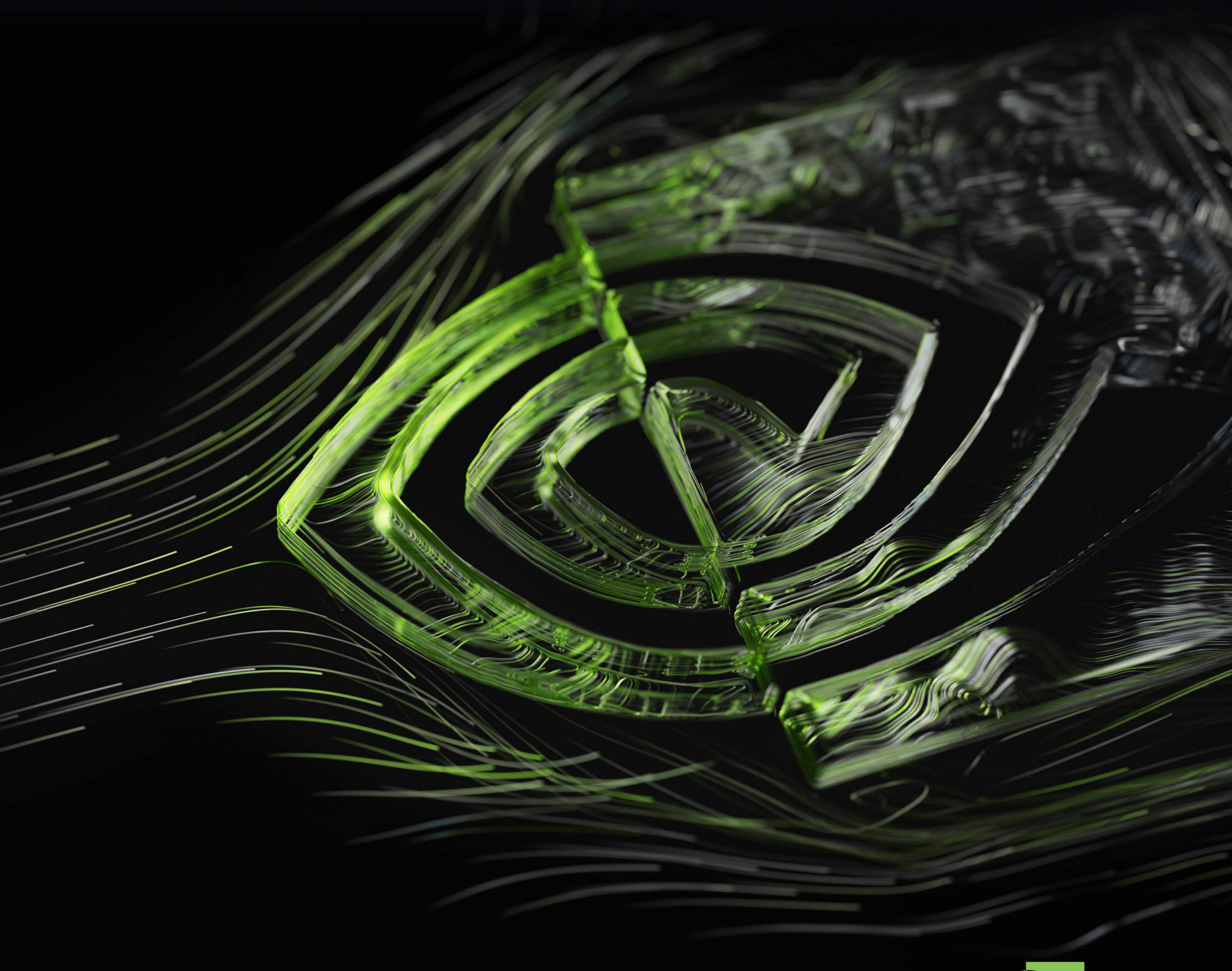
AdaCore CASE STUDY

NVIDIA: Adoption of SPARK Ushers in a New Era in Security-Critical Software Development



adacore.com

NVIDIA

Summary

NVIDIA Corporation, the world's best-known maker of graphics processing units (GPU), is also among the most trusted names in embedded systems, high-performance computing, and artificial intelligence. As a supplier of critical hardware, software, and firmware components across numerous sectors, they are recognized for tackling some of the tech industry's toughest problems.

Security is essential to NVIDIA's brand. However, with cybersecurity risks rising across the board, including in the verticals they serve, the company was facing the challenge of delivering more secure products without incurring a large increase in development time and cost.

An increasingly hostile cybersecurity environment

Several converging macro trends are currently putting pressure on the makers of embedded systems. Customers are demanding they demonstrate air-tight security in their products.

First, cyberattacks against firmware and hardware are on the rise. Hackers are targeting the lower levels of the technology stack hoping to exploit vulnerabilities that are difficult and expensive to correct. Due to the ubiquity of embedded systems, it is often very costly to update firmware and hardware in the field.

Second, the development and verification ecosystem hasn't kept pace with the scale of these attacks. Many languages and toolsets are insufficiently robust for critical embedded applications; they provide no guarantees during development that security vulnerabilities have been eliminated. For example, the C programming language is often preferred for embedded systems because it's great for writing fast and efficient code. But code developed in C is hard to get perfect and C lacks many verification capabilities.

Third, an industry-wide lack of secure code designers and software security practitioners is making project timescales problematic. These valuable resources have become extremely difficult to find. The cybersecurity professional organization (ISC)^2 recently calculated the global cybersecurity workforce shortage totalled some 2.7 million unfilled positions at the end of 2021. [1]

The convergence of these trends is causing extreme concern in safety-critical sectors such as medical devices, robotics, and automotive. OEMs are demanding a high degree of security assurance from their suppliers. And they have become very interested in the methods and techniques being used in secure environments.

[1] *(ISC)^2 Cybersecurity Workforce Study, 2021,* (ISC)^2, March 2022.

Customer:
NVIDIA Corporation is among the world's most trusted names in graphics processing units, embedded systems, high-performance computing, and artificial intelligence.

Challenge:
To find a more measurable solution for verifying embedded system security and safety in an increasingly hostile threat environment.

Solution:
SPARK - a programming language designed to facilitate the use of formal methods to guarantee the correctness of software with mathematics-based assurance.

Result:
NVIDIA is now using SPARK to achieve and demonstrate the absence of runtime errors in its most critical components and to prove conformance to specifications for its root-of-trust applications.

NVIDIA

Company Snapshot

NVIDIA has over **25,000+** employees

Operates in **50+** locations

Reported **$26.9 billion** revenue in FY22

A preference for measurable outcomes

In response to these concerns, NVIDIA examined all aspects of their software development methodology, asking themselves which parts of it needed to evolve. They began questioning the cost of using the traditional languages and toolsets they had in place for their critical embedded applications.

For years, they had typically used C and C++ for these applications. And they were still suffering from the decades-old challenges endemic to those languages, including ambiguities in the C and C++ specifications that limit the effectiveness of C and C++ analysis tools. When there is no certainty of verification, you never know when you're done testing a particular code module. Security assurance not only isn't certain, progress towards it isn't measurable under those circumstances.

"We like measurable outcomes," said Daniel Rohrer, VP of Software Security at NVIDIA. "We'd like to be able to ship products into the field and know what the customer experience is going to be before they see it. One of the key challenges we placed in front of the team is how to turn this from a non-measurable, where we can't even answer 'Are we done yet,' to a more measurable answer."

NVIDIA's software security team examined various methods and strategies that offered measurability. They soon recognized that the bases of many of these methods were mathematical formal methods and the use of formal provers. They also discovered that these tools had undergone a major evolution over the past decade or so.

"We wanted to emphasize provability over testing as a preferred verification method," said Rohrer. "Testing security is pretty much impossible. It's hard to know if you're ever done." Formal proof, on the other hand, offered guarantees that were very attractive to NVIDIA. These include the absence of runtime errors and common security vulnerabilities, and also proof that a program's functionality conforms to its specification.

A "heretical" proposition

NVIDIA began to ask themselves, "Can we change our fundamental approach? And if so, which of these tools can we actually use?" They put these questions to their software security team.

In reply, the security team came back with what Daniel Rohrer characterized as "a fairly heretical" proposition: "What if we just stopped using C?"

"SPARK has a capability that is not found in most other programming languages. That is the ability to specify program requirements within the code itself and use the associated set of tools to ensure that the implementation matches its requirements. Essentially you are proving your programs are correct. That's a very powerful capability."

**- Dhawal Kumar,
Principal Software Engineer, NVIDIA**

This led to other questions, like, "What alternative languages and tools are available to support the use of formal methods?"

In trying to answer those questions, NVIDIA discovered SPARK.

SPARK and its benefits

SPARK is a high-level computer programming language consisting of a well-defined subset of Ada. Like Ada before it, SPARK was designed for the development of high-integrity software used in systems where predictable and highly reliable operation is essential. SPARK uses a language feature known as "contracts" to specify components in a form that is suitable for static verification using formal methods.

"From the perspective of programming language capabilities, the paradigms are very similar to C and C++," said Dhawal Kumar, a principal software engineer at NVIDIA and one of their first SPARK users. "SPARK is an imperative programming language. You can write procedure-oriented or object-oriented code, and there are other facilities for programming in the large."

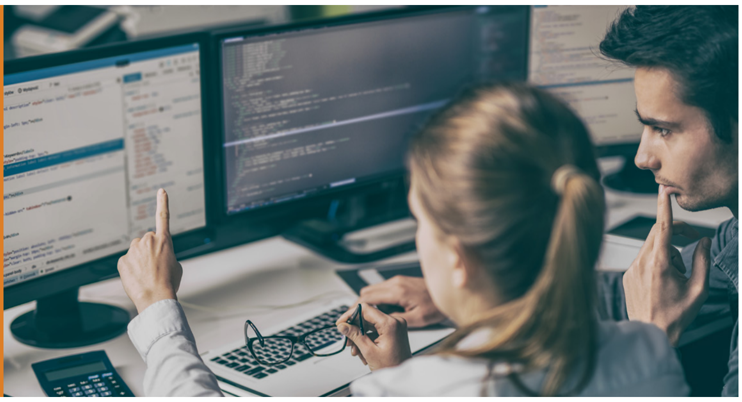
Thanks to its design, there can be no undefined behaviors in code developed with SPARK. The language comes with a set of built-in checks that make sure all its rules are obeyed, so run-time errors (such as buffer overrun) cannot occur.

Another key feature of SPARK is that it supports formal verification. In other words, through the use of SPARK and formal methods solvers, it is possible to prove mathematically that your SPARK code behaves in precise accordance with its specification. This process is known as formal verification.

"SPARK has a capability that is not found in most other programming languages," said Dhawal Kumar. "That is the ability to specify program requirements within the code itself and use the associated set of tools to ensure that the implementation matches its requirements. Essentially you are proving your programs are correct. That's a very powerful capability."

“AdaCore appears very focused on solving customer’s problems first. We know we can count on AdaCore to answer our questions with very low latency.”

- Dhawal Kumar,
Principal Software Engineer, NVIDIA



Advantages of SPARK

Daniel Rohrer listed several SPARK advantages that were important to NVIDIA.

First, SPARK is deterministic. This enables formal proofs to guarantee the absence of runtime errors and specification compliance.

Next, SPARK code is linkable with C. This meant NVIDIA would not have to recode everything at once or recode non-critical components. SPARK components are easily integrated within C/C++ environments. This allowed NVIDIA to wisely choose targets that warranted the investment in formal verification.

Finally, SPARK has a credible ecosystem. Ada, SPARK’s basis, is a very mature language. It has been upgraded several times since the 1980s. World-class tool support is readily available through AdaCore—the world’s foremost authority in Ada and SPARK—and other organizations as well. Plus, Ada and SPARK have an upstream Open Source Software (OSS) presence. So, while NVIDIA was glad to have AdaCore as a willing and responsive partner, they knew that even without AdaCore they could carry their SPARK initiative into the future.

Successful proof of concept

In the final quarter of 2018, NVIDIA conducted a proof-of-concept (POC) exercise to delve deeper into the SPARK language, the associated tools, and the available technical support; the purpose was to confirm their general fitness for NVIDIA’s purposes.

For this initial POC, the NVIDIA software security team applied SPARK to two applications. The first was a bare metal application that acts as a root of trust for code running on several other security processors. The second was a Real-Time Operating System (RTOS) application that handles the resizing of protection regions.

In only three months, the small POC team was able to convert nearly all the code in both codebases from C to SPARK. In doing so, they realized major improvements in the security robustness of both applications. Thanks to the soundness of SPARK and the ability to express program properties as contracts, NVIDIA found that reliance on scarce security practitioners could be significantly reduced. They also found that the developer teams adapted readily to the new language and tools.

Reviewing the results with their safety assessor, NVIDIA also found that SPARK can be safely mixed with C in a safety context. What’s more, the safety assessor deemed SPARK not only acceptable but preferable over C/C++ for safety-critical applications, provided developers have sufficient training.

Evaluating return on Investment (ROI) based on their results, the POC team concluded that the engineering costs associated with SPARK ramp-up (training, experimentation, discovery of new tools, etc.) were offset by gains in application security and verification efficiency and thus offered an attractive trade-off.

Finally, NVIDIA concluded that SPARK’s support infrastructure was excellent.

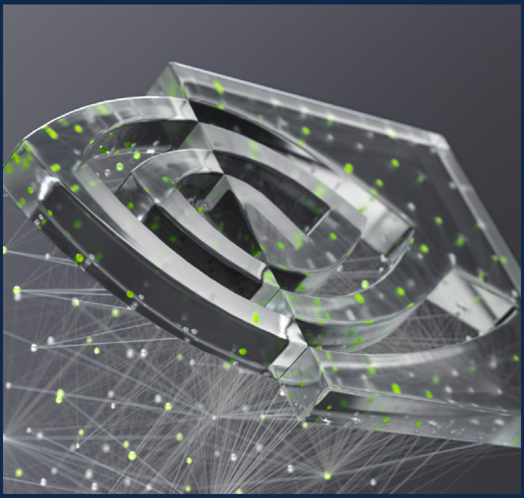
Dhawal Kumar, who led the POC, called AdaCore’s support system world-class. “AdaCore appears very focused on solving customer’s problems first,” he said. “We know we can count on AdaCore to answer our questions with very low latency.”

Daniel Rohrer acknowledged their POC put AdaCore in a tough corner, but AdaCore’s tools and support team responded admirably. “We were doing low-level firmware in a custom ASIC with registers and side effects that had to be modeled. Our problems were not uniform and generally not in AdaCore’s sweet spot,” said Rohrer. “Throughout the course of the POC, however, both the tooling and the support really lived up to the challenge. Surprisingly so.”

Coming out of the POC, NVIDIA felt quite confident that the savings and benefits from using SPARK would continue, while the added ramp-up costs would gradually disappear as SPARK expertise grew within the company.

"The main reason why we use SPARK is for the guarantees it provides. One of the key values we wanted to get out of this language was the absence of runtime errors. It's very attractive to know your code avoids most of the common pitfalls. You tend to have more confidence when coding in SPARK because the language itself guards against the common, easily made mistakes people make when writing in C."

- James Xu,
Senior Manager for GPU Software Security, NVIDIA



Adopting SPARK for NVIDIA's most security-critical development projects

NVIDIA began implementing SPARK in its security strategy in 2019 on select pieces of firmware. They began training additional personnel in SPARK and eventually developed an in-house training program.

Several NVIDIA teams are now using SPARK for a wide range of applications that include image authentication and integrity checks for the overall GPU firmware image, BootROM and secure monitor firmware, and formally verified components of an isolation kernel for an embedded operating system, to name just a few.

In general, their targets tend to be smaller code bases that would benefit the most from SPARK's strong typing, absence of runtime errors, and in some cases, rigorous formal verification of functional properties

Proof of absence of runtime errors

When asked to quantify the benefits they've gained from adopting SPARK, NVIDIA engineering managers spoke of the guarantees SPARK provides and of increased confidence in the quality of their products.

James Xu is the Senior Manager for GPU Software Security at NVIDIA. His group is charged with making sure software and firmware in NVIDIA's GPUs are shipping with industry-leading security postures. They also drive innovative security and privacy technologies that help make NVIDIA's products unique.

"The main reason why we use SPARK is for the guarantees it provides," said Xu. "One of the key values we wanted to get out of this language was the absence of runtime errors. It's very attractive to know your code avoids most of the common pitfalls. You tend to have more confidence when coding in SPARK because the language itself guards against the common, easily made mistakes people make when writing in C".

"It's very nice to know that once you're done writing an app in SPARK—even without doing a lot of testing or line-by-line review—things like memory errors, off-by-one errors, type mismatches, overflows, underflows and stuff like that simply aren't there," Xu said. "It's also very nice to see that when we list our tables of common errors, like those in MITRE's CWE list, large swaths of them are just crossed out. They're not possible to make using this language."

Varun Kumar manages the GPU Firmware Security team at NVIDIA. His group is responsible for components that initialize and boot up the GPUs, functions like secure boot, firmware update, and device recovery attestation. He now has 20 engineers building components in SPARK to NIST SP800-193 platform firmware resiliency guidelines. He says the qualities Xu mentioned are also attractive to their customers' security personnel.

"The fact that we can confidently say that we don't have any off-by-one errors, overflows, underflows, etc., and are backed by a formally verifiable method (tool), provides good guarantees about the security and robustness of the program." said Varun Kumar.

Formal verification—without code bloat or performance degradation

Cameron Buschardt, a Principal Software Engineer at NVIDIA, is responsible for the architecture of an embedded operating system. For his application, he needs a deeper level of proof. Beyond the absence of runtime errors, he needs formal verification: demonstrable mathematical proof that the critical portions of his code are functionally identical to the corresponding portions of his model.

"We have customers who review our security parameters very closely. Just being able to say, we formally verified our code—we didn't just run some bug checking hunting tool, we formally verified it—that's huge. The high level of trust this evokes drastically reduces review burden and maintenance efforts. It's huge for me and also for our customers."

Unlike James Xu and Varun Kumar, whose groups are using SPARK on new components, Cameron Buschardt was able to compare his newly-written SPARK modules with equivalent modules in C. He found no code bloat, nor did he see any degradation of performance. "Looking at the assembly generated from SPARK, it was almost identical to that from the C code," he said. "I did not see any performance difference at all. We proved all of our properties, so we didn't need to enable runtime checks."

Mitigated concerns and rising confidence

As one might expect when an organization tries a new (and possibly a bit heretical) way of doing things, many of NVIDIA's developers and engineering managers were initially skeptical of SPARK.

"The skepticism was due to a variety of concerns depending upon whom you talked to, whether they were in engineering or management," said James Xu. "Ada and SPARK are less well-known languages in some industries, which led to many questions: What is our ability to scale up engineering resources? How will it impact our development schedules? How do we find help? The absence of runtime errors sounds nice, but do we really get that in practice? When it comes to logic bugs, how does SPARK help us then? Plus, given the unfamiliarity and the restrictions of the language, there was a perception of a significantly longer development time compared to writing in C."

The answers NVIDIA has gotten to those questions have overcome much of that skepticism.

For training, NVIDIA initially took advantage of AdaCore's courses, but it didn't take them long to develop their own in-house training program. They found their developers ramp up quickly.

"We have customers who review our security parameters very closely. Just being able to say, we formally verified our code—we didn't just run some bug checking hunting tool, we formally verified it—that's huge. The high level of trust this evokes drastically reduces review burden and maintenance efforts. It's huge for me and also for our customers."

- Cameron Buschardt
Principal Software Engineer, NVIDIA

Schedule impact turned out to be far less significant than imagined. For example, James Xu had guessed development in SPARK would take twice as long as in C. That turned out not to be the case when care is taken to apply SPARK to well isolated and smaller (but more valuable) areas. He characterized the actual impact as "a minor annoyance."

"Some of us were concerned, initially, about how much schedule overrun there might be, or how we were going to fund these projects," said Xu. "But that turned out not to be a problem."

Varun Kumar also said that finding help has not been a concern, thanks to AdaCore's support system.

Finally, SPARK has delivered on its "absence of run-time errors" promise.

"It's nice to have confidence that your code base won't be plagued with common errors," said Xu. "About 70% of the common CVEs simply can't occur in SPARK. So, when you talk to an external auditor who looks at your code, for example, you can deflect some basic concerns they often have. When they ask if they have to read the code line by line to see if you've got misused pointers, overflows in your indexes, and stuff like that, you can just categorically say, 'Don't bother. Those things can't exist. You don't need to look at the codebase.'"

Improving customer relationships

Cameron Buschardt says the guarantees SPARK provides make discussions with customers over security far easier.

"It's far easier to focus discussions in a massive component if you're simply able to say, 'Hey, look, we've got this tool. We were able to prove these properties, let's focus on other areas of security.' That, for me, is a big motivating factor, because that's not going away: customers look at our security story. With SPARK, it's much easier to sell."

Former skeptics have become advocates

Seeing firsthand the positive effects SPARK and formal methods have had on their work and their customer rapport, many NVIDIA engineers who were initially skeptical have become enthusiastic proponents.

"I'll be honest; at the beginning, I was very, very skeptical," said Cameron Buschardt. "My first attempts to prove non-trivial algorithms in SPARK were disastrous. But after we got past the initial learning curve, I was shocked at what we could prove.

"SPARK promises to let us implement a component, prove it, and be done with it," he continued. "This is transformative and it evokes a level of trust no other language provides."

Daniel Rohrer indicated that Buschardt is just one of many who have shed their skepticism and embraced SPARK.

"There have been others who, as we've engaged through this process, were initially detractors but have subsequently become champions," Rohrer said. "For those who've looked at the challenges that are facing us and were willing to face them head-on, once they got past the initial inertia of 'Hey, this is the way we've been doing it the last 20 years,' it's been transformative."

> **"I'll be honest; at the beginning, I was very, very skeptical. My first attempts to prove non-trivial algorithms in SPARK were disastrous. But after we got past the initial learning curve, I was shocked at what we could prove.**
>
> **SPARK promises to let us implement a component, prove it, and be done with it. This is transformative and it evokes a level of trust no other language provides."**
>
> **- Cameron Buschardt**
> **Principal Software Engineer, NVIDIA**

Rapid growth

The use of SPARK and the formal methods tools built for the language has grown and spread rapidly within NVIDIA since their initial deployment.

At the conclusion of their initial POC at the end of 2018, NVIDIA had five developers trained in SPARK. By the second quarter of 2022, that number had grown to over fifty. During that same period, NVIDIA implemented numerous components in SPARK. These include many components of their GPU firmware image, portions of their hardware boot ROMs, and several libraries to simplify the discharge of proofs for the kernel of an embedded operating system.

Many NVIDIA products are now shipping with SPARK components, and awareness and interest in SPARK continue to grow within the company.

"We initially created a small proof-of-concept to convince ourselves and subsequently implemented a larger piece of the boot firmware in SPARK. That proved to other teams within NVIDIA that SPARK is a viable option for firmware or similar use-cases."

"It's coming up more often in architectural discussions," says Buschardt. "We're seeing security meetings where we'll talk about a critical security property that a customer is interested in, and we'll come back to that theme: we can show our customers proof that this does exactly what the spec says it does."

Overall, NVIDIA has demonstrated groundbreaking leadership and innovation in the software security domain. They had a very challenging goal and took an ambitious path to accomplish something that had never been done before in the semiconductor industry. For four years, they have successfully demonstrated time and time again that their choice to adopt SPARK was the right one - and have paved the way for anyone interested in following NVIDIA's lead.

AdaCore

adacore.com