White paper

Model-based HW/SW inline IPSec microarchitectural design and validation using Intel® CoFluent™ Studio

Abstract

In this paper, a team of Intel networking engineers presents a flow used to design and validate the microarchitecture of a new packet-processing module for Intel® chipsets, based on models captured using Intel® CoFluent™ Studio (CoFluent™).

With this design and validation flow, the team created an executable specification of the new packet processing module. The team first validated the functional behavior and the performance of the module for different use cases. The executable specification (a CoFluent model) helped identify and solve architectural issues before and during hardware/software development. The team then reused the same CoFluent model as a device model in order to accelerate the development of software in a Wind River Simics* virtual platform.

"Our design flow, based on Intel® CoFluent™ Studio models, enabled us to validate our complete product solution requirement from the SW application level down to microarchitecture, and do this early in the design cycle."

– Andrew Cunningham,
Senior Digital Design Engineer,
Intel Corporation

Andrew Cunningham
Intel Corporation

Patrick Fleming
Intel Corporation

Jérôme Lemaitre
Intel Corporation

Ireneusz Sobanski
Intel Corporation

Chris M. Wolf
Intel Corporation

Background

IPSec is a standard network protocol used to securely transfer packets over the Internet. To cope with today's stronger requirements for network security, the cryptographic algorithms of the IPSec protocol require more computing power and system bandwidth. However, when run on a CPU, these algorithms/tasks require so much computing power and system bandwidth that they can reduce the performance of the network connection. To avoid this situation, we can reduce the load placed on the CPU by offloading computationally intensive tasks of the IPSec protocol to our network interface controller (NIC).

In this white paper, a team of Intel networking engineers presents a flow used to design the microarchitecture of a new 50 Gbps (inline) module in a NIC for Intel® chipsets. This flow is based on models captured using the Intel® CoFluent™ Studio (CoFluent™) system-level modeling and simulation toolset.

Market overview

According to Gilder's law, the total bandwidth of communication systems triples every twelve months. Moreover, protecting the information that is exchanged over the internet becomes more important with every passing month. For these reasons, it is critical to be able to design an architecture that can meet IPSec security requirements, while still optimizing CPU load and system bandwidth.

Challenges

When relying on text-based specifications reviews, complex dynamic operational issues are usually discovered later in the development cycle. The time lag and feedback loop from initial specification to discovery of issues can have a significant impact in many areas of the development cycle. In turn, this can increase time-to-market and development costs for the final product.

Once a microarchitecture specification is validated, the pre-silicon development phase starts. The software/firmware (SW/FW) development phase can start even earlier, before pre-silicon development, by using a virtual platform based on general expectations of the microarchitecture specification. However, a device model of the new module is still required in order to start the SW/FW development phase in a virtual platform.

Objective

Our objective in presenting a design and validation flow is twofold. First, we want to validate the specification of the inline module for different workloads/use cases; and identify and solve potential issues earlier, before starting the development of the module.

Second, we want to accelerate the creation of an inline device model so that we can start the SW/FW development phase earlier in our virtual platform project than a traditional approach would allow.

Solution overview

We used the CoFluent toolset to create an executable specification of the microarchitecture of the new inline module in a NIC for Intel chipsets. The tool automatically generates SystemC* code, starting from simple graphics, C/C++ code, and timing annotations. Using this kind of executable model removes the ambiguity of a text-based specification, and therefore increases the overall quality of the system specification. Thanks to executable specifications, we can also validate complex dynamic use cases earlier, and reduce the risk of finding operational issues later in the product design cycle.

Figure 1 gives an overview of the flow we used to design and validate the microarchitecture of the inline module.

Once the executable model is validated, it becomes a live microarchitectural specification for the register transfer level (RTL) design team. Moreover, we can also integrate the microarchitecture executable specification into a Wind River Simics* (Simics) virtual platform as a new device model. This integration enables us to develop and validate SW/FW before the RTL module is available. An additional benefit is that reusing a unified executable microarchitecture model helps to formalize communication between teams.

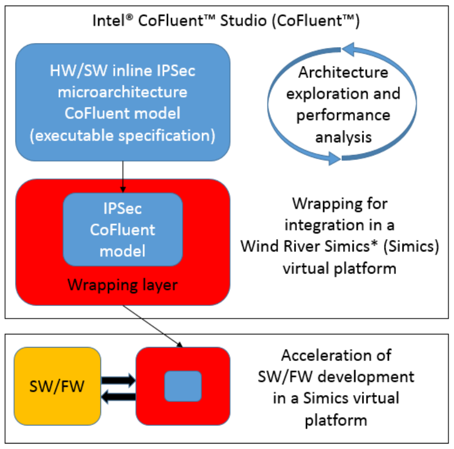
Intel® CoFluent™ Studio (CoFluent™)
HW/SW inline IPSec microarchitecture CoFluent model (executable specification)
Architecture exploration and performance analysis
IPSec CoFluent model
Wrapping layer
Wrapping for integration in a Wind River Simics* (Simics) virtual platform
SW/FW
Acceleration of SW/FW development in a Simics virtual platform

Figure 1. Overview of our flow, which was used to design and validate the microarchitecture of our inline module based on models

Usage and results

The microarchitecture of the inline module is modeled as a hierarchical composition of more than 20 blocks. These 20+ blocks have been captured and validated in separate CoFluent models. Each of the 20 blocks is itself composed of several hierarchical blocks.

We also used CoFluent to capture a simple testbench around this microarchitecture model in order to validate the behavior and performance of the inline module. The testbench includes a packet producer and a packet consumer.

The packet producer stimulates the inline module with multiple sequences of uniquely identifiable, time-stamped Ethernet packets whose payload size is parameterized. These packets are processed by the inline module, and are ultimately received by the packet consumer.

The packet consumer then scoreboards all the data it receives, by checking this data against the expected data. The packet consumer also generates latency and throughput measurements for each packet. Those measurements enable us to verify whether the inline module meets the performance requirements for complex use cases.

Architecture exploration

When we simulated the SystemC code that CoFluent automatically generated for our model, our testbench enabled us to identify a major performance issue in the inline module. We discovered this issue when stimulating the module with specific payloads, as shown in Figure 2. The advanced exploration and analysis features that are available in CoFluent helped us to find out that the problem was due to a microarchitectural issue.

We quickly captured a new executable specification of the microarchitecture. We then used our testbench to validate the new executable specification via simulations to make sure that the new architecture met all performance requirements.

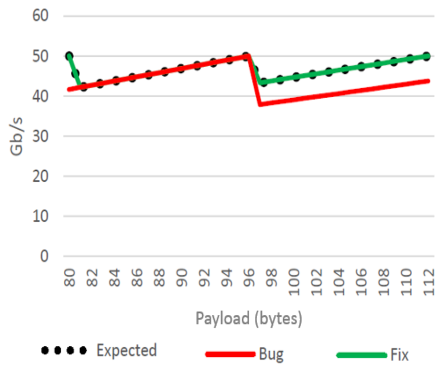

Figure 2: We identified and solved a major performance issue, thanks to our executable Intel® CoFluent™ Studio inline model

Without the CoFluent executable model, this particular issue would not have been discovered until later in the design process. For example, this kind of issue would not typically be identified or resolved until the HW/SW development phase. Moreover, the new microarchitecture we identified via modeling required 9% less physical silicon area than the microarchitecture we considered in the first place.

FW/SW development acceleration

To be integrated in our Simics virtual platform, our inline module first had to be integrated into a module that was already used as a SystemC module in the virtual platform. Since the interfaces of this SystemC module were not standard SystemC TLM interfaces, we had to develop a wrapping layer around the SystemC code that was generated for our inline module.

The wrapping layer converts the inter-faces and protocols that are imposed by the existing module into the interfaces and protocols that are used by the Inline module; and vice versa. Moreover, we had to develop this wrapping layer so as to support multiple outstanding transactions, as well as support out-of-order completions. We created a specific testbench that generates the appropriate traffic in another CoFluent model to develop and validate the wrapping layer and the processing of outstanding transactions and out-of-order completions by the inline module.

After the integration of the wrapped inline module into our Simics virtual platform, we were able to start the development of SW/FW three months before the RTL inline module was available. Using this virtual platform, we found and solved more than 40 issues before the HW/SW development phase began.

Conclusion

The team of Intel engineers used a system-level modeling flow to design and validate the microarchitecture of a new packet-processing module in a NIC for Intel chipsets, based on models captured using Intel CoFluent Studio.

With this modeling flow, the team created an executable specification of the architecture of the new module. The executable specification was then used to validate the functional behavior, and to predict the performance of the module for different use cases. The executable model enabled the team to identify and solve microarchitectural issues before and during HW/SW development. The team then reused the same CoFluent model to successfully accelerate the development of SW/FW around this module in a Simics virtual platform.

"The Intel® CoFluent™ Studio model was utilized by multiple teams working on the inline IPSec module, from the architecture team, through pre-silicon design and validation, to SW/FW teams. It provided a common base for all design activities."

– Ireneusz Sobanski,
Senior Pre-Silicon Validation/Verification Engineer, Intel Corporation

Key results

- We identified a major performance issue, and proposed a new architecture that solved the issue and saved 9% silicon area in our inline module for Intel® chipsets.
- Using the executable model, we found and solved more than 40 issues before RTL coding even began.
- By integrating the Intel® CoFluent™ Studio (CoFluent™) model in a Wind River Simics* virtual platform, we were able to begin development of SW/FW 3 months before RTL was available.
- We were able to reuse the same SystemC code that was generated automatically from the inline CoFluent model in our virtual platform. This helped to formalize communication between teams.

For more information on Intel® CoFluent™ Studio, visit

www.cofluent.intel.com

Printed in USA XXXX/XXX/XXX/XX/XX Please Recycle XXXXXX-001US