Accelerating software development for high performance compute using VDKs

Ravi Kumar[1], Rae Parnmukh[2], Troy Jones[2] and Jon Taylor [3*]

[1]RISC-V Engineering, Tenstorrent
[2]IP Product Team, Tenstorrent
[3] Synopsys Inc

Abstract

This presentation discusses how Tenstorrent and Synopsys co-developed virtual design kits to accelerate software development for high performance compute. It will describe the steps taken to virtualize the Tenstorrent design and ultimately achieve Linux boot to enable further testing of compute peripherals. The work provides a proven reference flow for software development that will benefit the RISC-V ecosystem.

Introduction

Growing system complexity is an ever-increasing problem for SoC designers. While CPU operating frequencies are no longer increasing, the demand for compute still is, especially for AI applications. This can be addressed through use of parallelism with many cores, or application specific accelerators. This growing system complexity creates two specific types of challenges. First, system architects have to prove cohesion of hardware and software architectures before triggering silicon workflows. Second, obtain pre-silicon confidence that hardware supports all software functionality, and software optimally utilizes all hardware features. Prevalent simulation and emulation platforms are unacceptably slow and/or expensive for these purposes and software development and system validation overflow into post-silicon phase fand software maturity ultimately becomes the long pole for getting to production. Worse, intractable problems usually show up late in the game, further increasing time to market.

For pre-silicon software development virtual prototypes are becoming invaluable platforms. These are fast fully functional system models that enable pre-silicon development and help mature unmodified production code. These models are available independent of hardware-based models. and provide a completely parallel software development platform. These platforms utility continue post-silicon since they are, by construction, light weight, easier to introspect, experiment, and instrument.

Synopsys' ***Virtualizer***[1] tool provides a complete virtual prototyping environment to accelerate presilicon software development activities ahead of hardware availability. Virtualizer models are written in SystemC[2] and are TLM[3] compliant. Virtualizer provides both a complete development environment with a rich library of fully functional components as well as debug and analysis tools. All these are ensconced in a ***Virtual Development Kit*** (VDK) which can be distributed to a larger audience. Thus extending software collaboration from hardware and software architects all the way to end customers.

Recently Tenstorrent and Synopsys have collaborated to generate VDKs for Tenstorrent's RISC-V AI computers. This presentation aims to provide the high-level steps taken to create the Virtual models and VDKs, challenges and initial outcomes.

Model Overview

The advent of chiplet based architecture is poised to enable very large silicon systems in a package (SiPs) composed of modular silicon components called chiplets. AI SiPs are typically composed of CPU, ML, memory and scale out chiplets. An abstract model of the SiP is shown below.

The virtual prototype models will follow this modularization and create chiplet models. The modeling abstracts out datapath complexities like caches, fabrics and die-2-die subsystems, but retains software relevant components like CPUs, mechanisms involving security, interrupts, address translation mechanisms etc. As an example, the CPU system's model is shown below. The green blocks are fully software faithful models (ISA, CSRs etc.) while the blue blocks are approximate and simplified models. Full CSRs and address space are modeled, but the CSRs are non-functional.

Virtual Models and VDK Creation

The most complex models are the processor cores. The CPU chiplet uses Tenstorrent Ascalon[4] IP (an RVA23 compliant RISC-V core) as well as multiple smaller cores such as the RocketCore[5] (an RV64G RISC-V variant). The Virtualizer tool incorporates the Imperas Fast Processor Models (ImperasFPMs) generator to auto generate these models. providing automatic ways to keep the model

updated and future-proofed. Similar generators provide all other processors including the Secure Processor.

Other RISC-V compliant modules like interrupt models (APLIC, ACLINT), and IOMMUs are developed within the Virtualizer development environment. (The Virtualizer IDE is very similar to Eclipse)

The next set of models are built around the processor cores. Components needed like AXI transactors, DMAs, RAMs, ROMs, DRAMs, PCI etc. Virtualizer libraries provide ready-to-use models of all these.

The models thus created are integrated to provide the final CPU chiplet model.

Similar workflows create the other chiplets needed and integrated to create the final Virtual model.

Challenges:

- While RISC-V has a well-defined set of standard extensions, implementations can still use some degree of optional and/or custom extensions. While the cores to model are standards compliant, several features are subject to configuration options and vendor specific features like machine mode constraints, optional interrupts/exceptions, cache control registers, custom debug functionality, ISA extensions etc. which must be accurately modelled. Further the features will change over time. Using a model structure with well-defined extension APIs such as Imperas Fast Processor Models (ImperasFPMs) makes it straightforward to add optional extension/custom features while not risking introducing bugs in the core model.
- Ascalon is a highly configurable CPU; both in terms of high-level features such as number of cores in a cluster and decoder width but also features such as level 1 & level 2 cache sizes, system interfaces and physical addressing capabilities. As with optional extensions and custom instructions, it is important that the model can accommodate these flexible configurations to support whatever the customer requires.
- Similar challenges exist for the Virtual library components which might not have full featured models or even miss new models of emerging components, like latest versions of PCI. Virtualizer provides plug in capabilities to receive such custom created components.
- Integration of dozens or hundreds of models to create the final suite of VDK is significant effort. Virtualizer development environment allows individual models to be unit tested continually and integrated (CI/CD flows) to enable fast maturity and deployment.

Outcomes:

We are in the early stages of this project.

- Starting with the ImperasFPM RISC-V RVA23 ISA base model, Tenstorrent's CPU featuring a few optional extensions, the ImperasFPM was extended to produce a complete Ascalon and RocketCore models.
- These were integrated into the CPU chiplet subsystems which in turn were integrated to create the full CPU chiplet model and made available to software architects.

Booting Linux

The first step to enabling software development is to be able to boot Linux on the CPU Chiplet.

Challenges:

- The boot image aspirationally needs to run unmodified on the CPU chiplet VDK. This VDK should include all the models, memories, CSRs etc. To make this seamless. This includes careful attention to the model's performance so that it does not degrade simulation speed. Ideally, we need the VDKs to runs 100's of time faster than eventual RTL emulation/simulation
- The ensuing debug and development of boot code does require VDK specific code for instrumentation and introspection features which are not part of the mission code and need to be carefully sequestered.
- We do foresee this effort to provide collateral to be made eventually available to RTL maturity. High judgement is needed to incorporate debug information and traces that can be used for this purpose.
- There is an added difficulty in creating and interface so that any physical devices that might need to interact with the outside world such as a mouse, a keyboard, sensor devices, etc. Can connect to the model.

Current Outcomes:

- We successfully booted Linux (linux-6.12.1) on the CPU chiplet model.
- No custom tool chains were needed. Open-source GNU RISC-V cross-compiler was used to build software images that ran on the virtual prototype.
- Boot times are acceptable (minutes).
- Workflows are in development to allow software to enhance boot code, add firmware and requisite drivers.
- Secure Linux boot is being debugged.

Future Work

Tenstorrent and Synopsys continue to further enable early software development beyond the CPU chiplet model. Tensix NEO is the AI IP anchored in RISC-V's open architecture which is the building block of the AI chiplet. Work continues to provide the VDKs to create the Al chiplet to enable software development to support AI workflows.

References

[1] https://www.synopsys.com/verification/virtual-prototyping/virtualizer.html

[2] SystemC is **a** C++ class library for system modeling/simulating at a higher level of abstraction providing higher level abstractions for hardware than Verilog or VHDL. (https://www.systemc.org)

[3] TLM is a systemC coding standard focusing on non-cycle but transaction level accurate modeling of system busses/networks. Levels of accuracy consist of loosely timed (LT) and approximately timed (AT) modeling styles. https://www.accellera.org/images/downloads/standards/systemc/TLM_2_0_LRM.pdf

[4] https://tenstorrent.com/en/ip/tt-ascalon

[5] https://chipyard.readthedocs.io/en/latest/Generators/Rocket.html