

KEYS TO SIMULATION ACCELERATION AND EMULATION SUCCESS

JASON ANDREWS, CADENCE DESIGN SYSTEMS

```
vpi='h05  
vci='h1020  
hec='h39  
src_port='h02  
seq_number='h04  
description="cellReceived"  
error_count='d0
```



INTRODUCTION

For better or for worse, the engineering community, the press, and the EDA vendors themselves have classified the world of simulation acceleration and emulation incorrectly into two camps: field-programmable gate-arrays (FPGAs), and application-specific integrated circuits (ASICs). Advocates in each camp declare the same tired facts. FPGAs take forever to compile and have internal-timing problems. ASICs are power hungry and require longer development time.

When it comes to choosing an emulation system, the underlying technology does contribute to the characteristics of the system, but far too much time is spent on low-level technology details and not enough time on how emulation gets the verification job done by providing high performance and high productivity.

What engineers intend to say when they discuss "FPGA vs. ASIC" is "prototyping vs. simulation acceleration and emulation." To add to the confusion, some semiconductor companies even call their internally developed FPGA prototype an "emulator."

This paper discusses the factors that are important when evaluating simulation acceleration and emulation and the different use-modes and applications for acceleration and emulation. With the acquisition of Verisity, Cadence now offers two of the most successful acceleration/emulation product lines in the market. From this unique position,

Cadence can best serve customers by evaluating real verification needs and recommending products and technologies that enable improved verification through the use of simulation acceleration and emulation.

DEFINITIONS AND CHARACTERISTICS OF SIMULATION ACCELERATION AND EMULATION

Before the important aspects and use-modes of emulation are presented, some definitions are needed. Four distinct methods are used commonly for the execution of hardware design and verification:

- Logic simulation
- Simulation acceleration
- Emulation
- Prototyping

Each hardware execution method has associated with it specific debugging techniques; each has its own set of benefits and limitations. The execution time for these methods range from the slowest (with the most thorough debugging) to the fastest (with less debugging). For the purpose of this paper the following definitions are used:

Software simulation refers to an event-driven logic simulator that operates by propagating input changes through a design until a steady-state condition is reached. Software simulators run on workstations and use languages such as Verilog®,

VHDL, SystemC, SystemVerilog, and e to describe the design and verification environment. All hardware and verification engineers use logic simulation to verify designs.

Simulation acceleration refers to the process of mapping the synthesizable portion of the design into a hardware platform specifically designed to increase performance by evaluating the HDL constructs in parallel. The remaining portions of the simulation are not mapped into hardware, but run in a software simulator. The software simulator works in conjunction with the hardware platform to exchange simulation data. Removing most of the simulation events from the software simulator and evaluating them in parallel using the hardware increases performance. The final performance is determined by the percentage of the simulation that is left running in software, the number of I/O signals communicating between the workstation and the hardware engine, and the communication channel latency and bandwidth. A simple representation of simulation acceleration is shown in Figure 1.

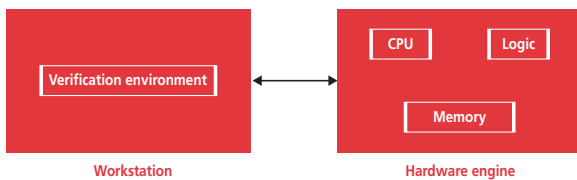


Figure 1: Simulation acceleration

During simulation acceleration, the workstation executes most of the behavioral code and the hardware engine executes the synthesizable code. Within simulation acceleration there are two modes of operation, signal-based acceleration and transaction-based acceleration. *Signal-based acceleration* (SBA) exchanges individual signal values back and forth between the workstation and hardware platform. Signal synchronization is required on every clock cycle. SBA is required for verification environments that utilize behavioral-verification models to drive and sample the design interfaces. *Transaction-based acceleration* (TBA) exchanges only high-level transaction data between the workstation and the hardware platform at less-frequent intervals. TBA splits the verification environment into two parts: the low-level state machines that control the design interfaces on every clock, and the high-level generation and checking that occurs less frequently. TBA implements the low-level functionality in hardware and the high-level functionality on the workstation. TBA increases performance by requiring less-frequent synchronization and offers the option to buffer transactions to further increase performance.

Emulation refers to the process of mapping an entire design into a hardware platform designed to further increase performance. There is no constant connection to the workstation during execution, and the hardware platform receives no input from the workstation. By eliminating the connection to the workstation, the hardware platform runs at its full speed and does not need to wait for any communication. A basic emulation example is shown in Figure 2. By definition, all aspects of the verification environment required to verify the design are placed into the hardware. Historically, this mode has restricted coding styles to the synthesizable subset of Verilog® and VHDL code, so this mode is also called *embedded testbench* or *synthesizable testbench* (STB). Even without a constant connection to the workstation, some hardware platforms allow on-demand workstation access for activities such as loading new memory data from a file, printing messages, or writing assertion-failure data to the workstation screen to indicate progress or problems.

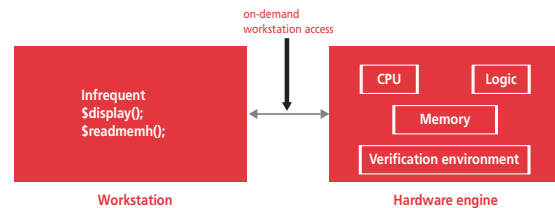


Figure 2: Emulation

In-circuit emulation (ICE) refers to the use of external hardware coupled to a hardware platform for the purpose of providing a more realistic environment for the design being verified. This hardware commonly takes the form of circuit boards, sometimes called *target boards* or a *target system*, and test equipment cabled into the hardware platform. Emulation without the use of any target system is defined as *targetless emulation*. A representation of ICE is shown in Figure 3. ICE can be performed in two modes of operation. The mode in which the emulator provides the clocks to the target system is referred to as a *static target*. When running ICE with static targets, it is possible to stop and start the emulation system, usually for the purpose of debugging. The mode in which the target system provides the clocks

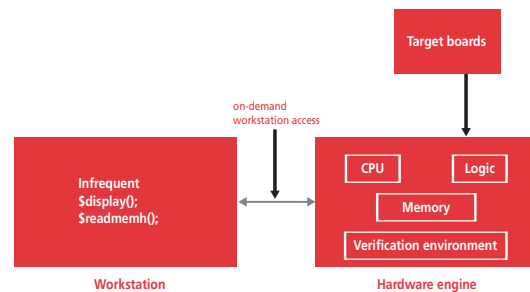


Figure 3: In-circuit emulation

to the emulator is referred to as a *dynamic target*. When using dynamic targets, there is no way to stop the emulator, because it must keep up with the clocks supplied by the target system. Dynamic targets require special considerations for proper operation and debugging.

Hardware prototype refers to the construction of custom hardware or the use of reusable hardware (breadboard) to construct a hardware representation of the system. A prototype is a representation of the final system that can be constructed faster and made available sooner than the actual product. This speed is achieved by making tradeoffs in product requirements such as performance and packaging. A common path to a prototype is to save time by substituting programmable logic for ASICs. Since prototypes are usually built using FPGAs, they are often confused with and compared to emulation systems that also use FPGA technology. As we will see, there are very few similarities between how FPGAs are used in prototypes and in emulation. Prototypes use a conventional FPGA-synthesis flow to map a design onto the target FPGA technology. Partitioning and timing issues are left to the prototype engineer to solve and require careful planning from ASIC designers at the beginning of the design process.

Now that the basic definitions of acceleration and emulation are clear, it's important to note that while some products in the market perform only simulation acceleration and some perform only emulation, more and more, the trend is for products to do both. The systems that perform both simulation acceleration and emulation usually are better at one or the other mode of operation, regardless of what the marketing brochure says. In the rest of the paper, the general term "emulator" is sometimes used for systems that do both simulation acceleration and emulation, but that "specialize" in emulation.

IMPORTANT CHARACTERISTICS OF SIMULATION ACCELERATION/EMULATION

For engineers evaluating simulation acceleration and emulation, there are many important factors to consider. The main motivation for using either simulation acceleration and/or emulation is always the significant performance increases that are possible over that of a logic simulator alone; but performance is not the only criterion to consider. This section discusses a baseline set of features that are must-haves for all users. Without these features, a product is probably not useful.

Automated compile process: Emulation offers a completely automated compile flow. The process is similar to compiling a design for logic simulation, but involves more steps to map the design onto the hardware architecture and prepare the database to be downloaded into the hardware. The user does not

need to learn about the details of the hardware, nor how to partition and route the design across boards and chips. There is no need to learn about timing issues inside the system.

Simulation-like debug: Emulation provides many advanced debugging techniques to find and fix problems as quickly as possible. When needed, 100% visibility for all time is available, as well as many other modes of debugging, including both interactive during execution and using post-processing methods when execution is complete.

Multi-user system: Project teams surely will want to verify entire chips or multi-chip systems using emulation. These tasks require large capacity. At other times, emulation is useful for smaller subsystems of the design, and capacity can be shared with other projects. To get the most value from emulation, high capacity should be available when needed, but also should be able to be split into smaller pieces and shared among engineers.

Support for complex clocking: Another benefit of emulation is its ability to handle complex clocking in a design. Designs today use every trick possible to manage performance and power requirements. Most of these techniques do not map well onto FPGA architectures, which support primarily a single-clock-edge, synchronous design style.

Easy memory mapping: Emulation provides the ability to map memories described in Verilog and VHDL automatically to the hardware with little or no user intervention.

Intellectual property (IP) integration: Many designs contain hard- and/or soft-IP cores. Emulation provides the ability to integrate processor models so emulation can be used to execute embedded software.

KEYS TO EMULATION SUCCESS

Emulation products are no different than any other product: no one product can be the best at everything. In every design process there are constraints such as cost, power, performance, and system size. Engineers make tradeoffs to determine which factors are important, what to optimize, and what to compromise. Understanding the keys to emulation success in each mode of operation will lead to the best solution for the situation.

SIGNAL-BASED ACCELERATION (SBA)

The key to success for signal-based acceleration is easy migration from logic simulation. SBA users want to turn on performance with minimal effort. System characteristics that make an easy migration possible include:

- Easy migration path from simulation. This includes partitioning of the code that goes to the hardware vs. the code that goes to the workstation and an automatic connection between these two partitions
- The closer the evaluation algorithms are to the logic simulator's algorithms, the better
- The ability to swap dynamically the execution image from the logic simulator into hardware and back again at any time
- Initialization flexibility, such as the ability to use Verilog initial statements, procedural language interface (PLI) code, and execute initialization sequences in logic simulation before starting the hardware
- Dynamic-compare for debugging when results don't match logic-simulation results

TRANSACTION-BASED ACCELERATION (TBA)

The key to success for transaction-based acceleration is the verification environment and the verification IP (VIP). TBA offers higher performance compared to SBA, but requires some additional planning up front. System characteristics that make TBA successful include:

- Easy-to-use TBA infrastructure for VIP creation
- High-bandwidth, low-latency channel between the emulator and workstation that enables the emulator to run as the master and to interrupt the workstation dynamically when needed
- Library of TBA-ready VIP for popular protocols that runs congruently in the logic simulator and on the emulator

EMBEDDED/SYNTHESIZABLE TESTBENCH (STB)

Synthesizable testbench is all about performance. In addition to runtime speed, performance also comprises the overall turnaround time, including compile time, waveform generation, and debugging. For those designs that don't have multiple-complex interfaces connected to the design, STB provides the highest level of performance possible. In addition, STB is used commonly for software development where performance is critical. The ability to interrupt the workstation infrequently in order to execute some behavioral functions, such as loading memory or printing messages, makes STB easier to use.

IN-CIRCUIT EMULATION (ICE)

The key to ICE is the availability of hardware interfaces and target boards. ICE is always tricky because the emulator is running much more slowly than the real system will run.

- Collection of ICE-ready vertical solutions for popular protocols to minimize custom hardware development.

- Support for all types of ICE targets
 - Static targets that enable the provided clock to be slow or even stopped
 - Dynamic targets where the target board needs a constant or minimum clock frequency to operate correctly and stopping the clock would be fatal
 - Targets that drive a clock into the emulator
- ICE-friendly debugging
 - Connections to software debugging environments
 - Large trace depths for debugging dynamic targets

CHARACTERISTICS OF PALLADIUM AND XTREME

The keys to emulation discussed above help determine the most important modes of operation and the solutions that best fit a set of verification projects. The next section describes the characteristics of the Cadence® Palladium® and Xtreme products, along with the strengths of each.

PALLADIUM FAMILY

Cadence Palladium systems are built using ASICs designed specifically for emulation. Each ASIC in a Palladium system implements hundreds of programmable Boolean processors for logic evaluation. Each ASIC is combined with memory on a ceramic multi-chip module (MCM) for modeling design memory and for debugging. The Palladium II custom processor is fabricated using IBM CU08, eight-layer copper-process technology. Each board in the system contains a number of interconnected MCMs to increase logic and memory capacity.

Palladium uses the custom-processor array to implement a statically scheduled, cycle-based evaluation algorithm. Each processor is capable of implementing any four-input logic function using any results of previous calculations. During compilation, the design is flattened and broken into four-input logic functions. Each calculation is assigned to a specific processor during a specific time slot of the emulation cycle. The emulation cycle comprises the number of time slots, or steps, required for all the processors to evaluate the logic of the design. Typically, the number of steps required for a large design is somewhere between 125 and 320 steps. Performance can be improved through compiler enhancements that result in a shorter emulation cycle. The other benefit of the evaluation algorithm used by Palladium is its ability to achieve higher performance by adding more processors. The availability of processors enables greater parallelization of the computations, and has a direct impact on performance. Because the scheduling of evaluations is fixed, performance doesn't depend on design activity or design gate-count.

The technology used in Palladium gives it many desirable characteristics for emulation:

Highest performance: Since custom processors are designed for high-performance emulation right from the start, using the most advanced semiconductor processes, the chips are clocked at hundreds of MHz, resulting in emulation speeds exceeding 1 MHz without any special advance design planning.

Fastest compile: The processor-based architecture leads to compile speeds of 10–30 million gates-per-hour on a single workstation.

Best debugging: The custom hardware of Palladium is also designed for fast debugging. It supports multiple modes of debugging depending on how much information the user would like to see. It also has dedicated memory and high-bandwidth channels to access debugging data.

Large memory: The custom hardware and the advanced technology enable very large on-chip memory with fast access time. This capability enables users to incorporate large on-chip and on-board memories with full visibility into the memory while maintaining high performance. Designers can use this memory for test vectors or storing embedded software code.

Highest capacity: The ability to connect multiple units without changing the run-time frequency and bring-up time of the overall environment makes the Palladium family scalable, and provides an environment that supports 2–128 million gates with Palladium and 5–256 million gates with Palladium II. This is the highest emulation capacity in the market.

Palladium offers a typical speed in the range of 600 kHz–1 MHz. (with many designs running over 1 MHz.). It is an ideal system for ICE and STB. A deterministic algorithm and precise control over I/O timing enable it to excel in ICE applications where constant speed is critical. Boundary scan (i.e., JTAG) is an example of an application that requires constant clock rates, the ability for the emulator to receive a clock signal from an external target board, and the need for hardware debugging when the target system cannot be stopped. Decades of ICE experience translate into the most available and reliable off-the-shelf ICE solutions for major markets including networking, multimedia, and wireless. Palladium can do simulation acceleration (including SBA and TBA), but lacks some of the advanced features such as bi-directional simulation swapping and dynamic-compare with the logic simulator. Palladium also supports assertions in all modes of operation, including ICE with dynamic target.

Palladium supports up to 32 users, 61,440 in-circuit I/O signals, and is a true enterprise-class verification system with the highest capacity and highest performance. It can be used by multiple engineers and multiple projects in a company from anywhere in the world.

XTREME FAMILY

Cadence Xtreme systems are built using reconfigurable computing co-processors (RCCs) implemented in programmable logic using high-density commercial FPGAs. Each FPGA implements hundreds of reconfigurable processors of different types based on the logic in the design. In addition to logic evaluation, each FPGA also contains internal memory that can be used to model design memory. Each board contains a number of FPGAs combined with additional static and dynamic memory to increase logic and memory capacity.

Although Xtreme uses commercial FPGA devices, it does not operate like other FPGA-based systems. When considering how to use FPGAs, most engineers immediately think of synthesizing their design into a netlist and mapping gate-for-gate, wire-for-wire onto the array of FPGAs and trying to manage all the timing issues associated with internal, FPGA timing and routing between FPGAs. This description in no way resembles how RCC works. The best way to think of RCC technology is to think about how a logic simulator works. A simulator uses an event-based algorithm that keeps track of signal changes as events. It schedules new events by putting them into an event queue. At any particular simulation time, the algorithm updates new values (based on previous events) and schedule new events for some future time. As simulation time advances, only the values that are changing are updated and no extra work is done to compute values that don't change. This event-based algorithm is the concept behind RCC. It uses patented, dynamic-event signaling to implement a simulation-like algorithm. The difference is that the events are all executing in hardware. Messages are sent between the co-processors as events occur and updates are required in other co-processors. As with a simulator, new values are only computed when necessary, based on design activity. Dynamic-event signaling leads to a system that has none of the timing issues associated with FPGAs, and a product that runs and feels like a logic simulator.

The technology used in Xtreme gives it many desirable characteristics for simulation acceleration and emulation:

Low power: FPGAs consume very little power and the dynamic-event signaling algorithm used in Xtreme translates into good emulation performance with lower clock rates inside the system.

Small form factor: Low power combined with high-density FPGAs translates into a 50-million-gate system (with 60–65% utilization) that fits in a desktop form factor, or a 6U rack mounted chassis. Xtreme is a portable system that is light, easy to transport, and is often placed in an engineer’s cubicle instead of a lab or computer room.

Frequent capacity upgrades and cost reductions: Since emulation capacity and cost follow the mainstream-FPGA technology curve, it is possible to introduce very rapidly new systems that increase capacity and/or lower cost with minimal effort.

Xtreme offers a typical speed in the range of 150–300 kHz and is an ideal system for simulation acceleration and TBA. It excels at targetless emulation and in any environment with a mix of SBA, TBA, and STB.

Xtreme offers better performance than classic simulation acceleration systems that enforce the workstation as master. Xtreme is also an ideal system for accelerating *e* verification environments using SpeXtreme, a product that utilizes the behavioral-processing features of Xtreme to increase overall verification performance. Xtreme can handle ICE, but lacks many of the advanced ICE features that are available in Palladium, such as dynamic-target support and robust ICE-debugging methods.

The Xtreme family supports up to 12 users, 4,656 in-circuit I/O signals, and is a design-team-class verification system that can be used by multiple engineers working on a project. It offers excellent price vs. performance.

Although there is some overlap in capabilities, the Xtreme and Palladium families of products excel at different modes of operation, as shown in Figure 4.

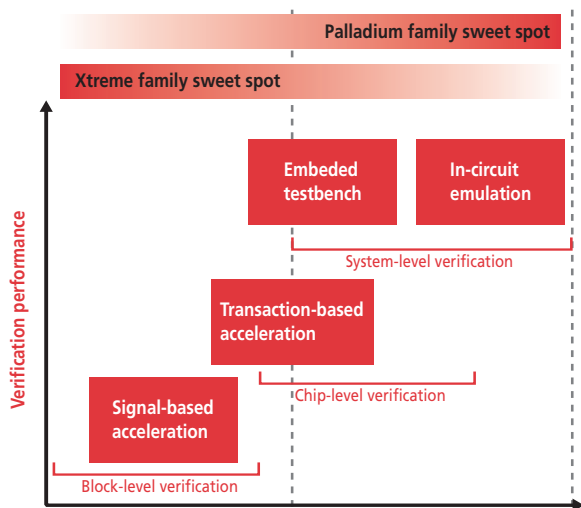


Figure 4: Emulation strengths

CONCLUSION

In this paper, we discussed that all products must make trade-offs in deciding which parameters should be optimized and which should be compromised. With the Xtreme and Palladium families, Cadence is in the ideal position to serve customers of all types. Some users choose one of the product lines to use throughout the phases of the project. Others utilize both technologies, each one for different phases of verification, or for different projects.

Palladium offers the highest performance and highest capacity for emulation power-users with very large designs. Palladium is the most comprehensive solution for ICE and was designed with advanced ICE users in mind.

Because of its simulation-like algorithm, Xtreme excels at simulation acceleration and TBA and provides higher performance than other acceleration solutions. It was designed with simulation acceleration in mind, with concepts like simulation swap and dynamic compare that make it run and feel like a logic simulator.

Although the technologies behind Palladium and Xtreme are different, both are processor-based architectures that provide automated compile, short bring-up times, and scalable multiuser capacity not found in prototyping systems. Both systems provide more design-turns-per-day than competing systems, and each has a proven track record of success in the marketplace. Together, these product lines are being used by more than two-thirds of all simulation acceleration and emulation users.

Engineers evaluating emulation need to understand the keys to success presented here for the different modes of operation and determine which solution is the best fit for their verification challenges.