Virtual Platforms Driving Software Quality in Pre-Silicon

Amith David Pulla

amith.pulla@intel.com

Abstract

Over the last two decades, tech industry has seen increasing pressure on software-development schedules and quality, this is true even for electronics design, device software, BIOS, firmware, device drivers, operating systems, hypervisors and end-user application development. Today in the semiconductor industry, majority of companies are adopting some type of system prototyping strategy, mainly in the form of simulators and emulators. The system level simulation and emulation tools available to software development teams can accelerate software development, testing, debugging and end-to-end validation in a virtual lab setting, driving the overall quality of the hardware and software platforms. Simulation tools are often referred to as virtual platforms because they are a fully software based solution running on operating systems. Virtual platforms or virtual prototyping platforms are the new class of tools based on high-level of design abstraction, can enable software development teams to start development and validation way ahead of actual hardware or silicon availability. Pre-silicon software enabling and validation is becoming the standard in compute platform development lifecycle in the semiconductor industry. Enabling the software stack in pre-silicon has a huge impact on quality of the software stack, reducing the number of hardware software integration issues in post-silicon and improving the time-to-market for end-to-end compute platforms. This paper explores the use of software based simulators and emulators for software/firmware development and validation, including industry leading simulation and emulation platforms, common virtual prototyping approaches, advantages and constraints. The paper will also cover how these tools and platforms help software teams to be more agile, responding to the changing business needs and quality expectations by continuous feedback mechanisms in software testing and validation processes.

Biography

Amith Pulla is a Technical Program Manager at Intel Corp, currently working at the Intel site in Hillsboro, Oregon. Amith works with engineering teams developing virtual platforms in the pre-silicon software space. Amith works with platform software enabling plans in pre-silicon to scope virtual platform features and capabilities. Over the last 18 years, Amith has been involved in software testing strategies and processes for applications from sales, marketing to data analytics and system software. Amith also worked in market research and intelligence space for data center and server market. Amith has worked extensively on projects involving multiple platforms and complex architectures. Amith worked on developing test methodologies and techniques to meet the business needs and schedules. As part of his QA lead role for the first 8 years of his career, Amith focused on improving and refining QA processes and standards for efficient software testing in agile environment. Amith also took on ScrumMaster roles working with agile development teams.

Amith has an M.S. in CIS from the New Jersey Institute of Technology; Amith got his CSTE and PMP certifications in 2006, CSM certification in 2012.

1. Introduction

With the increasing pressure on software-development schedules, the cost of software has become the undisputed leading factor in electronics design. Software development has been changing the electronics industry, semiconductor companies are constantly looking for product differentiation through software. To increase the software development windows and shorten the time to market (TTM), hardware platform development teams have started looking towards simulation and prototyping tools to begin the software stack development and validation before the real hardware is available. This new category of tools are sometimes referred to as virtual prototyping platforms or simply virtual platforms.

Virtual platforms are based on a high-level of design abstraction. Software development on virtual platforms is mainly focused on embedded software like BIOS/UEFI, firmware, device drivers, operating system enabling and diagnostics tools, but it's increasingly getting used for application development and workload optimization in Pre-silicon. Embedded software development teams need an accurate model of the hardware to test and validate software components, on the other hand hardware designers need complete software stack to fully validate the hardware. A hardware platform may contain a combination of CPUs, memory, storage, application specific integrated circuit (ASIC), SoC etc.

Software development has fundamentally changed the shape of the electronics industry, from large integrated device manufacturers (IDMs) to a complex disaggregated design chain of heavily interacting companies. Hardware intellectual property (IP) providers deliver blocks and subsystems to semiconductor companies, who in turn deliver chips and board subsystems to system houses who increasingly seek differentiation through software. All of the hardware-related players interact with software providers who create everything from low level drivers to complex end-user applications. Virtual platforms footprint in pre-silicon software development is growing steadily.

There are two types of models used in Pre-Silicon software development the cycle accurate models and functionally or register accurate models. The cycle accurate models or prototypes are built from real hardware with number of field programmable gate arrays (FPGAs) or pre-built solution like hardware emulator. These hardware-based solutions are ideal for silicon logic validation but typically very expensive and not very flexible for software development and validation. Virtual platforms close this gap by offering software designers early access to a development platform. A Virtual Platform is a software based system that simulates your full target System-on-Chip or board. Virtual platforms shift the software development earlier in the product development lifecycle and are cost effective compared to hardware equivalents.

Virtual Platforms

Architectural Level Accuracy

SW/FW Development and Testing

2. Physical Hardware and Software Stack

A compute system will consist of physical hardware and layers of software running on it. Let's focus on some of the key components of physical hardware that make up a compute platform.

2.1.1 CPU/Processor

A central processing unit (CPU) is the key component of the computer responsible for carrying out the instructions of a computer program. A CPU performs basic arithmetic, logical, control and input/output (I/O) operations that the computer programs instructs. A typical CPU consists of arithmetic logic unit (ALU) that performs arithmetic and logic operations, processor registers that holds data that is being processed and a control unit that coordinates the fetching of data from memory. CPUs have now become microprocessors where all components are integrated as a single integrated circuit (IC) chip. When an integrated circuit contains a CPU plus other components such as memory, peripheral interfaces, it’s called a systems on a chip (SoC). In a multi-core processor where single chip contains multiple CPUs also known as cores, the chip is considered as a socket.

2.1.2 Memory

Memory in computing refers to the integrated circuits that store data for immediate use by a CPU; memory should operate at a high speed and this is the reason it’s mostly referred to as random-access memory (RAM). Memory in computers are primarily silicon-based transistors on a chip. This semiconductor memory can be further classified as volatile and non-volatile memory. Flash memory, ROM, EPROM and EEPROM that is used for storing system firmware like BIOS or UEFI can be considered as non-volatile memory. Whereas dynamic random-access memory (DRAM), static random-access memory (SRAM) are considered as volatile memory.

2.1.3 I/O

Input/output or I/O is the communication between a computing platform and other computing platform or a human. Inputs are the data received and outputs are the data sent out from a computing platform. Transfer of data to CPU and memory and back to a storage device like a disk drive is considered I/O, for example by reading data from a disk drive, is considered I/O. Memory-mapped I/O (MMIO) and port-mapped I/O (PMIO) are methods that can be used for carrying out input/output (I/O) between the peripheral devices and computing platform like the CPU. I/O can also be performed by dedicated I/O processors that can execute their own instructions.

2.1.4 Storage:

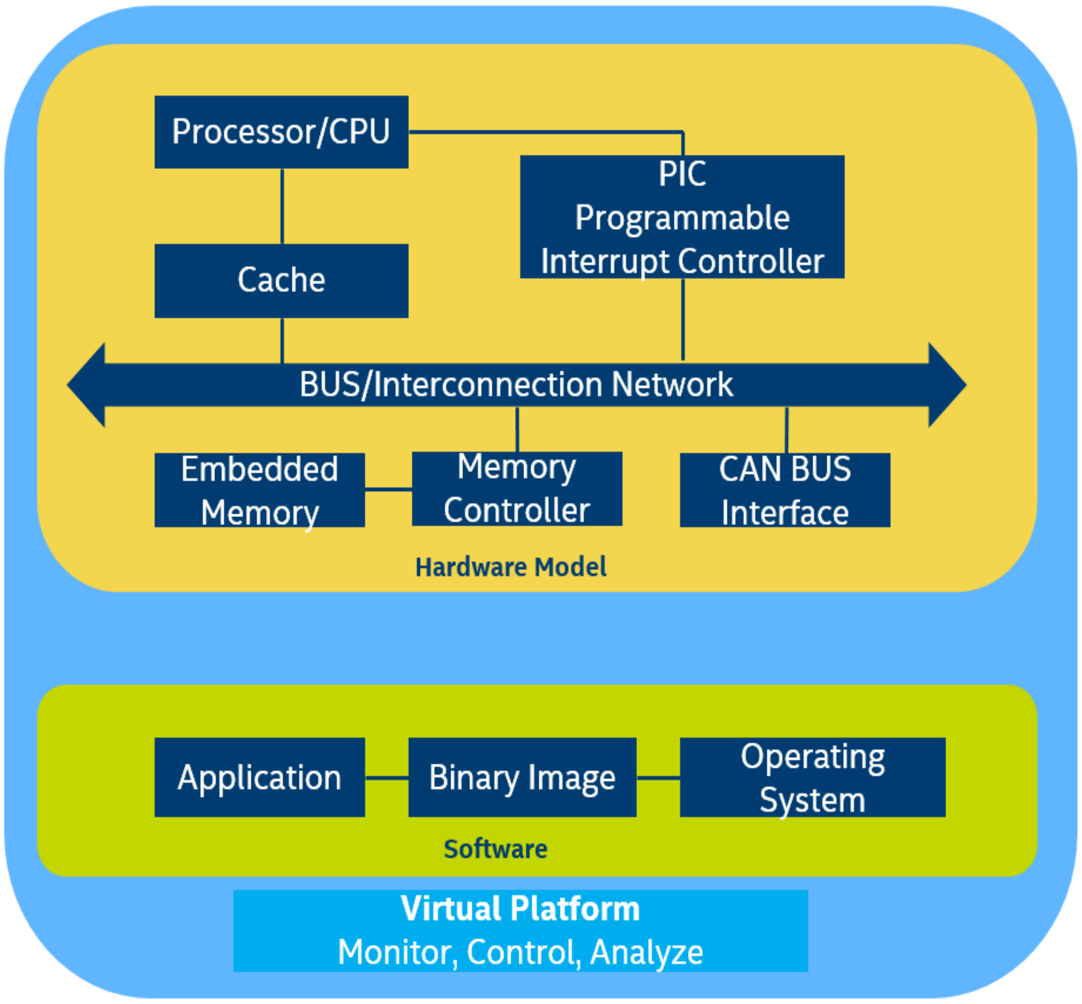
A storage device is a piece of computing hardware for storing and extracting data, it can be stored permanently or temporarily. Storage devices are part of the compute platforms and can be connected using several different options.

2.1.5 Board:

Board connects the central processing unit (CPU) memory and other devices on the platform, the board also provides connectors to peripherals that connect to the main compute platform. This is the main printed circuit board (PCB) found in microcomputers, it’s also referred to as mainboard, system board or logic board.

2.1.6 Platform:

A platform is the overall computing system where software applications are executed. It can include the hardware and all the software/firmware ingredients including BIOS/UEFI, IP specific firmware, device drivers. Sometimes Operating system (OS), application programming interfaces (APIs) and applications are considered as part of the platform.



2.2 Software Stack

2.2.1 BIOS/UEFI:

BIOS is the Basic Input/Output System (BIOS) firmware traditionally used to define the interface between the platform hardware and the operating system. Unified Extensible Firmware Interface (UEFI) replaced the BIOS in most of the modern computing platforms. UEFI covers the legacy support for BIOS enabled services. UEFI can have advanced features like remote diagnostics and maintenance that can be accessed independent of the operating system (OS), even without the OS being installed. Features like Reliability, availability and serviceability (RAS) are enabled in UEFI. RAS can protect data integrity, increase uptime making the platform more resilient and fault-tolerant.

2.2.2: Firmware and Device Divers

Firmware is a specific class of software often embedded in the hardware. Firmware is designed and developed for specific hardware to provide low-level control of the devices hardware capabilities and features. Firmware is typically installed in the non-volatile memory devices on the platform like the flash memory, ROM or EPROM. Each device or hardware capability on the platform may run its own firmware. All platform firmware needs to be developed and tested before the completed platform can be released. Device drivers enable a specific devices connected to the platform, the drivers installed on the platform controls that specific device. The driver is the software interface to underlying hardware enabling the OS to access hardware features. Drivers are specific to hardware and operating systems that allow connected devices to work with multiple versions of hardware OS combinations. Like the firmware, devices drivers are also software that needs to be developed and tested as part of the overall platform validation plan.

2.2.3: Operating System (OS)

An operating system (OS) manages the hardware and all the other software resources in a computer. The OS is initially loaded onto the platform by a boot program. The OS manages all the applications, the applications make requests for services with compute, network and storage resources using APIs defined in the OS. Users can also interact with OS using the user interface. As the new hardware features and capabilities, including new instruction set architecture (ISAs) added to the compute platforms, OS needs to be updated to take advantage of the new features and enhancements. In addition to OSs, there are hypervisors for virtualization that run directly on compute platform without on OS. The OS changes and updates are mostly done in parallel with the hardware design and development. The changes to OS needs to be tested as part of the platform validation plan.

2.2.4: Applications and Workloads

Software applications sometimes categorized as workloads is software that’s designed to perform a set of functions for user needs and applications. These end-user applications or workloads can be optimized to specific hardware platform or OS, this is generally referred to as workload optimization. Any code additions or changes in applications or the middleware needs to be validated and optimized with the hardware platform.

3. Hardware/Software Co-Development

Traditionally hardware and software design and development was treated as two parallel development efforts. The software is run on hardware first time when the silicon is ready and any integration issues found would require design updates in the hardware. In the past, this process worked fairly well since the software was simple to change to work around hardware issues or defects, as the software is becoming more complex and platform delivery schedules were becoming more aggressive, this model of disconnected parallel hardware and software development was becoming more challenging.

Hardware and software collaborative co-development has become a new reality as hardware and software integration was becoming the crux of building high quality end to end solutions for customer needs. For hardware and software development teams, the collaborating should start from requirements to the overall development lifecycle including design, development and validation. With this quality management in this integrated development environment has become more important than ever. Integrated hardware and software testing, collaborative quality management platforms that enable co-debugging are playing a vital role. Centralized quality management tooling and metrics would help teams work on issues with end to end mindset with cross-discipline collaboration.

4. Pre-Silicon Enabling Vehicles

Pre-silicon enabling vehicles encompasses all the hardware simulation, emulation and prototyping tools used by hardware design teams and software development teams to develop and test software before the actual hardware or silicon is available. There are variety of pre-silicon enabling vehicles in the industry today that are developed using software that serve this purpose. For example, some emulators provide rich debugging environment and virtual platforms that provide a design or register accurate models that mimic hardware.

4.1 Register Accurate Simulators

Any model or tool that creates virtual software environment that simulates hardware system exactly from registers to specs is considered as register accurate models. These simulators are architecturally accurate and simulates the system so accurately that the full software stacks can run on them with no changes. Most virtual platforms are register accurate models that mimic the hardware. Virtual platforms are designed for fast execution of code in a virtual environment, they can reach up to few hundred millions of simulated instructions per second. This allows the users to complete hours of simulated time to be run in minutes or seconds of actual time. This high-speed execution in simulation is only possible by abstracting and approximating the timing of the real hardware. This also allows register accurate models like virtual platforms to be run on user's desktops and workstations. Virtual platforms are ideal for validation of software functionality and feature set as they are not representative of cycle times of real hardware.

4.1.1: Virtual Platforms

Virtual Platforms are design accurate functional models of the hardware mainly targeted for software development, integration and validation. They are also known as virtual prototyping tools or full system simulators. Several leading Electronic Design Automation (EDA) vendors offer virtual platform tools and solutions for the semiconductor industry today. Some of the examples are Vista Virtual Prototyping by Mentor Graphics, Certitude, Platform Architect MCO and Virtualizer by Synopsys, Virtual System Platform, and Palladium Hybrid by Cadence and Simics full system simulator by Windriver. All of these tools at a minimum provide the benefits such as early availability of hardware model in pre-silicon for software development and integration, provide debug capabilities, visibility into software execution, simulation capabilities for OS enabling for application/workload optimization and validation.

Vista virtual prototype by Mentor Graphics is a stand-alone simulation engine that can run on either Windows or Linux desktops. It's based on a transaction-level modeling (TLM) platform. With the integrated IDE, software engineers can gain debug visibility into the hardware and power analysis. Vista can run in two modes, functional and performance. Functional mode is ideal for software integration, validation and debugging and performance mode can be used for analyzing and optimizing the software stack for power and performance tuning. SystemC is also widely used in simulation models, SystemC is system-level modeling language that contain libraries and frameworks for hardware simulation that can be used by hardware modelling teams.

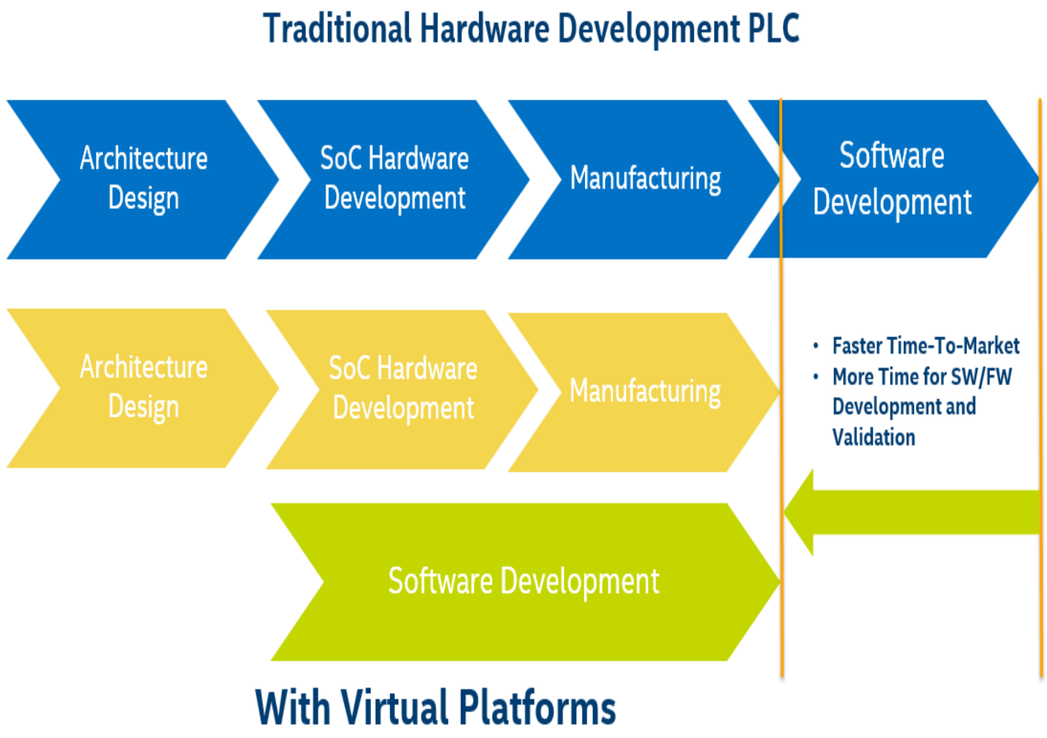
Simics is a full system simulator that can simulate any target hardware, with Simics software teams can experiment with different hardware setups, test the software behavior with different configurations of memory size, core counts, processor speeds, etc. Simics is based on the fast instruction-set simulators (ISS) which can execute several compute architectures like x86, ARM, DSP, MIPS, Power Architecture and SPARC. "Simics is designed for fast execution of code on the virtual system, typically reaching several hundred millions of simulated instructions per second, allowing full workloads to be executed." (Jakob Engblom and Dan Ekblom 2006). To run operating system on the model Simics needs other simulation models like memory-management units (MMU) to be included in the simulator. Simics can execute the system in forward and reverse direction, this capability of reverse debugging, in addition to start stop execution, checkpointing can help software teams to root cause bugs and issues faster.

4.1.2 Key Capabilities of Virtual Platforms

- Flexible and configurable, no restrictions of physical hardware for developers/architects
- Non-intrusively debug and inspect complex systems, can execute a system in forward and reverse direction for easy debugging
- Provides an abstract, executable representation of the hardware to SW/FW developers
- Runs unmodified target software from the physical target system (BIOS, FW, OS, BSP, middleware, applications)

4.1.3. Key Advantages of Virtual Platforms for Software

- Early software development on pre-RTL models, before silicon, RTL simulation, FPGA prototype is available
- Enables SW testing, debugging, find and fix SW/FW bugs in Pre-Silicon
- Improves collaboration between hardware and software design teams
- Platform SW/FW Completeness and Validation before PO (Power On)
- Path Clearing for SLE (System Level Emulation) Models
- Enables external hardware OEMs and ODMs to start SW/FW development early
- Replicable on compute platforms and shared across distributed teams
- Applications and Workloads qualification in pre-silicon



4.2 Cycle Accurate Simulators

Cycle-accurate models or simulators simulates the microarchitecture of the system based on cycle-by-cycle execution. Cycle-accurate models are dependent on specific implementation of architecture and can mimic the time precisions. For new microprocessors and microarchitecture designs, cycle-accurate simulators can help with testing and accurately benchmarking software execution in a virtual environment. All operations executed in a cycle-accurate model need to take into account architectural aspects of microprocessors like cache misses, data fetches, pipeline stalls, thread switching, branch prediction etc. A purely software based cycle accurate simulator is cost prohibitive to build. “a software-based cycle-accurate simulator would require tens to hundreds of thousands of processors which is currently infeasible and would be very costly.”

(Derek Chiou et al. 2006). This is the reason cycle accurate simulators use specialized hardware configurations using FPGAs.

4.2.1: System Level Emulation (SLE):

Hardware emulation mimics the behavior of the target hardware with other hardware like a special purpose-built emulation system. The emulation models are based on hardware description language (HDL), the primary purpose of emulation system is functional verification and debugging of the system that's being designed. Functional errors and bugs at RTL (Register transfer layer) stage of the design process can be identified and fixed using the emulation models, this will reduce the number of silicon respins and steppings saving big costs for semiconductor designers and manufactures. Functional verification using emulation also reduces overall product cost and time to market.

Major EDA vendors like Cadence, Mentor Graphics and Synopsis offer industry leading emulation platforms like Palladium Z1, Veloce Emulation Platform and ZeBu Fast Emulation. Emulation platforms take advantage of FPGAs (Field programmable gate arrays) and FPGA-based emulation software to design emulation systems. Emulation models help design and verification teams verify and debug functional and logic flows, make changes to designs as needed with a realistic cycle and architecturally accurate hardware model.

4.3 FPGA-Based Prototyping

FPGA-Based Prototyping is separate class of hardware models based on fast FPGAs for hardware verification and early software development and validation. Unlike virtual platforms which are purely a software based solution, FPGA-Based prototyping cannot be scaled to hundreds of software developers in the organization due to cost constraints. These prototyping models are used for SoC and ASIC design and verification. There is also hybrid prototyping approach that connects virtual platform with FPGA-based prototypes for software validation and hardware/software integration in early design cycles. Unlike virtual platforms, FPGA prototyping systems have almost zero debug capabilities, but are used to make stable design copies of target hardware for software development teams to use.

5. Enabling SW Stack in Pre-Silicon

5.1 BIOS/UEFI Development

BIOS or Basic Input/Output System is firmware typically stored in non-volatile memory is used for hardware initialization for booting process, BIOS also provides runtime services for operating systems and applications. Unified Extensible Firmware Interface (UEFI) has replaced BIOS, UEFI defines interface between OS and other platform firmware. BIOS is designed and developed for specific computing platform and development activities can start in pre-silicon using virtual platforms. Starting the BIOS development in pre-silicon gives the team additional time to implement features, improve quality and stability of the code. UEFI code can run on a virtual platform and can achieve basic and secure OS boot to multiple operating systems in pre-silicon. Starting this development and validation activity sooner in the overall platform development lifecycle will help the team find and fix integration issues and bugs in pre-silicon. The rich debug environment provided by virtual platforms offer the right tools for the BIOS developers to root cause bugs in software and can help the teams execute end-to-end functional and system level testing on virtual platform in pre-silicon. Most of the issues and bugs are already fixed improving the quality of the final BIOS release. Once the real silicon is available, the BIOS code is already tested and ready to be run on the silicon. The overall quality of the BIOS code is already improved at the silicon power on (PO), integration and PO time is reduced from months to days.

5.2 SW and FW Stack Development

Once the BIOS is enabled on the virtual platform, other software and firmware ingredients on the platforms can execute their set of use cases in the form of end-to-end functional and system level test cases. A typical server or client compute platform will have a 10 plus software or firmware ingredients that go with the platform. All these ingredient SW/FW releases can be validated on virtual platforms. In addition to SW/FW ingredients, there are multiple device drivers and other tools for systems diagnosis and management that can be validated in pre-silicon on virtual platforms. For power-on, the individual SW/FW ingredients will be already matured and well tested on virtual platform, improving the overall quality of the platform.

5.3 OS Enabling

Every hardware platform in the market follows a specific instruction set architecture for the processer like x86, ARM, MIPS, Power Architecture, and RISC-V to name a few. Instruction set architecture (ISA) abstracts the hardware for software programs to run on them. The ISA is the key interface between hardware and software. All software programs are written for a specific ISA and that software can run on diverse implementations of the same ISA. This allows binary compatibility among multiple generations of hardware. There are several different classifications of ISAs today, one of them is by architectural complexity, complex instruction set computer (CISC) and reduced instruction set computer (RISC) architectures are the two major categories. CISC has many specialized instructions while RISC simplifies the CPU by implementing only a subset of instructions commonly used by software applications. In RISC, the rarely used ISA operations are implemented as subroutines.

As hardware manufactures are adding new instruction sets to their CPU's or processors to meet the compute demand and improve efficiency of existing, new and emerging workloads and applications, operating systems need to take advantage of these new instructions sets and enable them in the OS. Traditionally lot of this OS feature set enabling happens in post-silicon that is after the hardware silicon samples are available, with the virtual platform all the OS enabling can move early into pre-silicon phase, giving OS enabling teams' additional time to develop and test new code. In addition to new instruction sets, any new devices that goes into an in development hardware platform like new memory technology, accelerators, I/O technology, hardware based security features, hardware based diagnostics and telemetry technologies need to be enabled in operating systems for software and applications to take advantage of these new hardware enabled enhancements. Major operating system families like Microsoft Windows, Apple iOS, Android, Linux, Chrome OS and Hypervisors, both native (bare metal) and hosted need to be enabled and optimized for new instruction sets and hardware features. OS enabling in pre-silicon on a hardware simulator like virtual platform moves the development and validation activities early in the development lifecycle giving software developers and testers ability to complete the quality processes and quality exit criteria before the silicon is ready. This means faster time to market with higher quality of the platforms ready for consumers.

5.4 Workloads Optimization

Going up the software stack, the opportunity to optimize the entire software stack for a specific middle ware framework, workload or application can be explored starting from pre-silicon, virtual platform tools offer capabilities to start the optimization and performance tuning in pre-silicon.

5.5 Workload Characterization

With the growing demand of new and emerging workloads like AI (Machine Learning/Deep Learning), Big Data Analytics, Blockchain, media streaming, web search, natural language processing and image recognition/computer vision, IoT etc. on compute platforms, workload characterization becomes a critical part of building and configuring scalable compute platforms and solutions. Measuring response times, CPU utilization, disk I/Os, memory access, network and storage latency in pre-silicon gives software developers key insights on software stack performance and power management tuning for target workloads and applications.

6. Software Development and Testing on Virtual Platforms

Leading industry standard software development and release practices like agile development, scaled agile frameworks, continuous integration and DevOps can be used for hardware software stack development and validation on virtual platforms in pre-silicon.

6.1 Continuous Integration: Automated regression testing of functional and system level test cases within a CI framework has now become an industry standard in software development. Expanding these capabilities and tools to hardware simulation platforms or virtual platforms can bring agility and efficiency to hardware software stack ingredient development teams, with seamless visibility and tracking of all integration issues and bugs in one CI framework can improve the collaboration and bug fix turnaround times. The CI environment enables faster feedback loops on bugs and fixes for all software ingredients within one framework. "Simulation can also bring other benefits, such as better feedback loops to developers for issues discovered in testing, and expansion of testing to handle faults and difficult-to-setup configurations" (Jakob Engblom. 2015).

6.2 Software Quality Qualification and Software Stack Readiness in Pre-Silicon

With Virtual Platforms and other pre-silicon software enabling vehicles, the entire platform software stack from BIOS/firmware to workloads/ applications can be run, integrated and validated to meet the expected software quality goals. This level of software integration and validation was only possible on real silicon few years ago, but with virtual platforms and other virtual prototyping tools, and with the convergence of agile development frameworks, this can be achieved in pre-silicon. Software development and integration teams can target executing 100% of system and functional test cases in pre-silicon, setting the goals of 90 to 100% test execution and test pass rates. With this level of software quality and readiness at pre-silicon exit or power on, compute platforms and solutions can be ready for market in matter of months, significantly reducing the number of post-production software bugs and issues. Application and workloads can run to their fullest efficiency and potential, taking advantage of latest hardware innovations as soon as the hardware platform is available to the consumer.

References

Derek Chiou, Huzefa Sunjeliwala, Dam Sunwoo, John Xu and Nikhil Patil, 2006 FPGA-based Fast, Cycle-Accurate, Full-System Simulators. https://pdfs.semanticscholar.org/3909/2c04278c375c1343e4ed0d2a24c616e4eb43.pdf

Jakob Engblom and Dan Ekblom 2006 SIMICS: A COMMERCIALLY PROVEN FULL-SYSTEM SIMULATION FRAMEWORK http://www.engbloms.se/publications/engblom-sesp2006.pdf

Jakob Engblom. 2015. Continuous Integration for Embedded Systems using Simulation, Embedded World 2015