BOSCH VISIONTEC RAPIDLY BRINGS NEW AUTOMOTIVE IP TO MARKET USING THE CATAPULT HLS PLATFORM

JEAN-JACQUES AIME, BOSCH VISIONTEC
VISHAL SINHA, MENTOR A SIEMENS BUSINESS

Mentor®
A Siemens Business

WHITEPAPER

HIGH-LEVEL SYNTHESIS

www.mentor.com

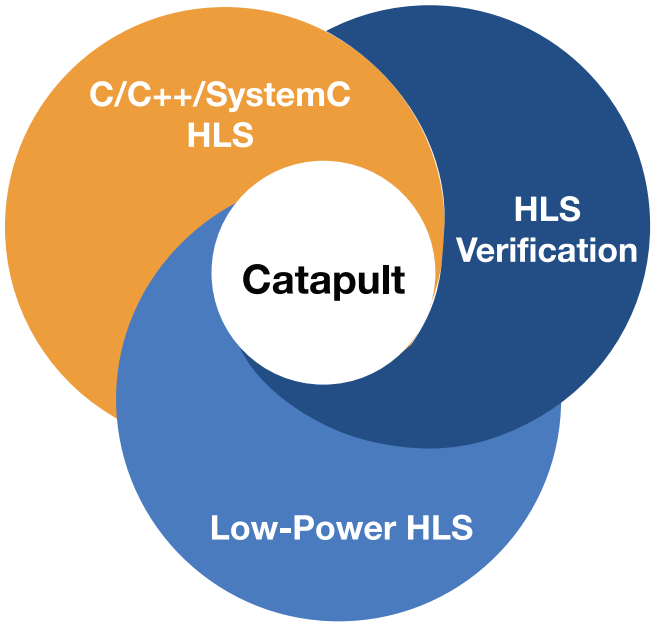
Source: WebTimeMedias

In the world-class setting of France's Sophia Antipolis science park, the BOSCH® Visiontec team innovates assisted and autonomous driving technology. This team develops state-of-the-art IPs and ICs containing high-performance processors that implement algorithms to recognize images from cameras in automobiles. They were tasked to create several brand-new designs that implemented mathematically-intense algorithms in less than a year. The specifications of these designs were dynamic and changed throughout the design cycle. There was no way that the team could manually code RTL and verify these designs in this ever-changing environment. So, they decided to move up to the C++ level and employ the Catapult® High-Level Synthesis (HLS) Platform and PowerPro® solutions from Mentor, A Siemens Business.

The Catapult HLS Platform empowers designers to use industry-standard ANSI C++ and SystemC to describe functional intent and move up to a more productive abstraction level. From these high-level descriptions, Catapult generates production-quality RTL. By speeding time to RTL and by automating the generation of bug free RTL, Catapult significantly reduces the time to verify RTL. The Catapult Platform provides a powerful combination of high-level synthesis paired with:

- PowerPro for measurement, exploration, analysis, and optimization of RTL power
- Verification infrastructure generation for seamless verification of C++ and RTL

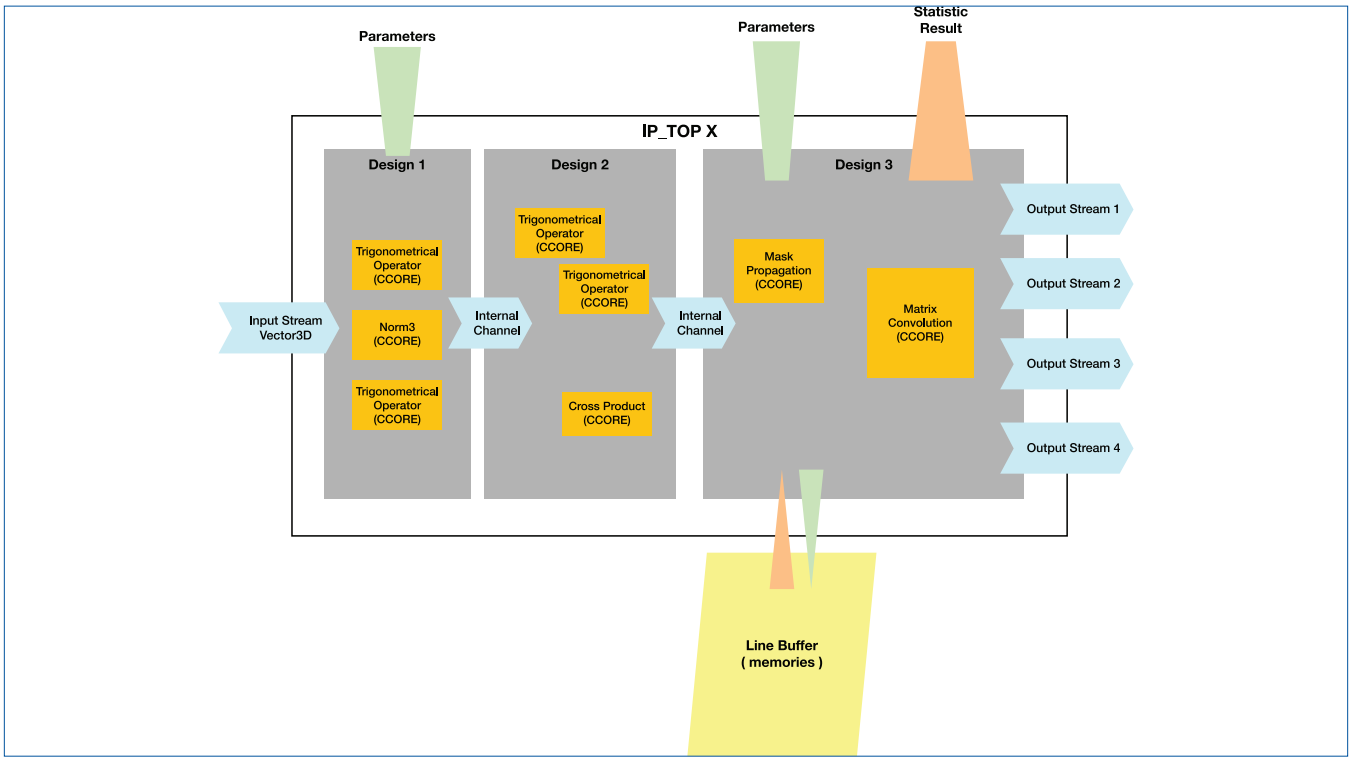
This paper presents how BOSCH Visiontec design team successfully met the challenges of creating their new designs, well within schedule.



INTRODUCTION

After an early evaluation, the Bosch team decided to use the Catapult HLS Platform for three image processing designs. Each algorithm to be implemented was first modeled in Matlab® and then handed over to the design team. It was the responsibility of the team to write the C++ code version of the designs. through overall algorithm implementation and IP creation, a Mentor consultant met constantly with the team to assess design architecture and performance options and to support C++ code transcription from Matlab. Catapult methodology has proofed to be easy to learn with a resulting fast ramp-up within Bosch team members.

After the C++ code establishment, Mentor support was of great help in synthesizing C++ code into RTL with desired performance. After the first three designs to be delivered with such support, BOSCH team took the expert/ leader role and on this basis two more designs were developed using the Catapult flow, with very little support.



THE DESIGN AND VERIFICATION FLOW

The design and verification flow includes:

- Writing and verifying the C++ code
- C++ synthesis and micro-architectural exploration
- Automating the verification flow

WRITING AND VERIFYING THE C++ CODE

The design and verification flow (Figure 1) begins by writing synthesizable C++ inspired from the Matlab model and from the specification.

While writing C++ code, the team ensures that it remains in the defined hardware architecture. For example, the design is partitioned into hierarchical blocks. Writing synthesizable C++ code based on the defined hardware architecture in the beginning of the flow, avoids multiple iterations during synthesis in order to meet performances goals. The team performs extensive verification in the C++ domain, including bit-accurate checks against the reference Matlab models.

All of the designs are new and rich in complex mathematical functions, either trigonometric, scalar, vector or even matrix multiplication and normalization. The team utilizes these functions in multiple designs, so they create parameterized functions to avoid writing the same functions multiple times for different designs. This technique helped save time and to meet precision goals when bit-width changes occurred during the algorithm evolution. Writing these functions in C++ is much easier than writing them in RTL. Catapult can automatically meet timing for these functions as compared to RTL where designers need to ensure timing of these functions after RTL synthesis.

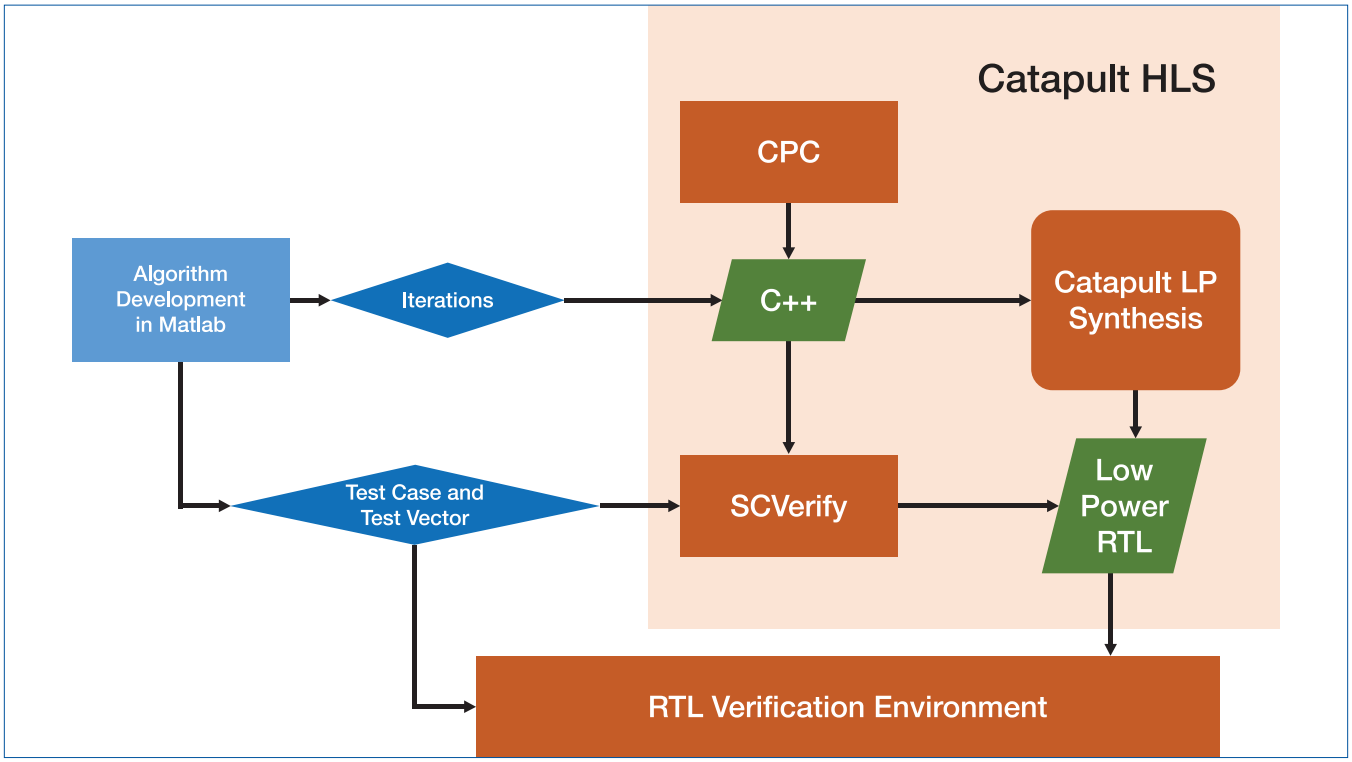


Figure 1: The design and verification flow

C++ SYNTHESIS AND MICRO-ARCHITECTURAL EXPLORATION

After the team verifies the C++ code, it is synthesized into RTL using Catapult high-level synthesis. However, at very early stage, some of the designs had been synthesized prior to verification because of the unavailability of reference vectors from Matlab. This is not recommended in the Catapult flow for final RTL, but in this case, it helped to freeze the constraints for the design. After verification, very little change was required to set up the constraints.

All the algorithms kept evolving during the development stage. So, it was necessary to quickly adopt these changes while generating RTL. With Catapult, the team handles these changes by simply changing constraints in the tool and re-running it to see what the resulting design looks like. This allows them to validate the algorithms and to see the effect on area and performance due to the changes. This is a key differentiator from traditional RTL design where any change in algorithm requires code changes and a lot of subsequent verification.

Using Catapult, the team could quickly make changes with very little modification to the architecture. For example:

- The micro-architecture constraints on the algorithms had been set at the early stage of development. But, during exploration, the team can make constraint changes as the algorithms evolve without changing the C++ code. This results in dramatically cutting development time as compared to working at the RTL.

  Some of the basic constraints, loop rolling and initiation intervals changed during the development cycle due to performance requirements. For example, in one design, the throughput requirement changed from 5 to 20 clock cycles. This was achieved by simply constraining the initiation interval to 20 and setting the proper rolling value at the inner loops. This also automatically allows resource sharing and it reduced overall area of the design.

- There were a few iterations in the target library and in the frequency of operation of the designs. The team found that Catapult provides a large advantage in this scenario. For example, the team could characterize new libraries within few days and re-port the design without any change in the directives.

- The clock frequency was modified and Catapult synthesized the C++ code without a change in the directives and the post-synthesis timing met specification.

In all these cases, manually recoding the RTL would have taken days and even longer to determine if the re-architected result met timing specifications. It only took hours with Catapult.

AUTOMATING THE VERIFICATION FLOW

The SCVerify verification flow (Figure 2) in Catapult automatically generates the infrastructure for verifying the functionality of the HLS-generated RTL against the original source code and it reuses the original C++ testbench.

In addition to the SCVerify flow, the team also used their own verification environment where C++ and the generated RTL could be verified simultaneously. Going forward, they can eliminate this flow and replace it with SCVerify.

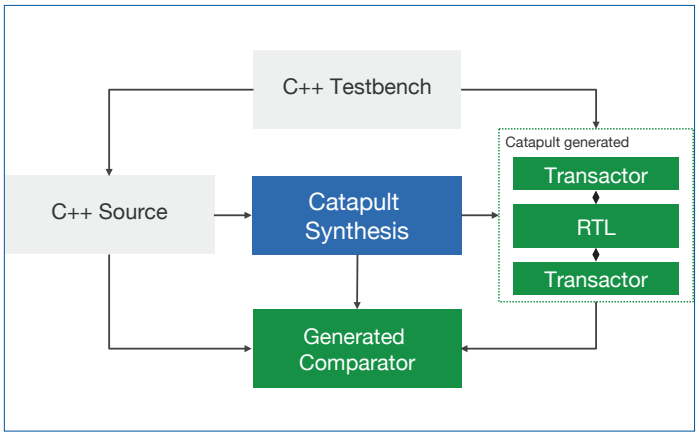


Figure 2: The SCVerify flow

An important step of the verification process is to write a C++ test-bench according to specifications and to define reference test vectors to verify the design. When Catapult compiles C++ design files, it also generates scripts that drive the simulator for debugging and verifying the design. SCVerify provides a big advantage to the team for verifying the RTL because with a push of a button it uses the same testbench written to verify C++ code to verify RTL code without writing a RTL testbench. The SCVerify flow saves considerable verification time compared to a manual flow.

DOWNSTREAM TOOLS & DESIGN FOR TEST (DFT) COVERAGE

After working the designs through the Catapult design and verification flows, the team synthesizes the RTL using a 3rd-party tool, targeting their chosen technology. For each design, all signoff criteria were fully met, as Table 1 shows.

	Area with 16nm (mm2/Kgate count)	Frequency (Mhz)	Throughput (clk cycle)	Latency (clk cycle)	RTL Synthesis	Formal Checks	DFT Coverage
IP_TOP1	0.013 / 71	250	25	55	Timing Met	Pass	>99%
IP_TOP2	0.136 / 738	250	20	110	Timing Met	Pass	>99%
IP_TOP3	0.10 / 500	250	25	131	Timing Met	Pass	>99%
IP_TOP4	0.25 / 1400	250	2	154	Timing Met	Pass	>99%

Table 1: Post-synthesis results

99 percent DFT coverage is mandatory for these safety-critical automotive designs. The team uses a third-party DFT tool to determine coverage. At low clock frequency, the DFT tool initially reported coverage of 95 percent. However, using the Catapult LOGIC_OPT directive further increased to coverage to 97 percent. This directive allows the tool to increase the area of the design in order to improve reported coverage. However, the coverage was still below 99 percent.

An initial idea was to increase the clock frequency using the CLOCK_OVERHEAD directive to insert more test points in the design. However, this solution was not optimal as it increased the number of registers in entire design. So, as a proof of concept during the initial design phase, the team created functions for the low coverage elements using sequential Catapult Optimized Reusable Entities (CCOREs) and increased the clock overhead. By adopting this methodology, the team reached over 99 percent coverage with minimum increase in register area. However, during a later design phase, the clock specifications were modified from 200 MHz to 250 MHz, so increasing the clock overhead for the CCOREs was not needed.

USING CATAPULT PROPERTY CHECKING (CPC) FOR VERIFICATION

During the latter part of the project, the team employed CPC for all of the designs. CPC automatically finds bugs prior to synthesis, saving weeks of verification debugging time. CPC automatically identifies and formally proves hard to find issues like:

- UMR (Uninitialized Memory Reads): checks for uninitialized memory reads.
- DBZ (Divide by Zero): checks for divide-by-zero violations.
- ABR (Array Bounds Read Error): checks arrays, bit-vectors, and memories in the design for out-of-bounds read access violations.
- ABW (Array Bounds Write Error): checks arrays, bit-vectors, and memories in the design for out-of-bounds write access violations.

- CAS (Incomplete Case Statements): checks for incomplete switch/case statements.
- ISE (Illegal Shift Error): checks for illegal shift operations.

In addition to automatic checks, CPC also formally proves custom assertions and cover points that the team writes to complement dynamic simulation and to provide comprehensive verification of the C++ models (Figure 3).

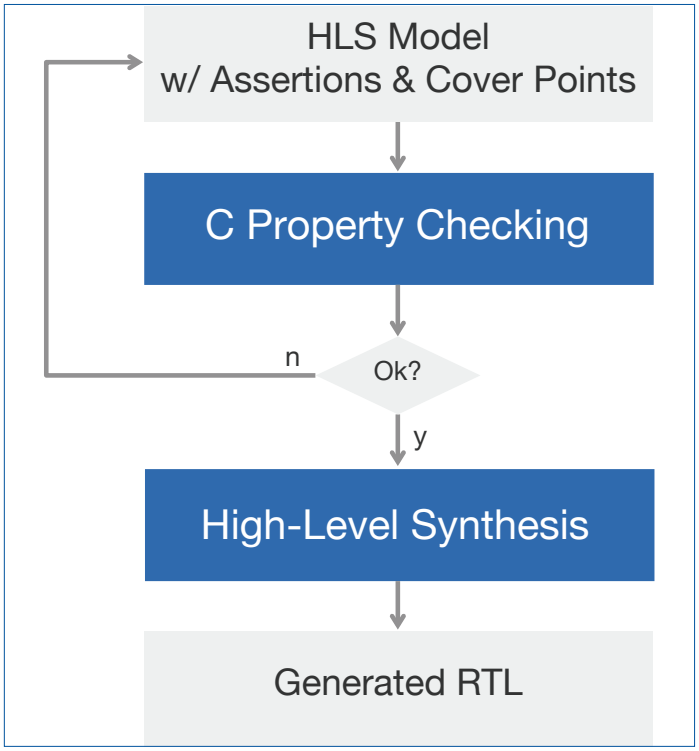


Figure 3: The CPC flow

Using CPC, the team was able to find and fix UMR violations (Figure 4), where one array was read before writing, which were impossible to detect in C simulations.

```
# ================================================================
#                    Status of Property Checks
# ================================================================
#                                Total  Assumed Verified Violated
# UMR (Uninitialized memory read)   122        0      122        0
# ABR (Array bounds read error)      69        0       69        0
# ABW (Array bounds write error)     16        0       16        0
# DBZ (Div/0, Mod/0)                  0        0        0        0
# ISE (Illegal shift error)           0        0        0        0
# CAS (Incomplete case statement)     0        0        0        0
# ASC (Assertion check)               0        0        0        0
```

Figure 4: All violations detected and fixed prior to synthesis

USING THE CATAPULT LOW-POWER FLOW

Next, the team investigated using the Catapult low-power flow (Figure 5).

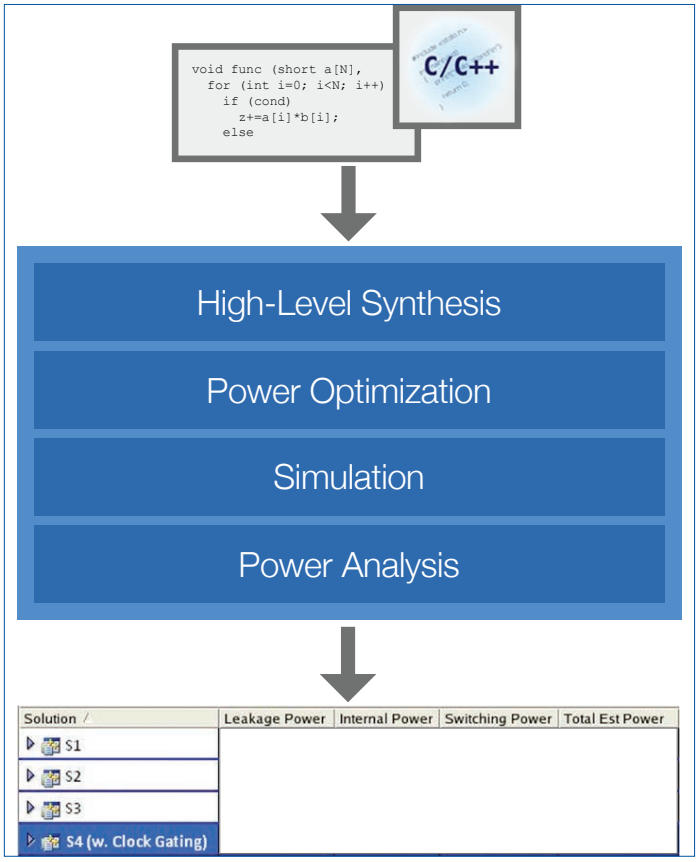


Figure 5: The PowerPro flow

The Catapult low-power flow integrates high-level synthesis with PowerPro technology to provide RTL power optimization. In addition to PowerPro optimization capabilities, Catapult runs supplementary optimizations to control power consumption of the RTL when using the HLS flow.

PowerPro uses deep sequential analysis of the RTL for estimation and optimization of power. Using stability and observability clock gating, the tool generates power-optimized RTL with minimum guidance. It also uses the same testbench used in the SCverify flow to capture switching activities in the Switching Activity Interchange Format (SAIF) or Fast Signal DataBase (FSDB) format. The team does not need to write a separate testbench or create an environment for capturing switching activities.

The built-in formal verification tool insures that resulting, power-optimized RTL remains equivalent to the previous RTL. In order to calculate power savings, the team first estimates power at the RTL without power optimization and again after using the Catapult low-power flow. The team uses the same testbench and test vectors that were used in SCverify flow to verify C++ and C++ to RTL. Table 2 shows the results.

	Catapult HLS	PowerPro	Savings
IP_TOP1	<1.50mW	<1.00mW	19%
IP_TOP2	<20.00mW	<15.00mW	33%

Table 2: Comparison of power savings between generated RTL and power-optimized RTL

CONCLUSIONS

By adopting the Catapult C++ HLS flow with power analysis and optimization, BOSCH Visiontec team was able to successfully deliver the new designs ahead of schedule in seven months even though the specifications evolved over the design cycle. The quality of results was not only maintained but improved by using Catapult's micro-architectural exploration, which enabled the team to quickly produce higher quality designs through continuous refinement.

Without using the Catapult flow, it would have been impossible to complete the project on time and to specifications. Because of the team's success, this flow will be used for future new designs and for updated designs that target new standards or process technologies.

Early evaluations show that using the Catapult low-power flow yields an overall reduction in power of 30 percent. And, CPC found design issues that would have been impossible to detect. Therefore, the team has decided that the lower-power flow and CPC will become mandatory going forward. The team are now experts in using the Catapult methodology and they are delivering their own designs without help.

For more information about the Catapult and PowerPro solutions, visit: https://www.mentor.com/hls-lp/

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 09-17 TECH15970-w