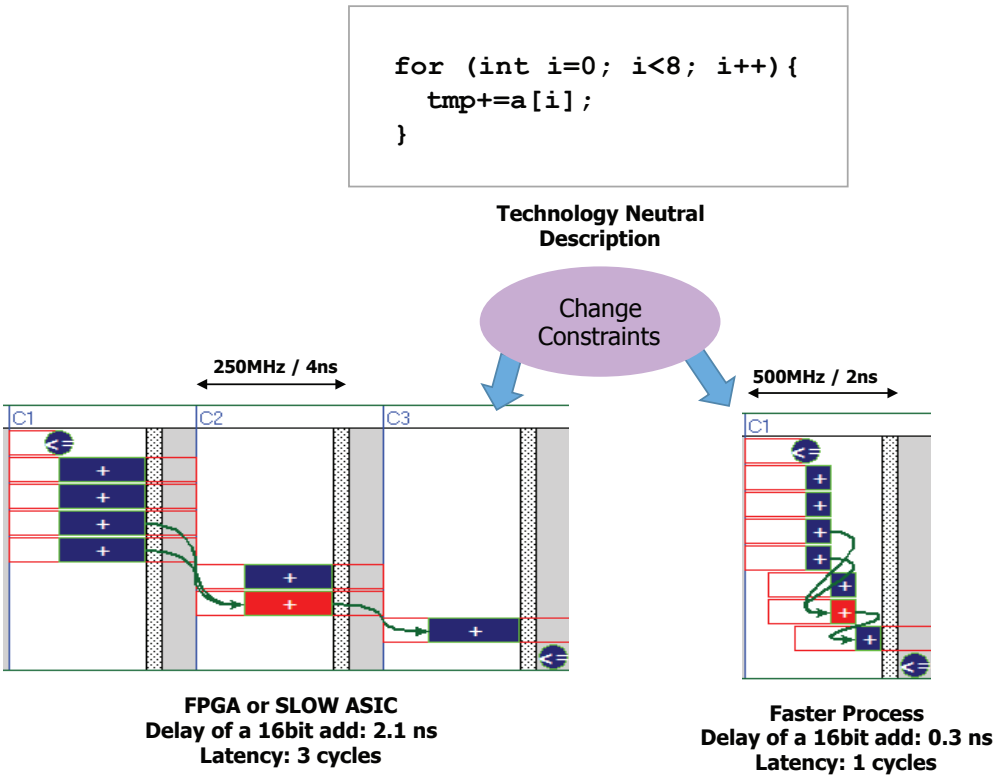

Figure 8: Changing frequency and technology without changing source code

REDUCING TIME-TO-MARKET

Moving to a proven, production HLS flow is not about obtaining incremental time savings. It is about a significant reduction in the time and cost of getting to market with your product. While matching the power, performance, and area of hand-coded RTL, the HLS flow slashes the design and verification time of video, imaging, and communication products, as Figure 9 shows.

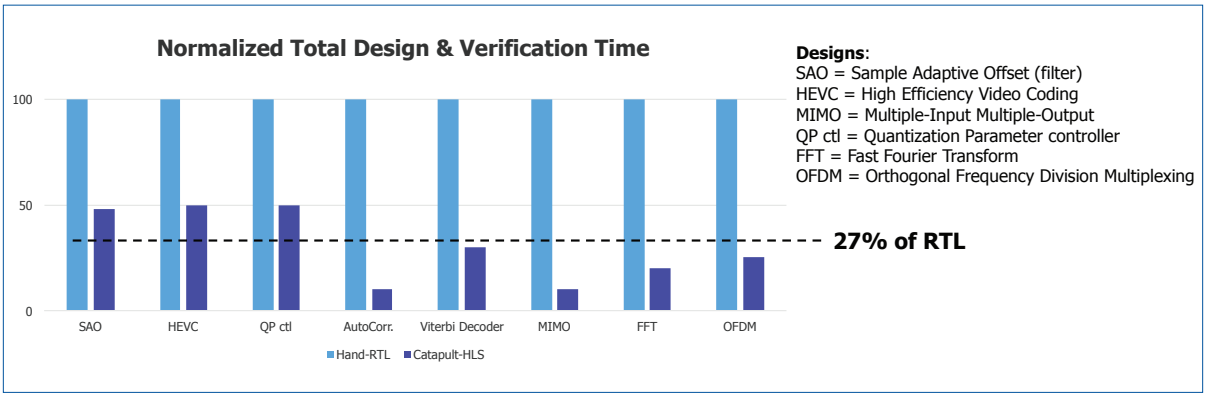

Figure 9: Results from actual customer designs

Reducing time-to-market means that teams can explore many alternate solutions, target different technologies, and run significantly more tests to assure quality designs while meeting schedule. And, because the Catapult HLS Platform allows you to quickly make last-minute changes due to new requirements, that schedule remains unchanged. This solution supports an ECO methodology, but in the last decade of deployment, Mentor has rarely seen customers needing to use it.

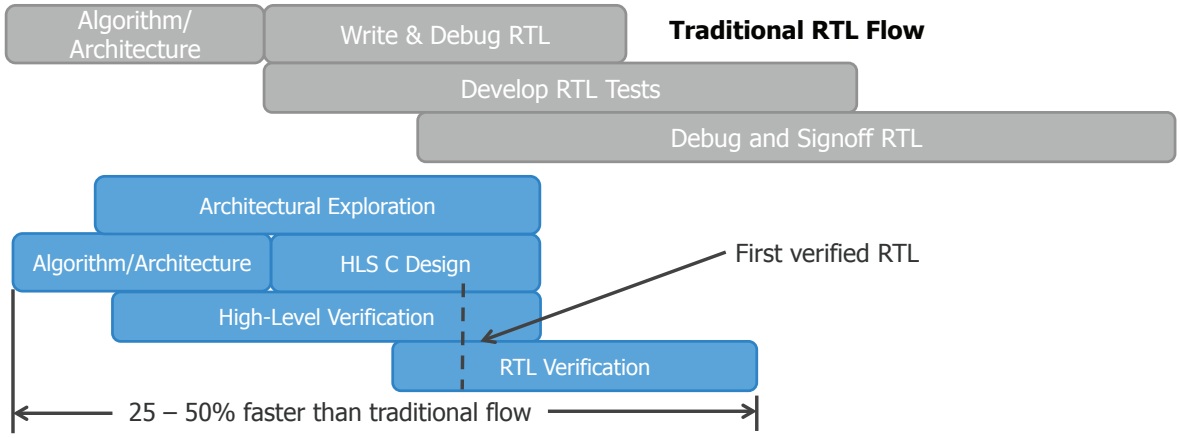

Figure 10: Reducing time-to-market when compared to Figure 2

EASING THE TRANSITION TO HLS

The Catapult HLS Platform is targeted at experienced RTL designers that want to move up in abstraction in order to experiment with creative solutions while still meeting demanding schedules. The question of whether or not the flow can match quality of results from expert hand-coders has been answered years ago, so taking your expert RTL design experience to a new level can help take your ideas to hardware solutions fast. All the RTL concepts that you know well, like register allocation, memories, timing, power optimization, and technical tradeoffs all still apply to the HLS flow.

The Catapult HLS Platform helps RTL designers quickly see how their constraints map from the high level down to RTL by visualizing C++/SystemC transformations and relationships, cross-probing for debug, and analyzing scheduling impact due to changing specifications. The automated verification solution maps seamlessly into the UVM methodology, making experienced verification teams more productive. Mentor also offers seminars, self-paced training, open source libraries, templates, examples, and consulting to get experienced RTL designers up and running quickly on the HLS flow.

PROVING THE POINT

Writing joint case studies that point out the benefits of using the Catapult HLS platform for production designs is a powerful way to share successful ideas with the world. Here are some samples of these case studies that prove the value of using the HLS flow:

- Qualcomm®: had discovered that IP complexity equaled significant design and verification time for their Snapdragon-based codecs, Ultra HD resolution, and aggregated wireless designs. By adopting the Catapult HLS Platform, they sped up time to market by 1.5 to 2X over their previous design and verification flow for these designs.
- Google®: designed an independent video decode IP in half the time as their previous RTL design flow by using the Catapult HLS Platform. Because of this time savings, they were able to add H.265 support in less than three months and added 10-bit color in weeks using the HLS flow. And, the IP went from 300K lines of RTL code to 69K lines of C++ code making it easier to maintain and debug. They recently completed a 5 million gate VP9 encoder using the HLS flow.
- NVIDIA®: has publically spoken about how using the Catapult HLS Platform has "saved our skin, twice." They achieved a cost reduction of about 80% for functional verification by reducing their simulations (that run in the Cloud for a fee) from 3 months on 1000 CPUs to 2 weeks using 14 CPUs. The HLS flow allowed them to change their VP9/HVEC code from 8 to 10 bits in two weeks and change from 20nm/500MHz to 28nm/800MHz in three days using the HLS flow.

THE TAKE-AWAY

The Catapult HLS Platform brings design to a higher level of abstraction in order to get your products to market faster at a much lower cost. The Platform delivers power-optimized code that fits seamlessly into the RTL verification flow and it is virtually immune to schedule delays due to changes in specification. Teams that move to this HLS flow never return to hand-coded RTL as a starting point for their products.

To learn more about the Catapult HLS Platform, visit the website here.

To explore further the successful companies mentioned in this paper, click on these case study links:

- STMicroelectronics Case Study
- Qualcomm Case Study
- Google Case Study
- NVIDIA Case Study

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

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 Trust Tower
7-35, Kita-Shinagawa 4-chome
Shinagawa-Ku, Tokyo 140-0001
Japan
Phone: +81.3.5488.3033
Fax: +81.3.5488.3004

MGC 11-17 TECH16700-w