Catapult High-Level Synthesis and Verification

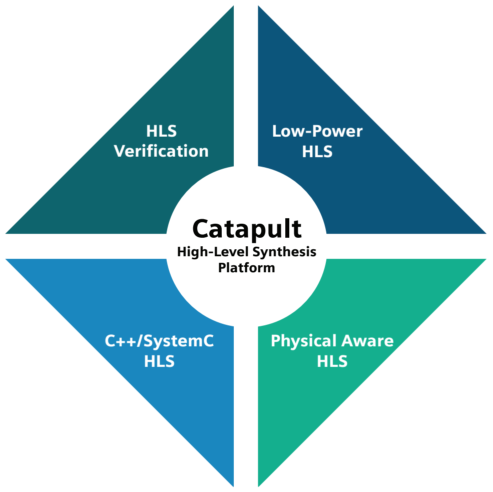
Siemens Digital Industries Software Design Platform Empowering Designers

Benefits

- Easier to design functionality in standard C++/SystemC
  - Create C++/SystemC synthesizable and executable specification
  - Write 80% less code for easier development and debug
- Complete platform for power, performance, area optimization
  - 10X productivity over hand-coded RTL with equivalent QoR
  - Explore microarchitecture alternatives
  - Deep-Sequential analysis with PowerPro "under-the-hood"
- Reduces verification cost by 80%
  - Simulate functionality 100-500x faster than RTL
  - Catapult Design Checker to find bugs fast before synthesis
  - Catapult Coverage for HLS-aware coverage metrics
  - Automatic generation of RTL-to-C verification environment
  - Fast path to automated RTL coverage closure

The Catapult High-Level Synthesis (HLS) Platform empowers designers to use industry-standard C++ and/or SystemC to describe functional intent, enabling them to move up to a more productive abstraction level for both design and verification of ASICs and FPGAs. For designs and IP where time-to-market is critical, power/performance information is needed early and specifications are frequently changing (Automotive Vision, Image Processing, Deep Learning, Video CODEC, 5G/IoT Communications, etc.), Catapult HLS provides the only effective way to meet these pressures without compromising quality and functionality.

To achieve the maximum productivity gain from a C++/SystemC HLS methodology, it is necessary to have the performance and capacity to handle today's large designs coupled with a comprehensive flow through verification and implementation. Catapult has been proven in production design flows with 1,000s of designs and the resulting RTL adheres to the strictest corporate design guidelines and ECO flows. In addition to Catapult Synthesis, only Catapult has integrated High-Level Verification (HLV) tools and methodologies that enable designers to complete their verification signoff at the C++ level with fast closure for RTL.

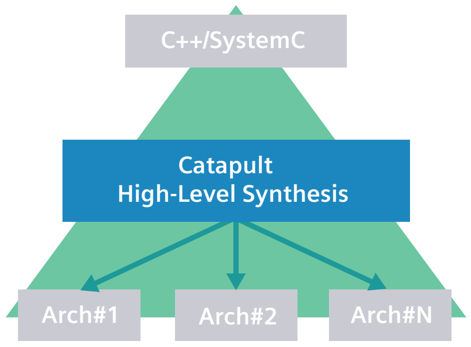


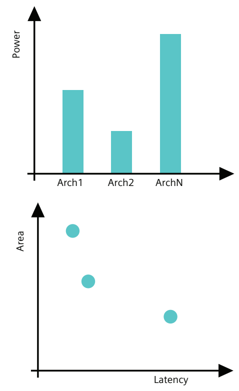
Catapult High-Level Synthesis Platform

Catapult High-Level Synthesis

Benefits *continued*

- Easy to use HLS debug and visualization
  - Design Analyzer for visualization of the HLS transformation process
  - Design Advisor detects the worst coding style mistakes
- Production proven flows with thousands of designs
  - Full support for datapath and control logic synthesis
  - "Physically Aware" synthesis
  - Tight integration to RTL synthesis with "on-the-fly" characterization for better QoR
  - Capacity for multi-million gate ASIC and FPGA designs
  - Top-down and bottom-up design management
  - Complete ECO flow
- Formally proves the functional correctness of RTL against C++ models
- Supports C++ to Catapult-generated RTL
- Find difficult to detect bugs without writing complex testbenches
- Formally verifies designs despite language & abstraction differences
- Does not require mapping flip- flops or intermediate state-points before starting verification
- Ensures hardware intent remains consistent during system-level model refinement





Catapult transforms the designer's algorithmic/behavioral description and applies micro-architectural constraints through user- specified directives.

Catapult enables faster and easier design in C++/SystemC

Using Catapult HLS simplifies the traditional design flow by automating the RTL generation based on a higher level functional description and architectural constraints. Designing using C++/SystemC, compared to RTL, requires up to 80% fewer lines of code, making HLS code significantly easier to write and debug. Incorporating last minute specification changes and even retargeting to a different technology is possible because of the separation of the design functionality and the implementation details. The RTL can simply be regenerated based on the modified HLS model and new constraints.

Catapult supports multiple abstractions for faster simulation and modeling

With Catapult, the designer can choose from an RTL-like coding style in SystemC to a fully untimed high-performance model in C++/SystemC, or a combination of both. During the synthesis process, Catapult transforms the designer's algorithmic/behavioral description and applies micro-architectural constraints through user-specified directives. Catapult's patented interface synthesis technology allows timing, protocol, and bandwidth to be defined, adding the necessary RTL hardware during the C++ synthesis process. SystemC designs use the MatchLib Connections interface library which contains pre-built protocols from simple data/rdy/vld protocols up to complex AXI4 memory masters and slaves. Catapult allows the designer to specify parallelism, design throughput, and memory vs. register implementation using tool options instead of hardcoding them in the RTL. This results in an easy to reuse design and an optimized implementation.

Catapult is a complete platform for power, performance, area optimization

Catapult automatically applies general performance and power optimizations during design transformation. In addition, Catapult is the industry's first HLS tool that targets power as an optimization goal and uses PowerPro® "under-the-hood" for power analysis and to apply advanced power optimizations. This provides a fast and accurate flow for tradeoffs of power, performance, and area. The generated RTL is able to match or surpass the quality of results (QoR) of hand-coded RTL.

Catapult provides easier debug and optimization control

Catapult provides built-in graphical analysis tools, such as a Gantt Chart Viewer and the Design Analyzer tool, to enable full visibility of the HLS results and to allow the designer to exert control over design decisions. The integrated cross-probing support between different analysis views, including the C++/SystemC source code, enables the designer to rapidly focus on the problematic areas, to add or change directives, and to converge interactively on the optimal solution. Design Analyzer enables easier and deeper debugging where designers are able to utilize advanced source code navigation, control flow analysis tools, and focussed debug capabilities such as failed schedule analysis and automatic identification of coding style mistakes.

Catapult High-Level Synthesis

Catapult Physical for better QoR below 10nm

"Physically aware" with tight integration to RTL synthesis. Optimizations take advantage of more accurate characterization data leading to shorter pipeline stages and fewer registers. Logic is more evenly distributed in each pipeline stage allowing for more aggressive resource sharing.

Catapult is proven in 1,000's of projects to reduce complete project time by 50%

Catapult was released in 2004 as a C++ based ASIC synthesis tool for datapath dominated wireless communication hardware. Since 2004, Catapult has grown into a C++/SystemC synthesis tool that supports virtually any FPGA or ASIC digital hardware design type. Catapult has been proven on numerous production projects to cut the overall design time in half.

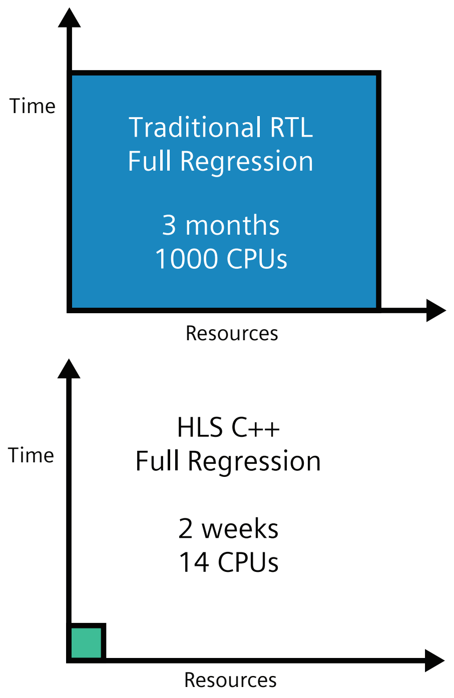
Catapult has been used on thousands of projects, and Catapult generated hardware can be found in hundreds of millions of cell phones, tablets, cars, computers, printers, cameras, gaming consoles, and satellites. Due to this extensive customer experience, Catapult has been optimized to work with existing RTL linting, coverage closure, synthesis and ECO flows. An example is illustrated in the whitepaper with Qualcomm, which demonstrates how Catapult and HLS works within very strict corporate RTL verification, synthesis and ECO flows.

Catapult is proven to easily adapt to last minute design changes

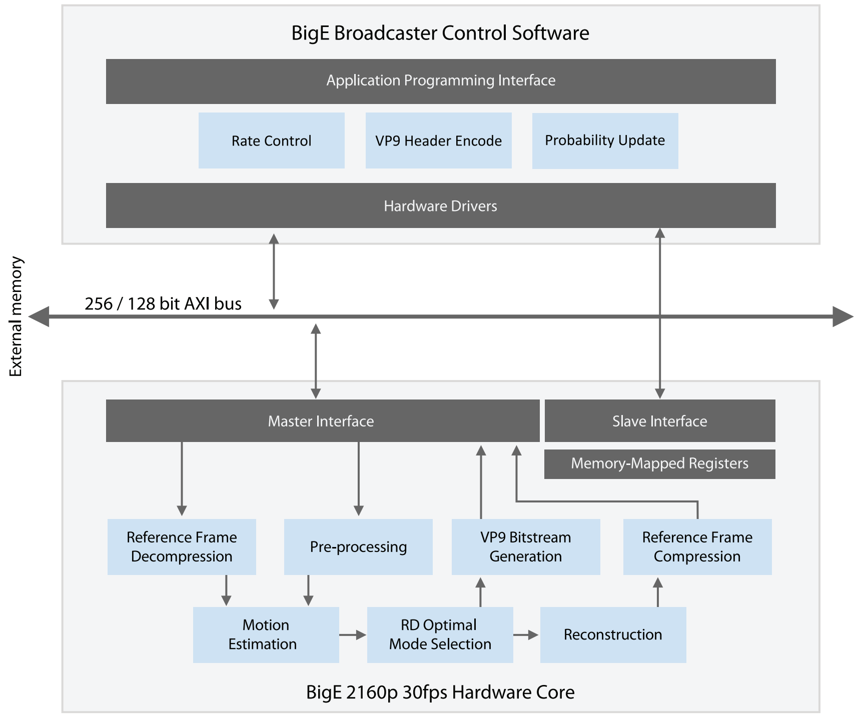
One of the biggest advantages of using HLS is the ability to quickly reuse or modify existing design functionality, something that is not possible to achieve in a traditional RTL design flow. Because the specifics of the timing, registers, datapath, etc are not contained in the source but specified during the HLS synthesis process, big changes in specification can be both implemented and verified while still staying on schedule. In a recent whitepaper from NVIDIA, they describe 2 changes that they were able to implement within months of RTL freeze that would have been impossible using a traditional RTL flow.

Catapult is proven for large designs

Catapult designs are often very large and complex subsystems. For example, Google used Catapult to synthesize their VP9 and AV1 video encoder and decoder, a complex, 8 million gate design with over 150 leaf blocks with an even mix of both datapath and control logic. To achieve this level of complex hardware Catapult has both top-down and bottom-up hierarchical design management capabilities. Designers can focus on the blocks they are working on while locking down other regions of the design, allowing an efficient design flow.



NVIDIA describes 2 changes that they were able to implement within months of RTL freeze

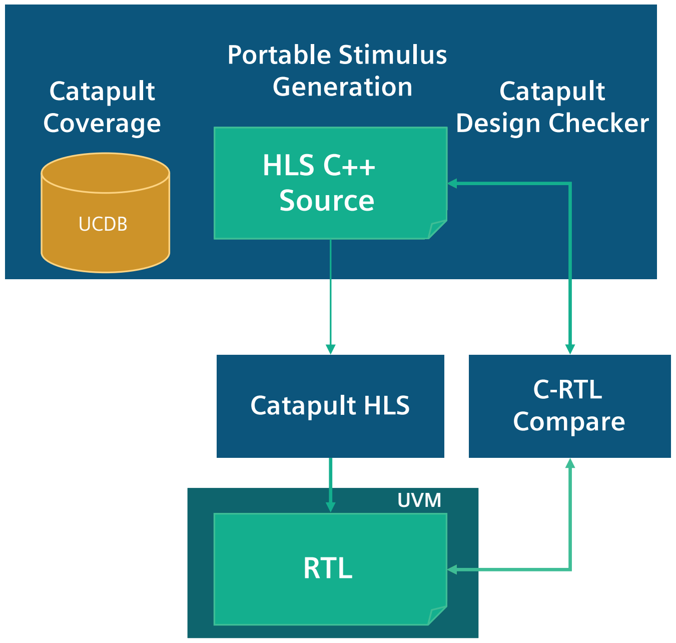


BigE is fully designed with an HLS flow.

Catapult High-Level Verification

HLS Verification (HLV)

The benefits for verification in an HLS design flow are numerous. HLS synthesizable C++/SystemC code is one fifth the number of lines of code compared to RTL which makes it easier to write and debug. The simulation speed is typically between 30-500X faster than RTL allowing much more verification and consuming far less compute resources. An HLS design flow also enables the verification team to be involved very early in the design process before any RTL is ready. All of this translates to enormous productivity gains when moving to an HLS/HLV design flow resulting in significant reductions in verification time and costs.



Catapult High-Level Verification and Synthesis

Catapult Design Checker to catch bugs early

C++ simulation often misses critical bugs such as Uninitialized Memory Reads, Out-of-bound array-accesses, and overflow/underflow problems, which can lead to hard-to-debug failures during both C++ and RTL verification. Furthermore C++/SystemC language has gray areas that may give unexpected simulation or synthesis results depending on what platform they execute on. The advantages of the Catapult Design Checker, because it uses formal technology at its core, is that bugs can be both found early in the design cycle and automatically, without a test bench, thus saving valuable debug time later in the design cycle.

Catapult Coverage to provide quality coverage metrics

As with RTL, HLS designers need a way to have metrics on the quality of the testing of their design. Catapult Coverage is HLS aware coverage for C++/SystemC that understands concepts such as function inlining and loop unrolling to provide a complete coverage picture (statement, branch, FEC and toggle) for the design. Catapult Coverage also writes its coverage data to the UCDB (Unified Coverage Database) that provides the user with a complete set of post-processing tools for merging, ranking, reporting and connecting to a testplan. Using Catapult Coverage on the HLS source enables achieving the equivalent coverage metrics automatically on the resulting RTL and closing RTL coverage.

Catapult Automatic C++ to RTL co-verification

Catapult provides an automated verification flow (SCVerify) that verifies the synthesized RTL against the original C++/SystemC design. This flow auto-generates a SystemC test infrastructure that reuses the original C++/SystemC testbench to verify the RTL, providing designers with a push-button unit test solution to quickly sanity-check the RTL so it can then be handed off to the downstream RTL integration and verification teams. For users of the UVM, Catapult also provides an automated verification flow that generates a complete UVM environment.

Catapult High-Level Verification

Formal equivalency checking

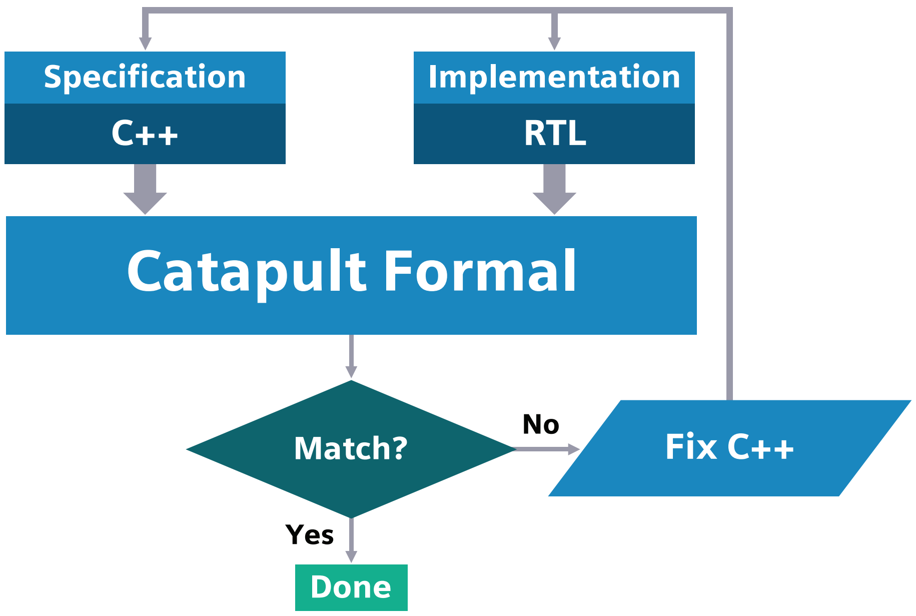
When designers use HLS to move their untimed C++ design to an RTL implementation, they may wonder if the timed RTL is exactly functionally equivalent to the original, high-level description.

Dynamic directed test verification provides good confidence that the RTL functions as expected vs the source C++. Whether external testbench, or SCVerify, the methodology for functional verification provides good confidence in functional correctness Catapult Formal verifies that Catapult C++ and RTL have functional equivalency without requiring simulation.

Mismatches are formally proven and flagged with counter-examples.

Formal stall checking

As part of a growing suite of HLS-centric formal apps, Catapult Formal provides formal stall proof to determine whether a design block can stall and therefore benefit from external clock gating.



C++ to RTL Formal Verification with Catapult Formal

Siemens Digital Industries Software
siemens.com/eda

Americas	+1 314 264 8499
Europe	+44 (0) 1276 413200
Asia-Pacific	+852 2230 3333

82981-C1 12/20 C