

Gabe on EDA

Published on Gabe on EDA (<http://www.gabeoneda.com>)

Specification vs. Modeling: The Real Language Question

By Simon Napper
Jul 13 2007 - 3:54pm

Simon Napper, CEO, Synfora, Inc.

Is ANSI C or SystemC the best language for electronic system level (ESL)? The debate rages but EDA language wars almost never reach a conclusion. We all want a single language for every stage of the design, but the range of systems make this unlikely. Instead of an ESL Esperanto, multiple languages have been developed that need to co-exist and interoperate for the many aspects of integrated circuit (IC) design and verification. The language debate seldom defines its scope, and rarely gives specific examples of where a particular language can be used. You must define the area you are working in, before you can decide which language is best.

Today, a typical consumer system-on-chip (SoC) comprises different types of intellectual property (IP), performing different tasks. In the digital domain, at the highest level, you can break these down into:

- Complex application engines (video codecs, wireless modems): These define the IP's functionality, are critical for differentiating the end product, and change rapidly with each revision. Significant investment is continually being made to improve the power, performance and area (PPA) in application engines. This type of IP is based on reference algorithms that are already in C and are simulated and verified by architects who work in C.
- Star IP such as CPUs and DSPs: These need significant investment in terms of building the hardware as well as the creation, debugging and compatibility of the software. This type of IP is usually hand-crafted, built bottom-up, doesn't often change, and is very hard to alter when it does.
- Connectivity and Control IP such as USB and DMA: This is system level glue that never defines the functionality nor differentiates the end product. It does sometimes need a limited amount of tailoring. It can be purchased or created from RTL and from higher-level languages that are very close to the actual hardware implementation.
- Memory: This takes up the largest amount of silicon area, but also neither defines the function nor differentiates the end product. Memories are almost always compiled and built and verified bottom-up. Their models are generated from the transistor level behavior.

The insatiable demand for faster, cheaper products drives increasingly complex consumer SoCs. But although the number of IPs will continue to grow, consumer SoC architecture will remain similar to its current form.

Within the defined scope of consumer SoCs, the language argument needs to be broken into the ability to fit in the top-down design implementation/synthesis flow and the simulation/verification flow. Based on the breakdown above, we can see that it is imperative to separate the languages capabilities and to rank them according to the main tasks they must accomplish:

Capture and implement complex sub-systems: As the algorithms that constitute the foundation

and specification already exist in C and were created by system engineers, ANSI C is the natural language for this task. Any other language requires the manual creation of yet another specification and the resulting burden of equivalence proof. Whatever ANSI C's perceived limitations for hardware, it has often been used to create complex, efficient hardware from an untimed ANSI C source.

In future, designers will need to reduce the number of blocks they create. For example, an imaging pipeline with 15 blocks that, today, are made individually and manually assembled and tested, must become one engine that is automatically assembled and tested – especially when some small feature change effects multiple blocks in the pipeline. Many of the attributes of SystemC – such as the ability to specify detailed hardware behavior or to create multiple threads -- rapidly become liabilities when the designer wants to create complex hardware quickly.

Model hardware-level behavior: Whether the IP blocks come from algorithmic descriptions, as in complex application engines, or are built bottom-up, as is the case for memory or Star IP CPUs or DSPs, a unique and interoperable modeling description language is needed. SystemC is a widely used backplane for system level modeling where the different types of IP are assembled. As such, the creation of SystemC transaction level models (TLMs) is an important part of the system modeling and verification process. Those are built bottom-up for the memories and Star IP CPU/DSP and ideally are automatically created along with the RTL from the algorithmic level descriptions for the complex sub-systems. The automatic generation of SystemC TLMs gives the verification team an early start.

No single language can describe every type of IP on any type of IC. Each IC design organization has to determine which types of IP are critical to it and prioritize time and resources accordingly. But as we look at the evolution of consumer SOC architecture, we see that ANSI C is the language of choice for the specification and implementation of application engines which are set to dominate the design effort and the system differentiation. SystemC is the modeling language and is better suited as an output language for high-level synthesis systems rather than an input language.

There is an urgent need to divide and conquer the complexity of today's SoCs. This means using whichever language is best for a given requirement, while ensuring that a methodology exists to tie everything back together at the highest level.

To comment on this article send email to:gmoretti@gabeoneda.com [1]

Source URL:

<http://www.gabeoneda.com/node/82>

Links:

[1] <http://www.gabeoneda.com/mailto:gmoretti@gabeoneda.com>