GETTING MORE FROM HARDWARE-ASSISTED DESIGN ENVIRONMENTS WITH ALTAIR HERO 2.0

Stuart Taylor – Altair / July 7, 2022

When it Comes to Accelerating EDA Verification, the Industry Needs a Hero

There's perhaps no industry more competitive than electronics design and manufacturing. As Moore's law has stalled and designs grow in size and complexity, organizations are looking for new ways to speed up verification and regression testing. Increasingly, firms are turning to hardware-assisted verification technologies to simulate designs faster and speed up regression tests. However, unlike software-based verification, where workload management tools are commonly used to improve throughput and optimize resources such as hardware and tool licenses, these are few such tools for emulators. This is especially true in multi-vendor environments.

This paper will explain some of the challenges associated with hardware verification and introduce Altair® Hero™ – an enterprise job scheduler purpose-built for multi-vendor emulation environments. It will also discuss techniques used to improve emulator efficiency and show how organizations can improve throughput and realize a better ROI on their emulator investments by leveraging advanced workload management.

The Need for Hardware-Assisted Verification

Designers of modern semiconductors face a plethora of challenges. Challenges range from increasingly complex designs, intense competition, stringent quality requirements, and the need to curb energy consumption and prolong battery life. Electronic devices incorporate a growing variety of peripherals and interfaces, and software environments are becoming more complex. To meet time-to-market objectives, engineers must begin integrating hardware and software earlier in the design cycle, pre-silicon.

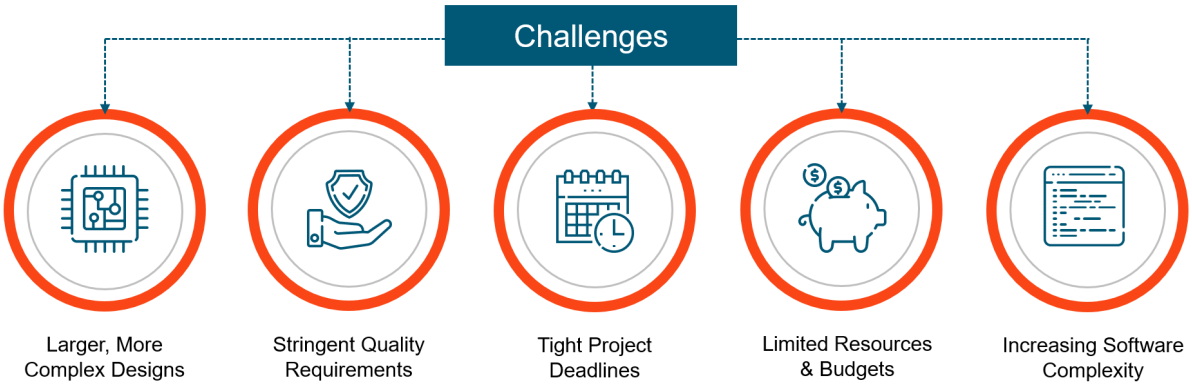

Figure 1 – Challenges in semiconductor design environments

Ensuring that designs are error-free requires extensive verification following design changes and before fabrication. By most estimates, verification accounts for approximately 70% of the simulation workload in EDA server farms, so optimizing verification is critical.[1]

As shown in Table 1, traditional software-based simulators are too slow for many requirements. When performing register-transfer level (RTL) simulations on a complex design, a single-threaded simulator can model device functionality at approximately one cycle per second — billions of times slower than the target device itself. To improve throughput, verification engineers typically exploit massive parallelism, running thousands or millions of verification jobs across large server farms comprising multi-core processors. Even with all this parallelism, a large site with the capacity to run 20,000-50,000 concurrent verification jobs can, at best, achieve effective simulation rates in the range of 20-50 kHz.

Table 1 - Verification options – software vs. hardware-assisted verification (HAV)

Technology	Example	Cost	Build Time	Cycle Time	Scheduling Complexity
CPU (software-based simulators)	Intel® Xeon®, AMD® EPYC™	$	Minutes	1 Hz	10
Custom Processor (Emulator)	Cadence® Paladium®, Mentor® Veloce®	$$$$	0.5 to 3 days	100 kHz to 1 MHz	1000
FPGA Platform (Emulator)	Synopsys® ZeBu®, Cadence® Protium™	$$$	Several days and a few weeks of preparation	1 to 10 MHz	1000
FPGA Prototyping	Synopsys HAPS®	$$	Months / co-design	10 MHz	1
GPU MIG	A100	$$	Minutes	1-2 GHz	100

Organizations increasingly turn to hardware-assisted verification technologies to achieve higher simulation rates. Today, emulators are widely used for pre-silicon software development, hardware/software verification, debugging, and in-circuit emulation (ICE). Emulation platforms such as Cadence® Palladium® Z1/Z2 and Mentor® Veloce® can deliver simulated cycle times in the range of 100 kHz to 1 MHz. They also provide built-in support for standard peripherals, including ethernet adapters, USB interfaces, and disk controllers. Hardware-accelerated platforms based on FPGAs are even faster. However, it takes more time to load new designs into these environments.

According to research by the ESD Alliance, the use of hardware-assisted verification tools is on the rise. While HAV lagged software-based simulators using hardware description languages (HDL) in the late 1990s and early 2000s, this has changed. Since 2018, investments in hardware emulation now exceed software-based verification growing to $718 million in 2020.[2] According to Global Market Insights, the HAV market is expected to exceed $15 billion by 2027, representing a CAGR of over 15%.[3]

While emulators improve throughput, they are not without challenges. Even a single emulator can cost millions of dollars – not including the cost of a dedicated team to support the specialized workflows that it requires. As a result of the significant investment required, maximizing the use of emulator resources is critical.

Getting More from Emulator Investments

Given the high cost of acquiring and maintaining emulators, they must be used efficiently. For verification engineers, it's tempting to assess emulator utilization based on whether a design compiles to use all available gates in an emulator. This is an overly simplistic way of viewing utilization, however. The dynamics around utilization are more complicated. Design teams running in-circuit emulation (ICE) jobs typically run these workloads during business hours, as illustrated in Figure 2. These jobs run partly in the emulator and

[1] Altair internal estimate.
[2] June 2021 – New-Tech Europe – Market-Driven Trends in Hardware Emulation
[3] October 2021 – Global Market Insights – Industry Trends – Hardware-Assisted Verification Market

partly on real hardware, so a human is generally present to conduct the test. For organizations that run only ICE jobs, emulators may sit idle overnight and on weekends.

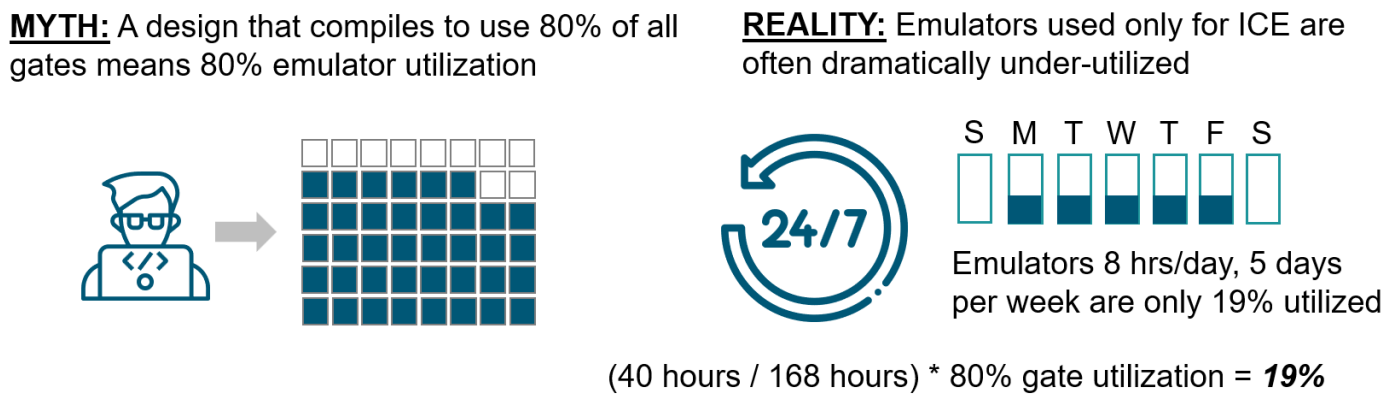

Figure 2 - Emulators are often underutilized — especially when used only for ICE/DEBUG activities

While the utilization challenge can be mitigated by running simulation acceleration (SA) jobs (both overnight batch and interactive), this is easier said than done. Designs must be compiled from RTL code, emulation boards must be allocated and programmed, and virtual target devices such as PCI, USB, and video controllers must be soft-assigned before jobs can run. The number of emulation boards assigned varies with the complexity of the design.

Organizations often use hard-partitioning strategies to allocate emulator resources among teams. However, these allocations may be incompatible with the needs of SA jobs. There is also the time required to cut over between workloads. SA jobs running overnight may fail or jam. If this occurs, the environment may not be available to resume ICE jobs the following day, impacting utilization and engineering productivity. Many scheduling challenges, such as allocating resources fairly across projects and design teams, are similar to scheduling challenges in software-based verification. Hardware emulators bring unique challenges, however.

A Challenging Scheduling Problem

Those familiar with traditional software-based verification know that optimizing compute resources and license features is critical to productivity. The same is true for emulators, but the scheduling complexity is orders of magnitude more complex.

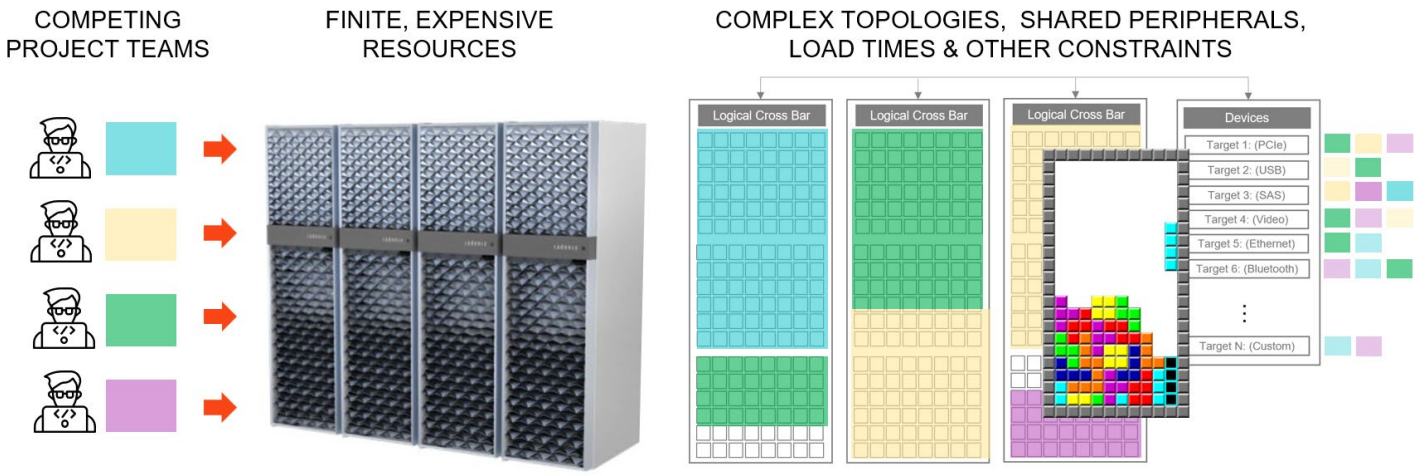

Figure 3 - Scheduling emulation jobs is much more complex than software-based verification jobs

Large system-on-a-chip (SoC) designs can have billions of gates. Accommodating these designs means not just spanning individual boards or modules, but multiple emulator racks. Each emulation job has its own size and shape.

Given the complex topology of emulators, and the need to compile and load designs in advance, scheduling workloads is difficult. The challenge of packing workloads efficiently is often referred to as the Tetris problem by emulation users. Compounding the scheduling challenge is that different emulators have different topology characteristics. Schedulers need to:

- Account for existing utilization and interactive jobs when placing new workloads
- Schedule and share a limited number of emulated peripheral devices
- Consider long lead times and workflows required to compile designs and load them into the emulator
- Accommodate design teams requesting hard usage windows and future resource reservations

Because of these challenges, the use of automated scheduling remains limited in most emulation environments. Engineers perform tasks such as compiling designs, assigning virtual devices, and deciding which physical boards to program to support each emulation task. Traditional schedulers may help with some workloads, but they tend to lack the features required by emulation environments. As a result, in many environments, emulation workloads are still managed manually. While manual approaches are acceptable when just one project team uses an emulator, it quickly becomes an issue when multiple groups compete for the same resources. Ideally, the allocation of hardware emulation resources should be automated for greater efficiency and throughput.

Focusing on Metrics that Matter

Rather than focusing on simple metrics such as the percentage of emulator gates used, improving efficiency requires that organizations focus on metrics more relevant to design activities. Examples include:

- The number of simulated cycles per week
- How quickly an emulation can begin after a new netlist becomes available
- DEBUG/ICE turnarounds per week
- The time required to get an interactive job started.

As with software-based tools, the scheduler must balance considerations such as keeping resources utilized, sharing resources fairly, and ensuring that capacity is available to support urgent requirements.

A New Breed of Scheduler

Improving the metrics above, and making better use of hardware, requires new scheduling capabilities specific to emulators. Among these capabilities are:

- Compile/simulation pipelining
- Job chaining – the ability to run multiple tests against a loaded design
- Improved robustness for batch simulation acceleration (SA) jobs
- Preemption of batch SA jobs to support interactive ICE activities
- Monitoring and reporting tailored to emulators

While Altair doesn't design chips or operate emulators, it does have deep expertise in optimizing workload and resource utilization in EDA environments. As early as 2013, Altair EDA clients have sought to leverage the schedulers used in software-based verification environments and apply them to emulators. There was a need for schedulers able to deal with the complexities and nuances of hardware emulators.

Altair began extending its advanced scheduling capabilities to hardware-assisted verification environments, building on its success in optimizing traditional CPU-based verification workloads. In 2018, Altair announced its first commercial product explicitly aimed at emulators – Altair Hero (an acronym that stands for Hardware Emulator Resource Optimizer).

Before delving into Hero and explaining how it helps optimize emulation workloads, it's helpful to describe Altair® Accelerator®. Altair Hero is based on Altair Accelerator and essentially provides a superset of Altair Accelerator's functionality.

Altair Accelerator

Altair Accelerator is high throughput, enterprise-grade job scheduler designed to meet the complex needs of the semiconductor industry. It features an event-driven, low latency design for better throughput and utilization without compromising on scheduling features. Altair Accelerator provides complete visibility to what's running, making it easy to identify job status and failing jobs and quickly drill down for root cause analysis.

Scheduling policies such as FairShare allocation, job preemption, and resource reservation built into Altair Accelerator ensures resources are used according to business priorities. For example, Accelerator allows urgent, high-priority jobs and workflows to take precedence over lower-priority tasks. Accelerator can suspend running jobs and resume them once the high-priority job has been completed. This eliminates the need for organizations to hold licenses in reserve for urgent jobs and ensures that business deadlines are met while optimizing utilization and throughput across the environment.

For customers, Altair Accelerator can deliver dramatic improvements in throughput and efficiency. One such example is the case of CEA Tech, the Grenoble-based technology research unit for the French Alternative Energies and Atomic Energy Commission (CEA). In a recent case study of detailing the use of Altair Accelerator at CEA Tech, EDA server farm administrators confirm that ***Altair Accelerator enabled an impressive 4.5x throughput gain*** for selected EDA workloads.[4]

It's worth noting that there are multiple products in the Altair Accelerator portfolio that can help clients optimize their EDA environments. The Altair Accelerator product portfolio is shown in Figure 4. The remainder of this paper will focus on Altair Hero, a high-performance scheduler for hardware emulation environments.

[4] 2022 Altair - CEA Speeds Up EDA for Research - Powering R&D at the French Alternative Energies and Atomic Energy Commission

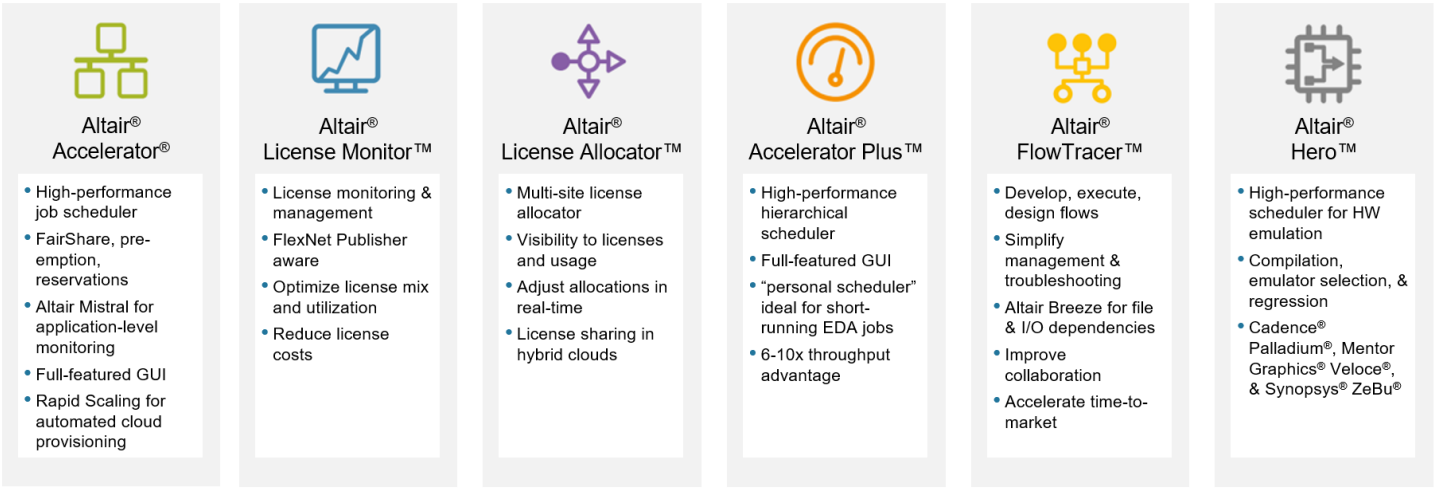

Figure 4 – The Altair Accelerator product portfolio

Altair Hero

Altair Hero is an end-to-end solution designed specifically for hardware emulation environments. It addresses all aspects of emulation flow including design compilation, emulator selection, and software and regression tests. Hero's vendor-independent architecture and comprehensive policy management features provide organizations with flexibility and control. It also provides visibility into emulator status, customizable alerts and notifications, and optimized hardware utilization.

With Hero, organizations can:

- Share multiple emulators across projects and design teams
- Dramatically improve emulator utilization with fair sharing policies
- Monitor usage with real-time visibility and control
- Mix ICE/interactive with batch SA workloads

Support for Multiple Hardware Emulators

As illustrated in Figure 5, Hero is designed to support a variety of hardware assisted verification platforms. As a superset of Altair Accelerator, it supports traditional software-based tools as well as hardware emulators based on custom processors and FPGAs. Although FPGA prototyping platforms such as Synopsys HAPS® are straightforward to schedule, it supports these as well for completeness. GPU-based acceleration is not a focus of Hero presently, but the scheduler is designed to support multi-instance GPUs (MIGs) enabling GPU-aware tools to share NVIDIA GPU resources more efficiently.

Hero 1.0, released in 2018, initially focused on SA workloads on Cadence Palladium Z1 emulators. For this narrow use case, Hero demonstrated a **95% emulator utilization rate over a weekly period**. Functionality in Hero was subsequently expanded to support ICE and SA workloads on additional emulators including Cadence Palladium XP II and Synopsys ZeBu Z4. Hero 2.0 is a complete re-write of the original Hero offering designed from the ground up to offer new capabilities and support a wider range of scheduling problems.

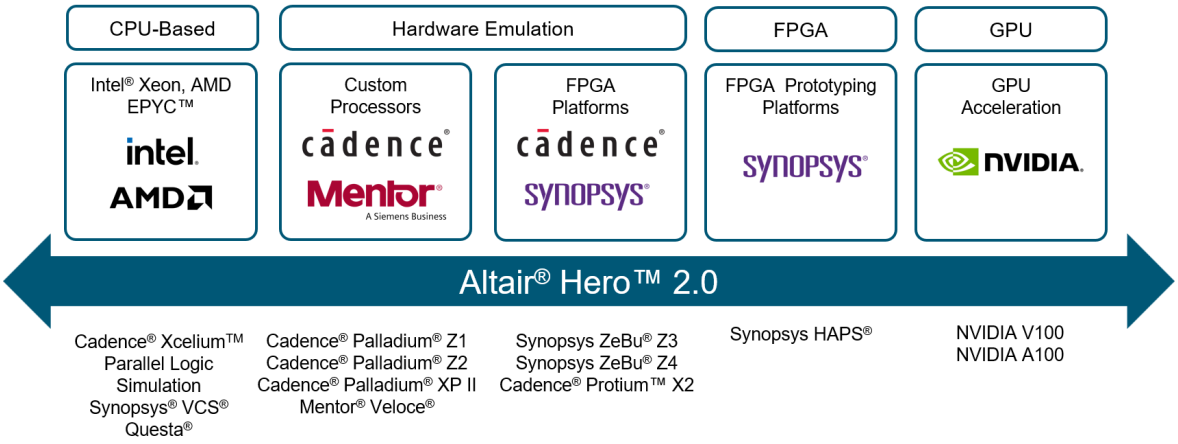

Figure 5 - Altair Hero is designed to support multiple verification and emulation technologies

The Hero architecture is designed to be emulator-agnostic. It provides a generic scheduling model that treats boards and modules as "leaf" resources, making it adaptable to most commercially available emulation platforms.

- To date, Altair has field experience successfully deploying Hero with Cadence Palladium Z1, Cadence Palladium Z2, Synopsys Zebu Z4, Synopsys HAPS, as well as previous generation Palladium XP2 emulators.
- Hero is also designed to support various versions of the following environments: Cadence Protium, Mentor Veloce. Synopsys ZeBu Z3 is expected to be very similar to Synopsys Zebu Z4 already supported. Altair is looking for field experience on these additional platforms.
- Future Hero releases will improve support for GPU-accelerated tools using complex GPU configurations including multi-instance GPU (MIG) support.

Hero is the industry's only end-to-end enterprise job scheduler specifically designed for multi-vendor hardware emulation environments. Key features of Hero 2.0 are:

- Policy management including FairShare and preemption
- Soft reservations enabling users to reserve blocks of time on an emulator in advance
- Visibility to emulator-specific metrics for hardware asset optimization and organizational planning
- A rich GUI to simplify monitoring and determine the root cause of failing jobs
- Support for emulation and prototyping platforms from multiple vendors

Using Emulators More Efficiently

Hero incorporates several key features that can help organizations use emulators more effectively. With more effective scheduling, organizations can improve utilization, share resources more effectively, and reduce the amount of time that verification engineers spend waiting for resources to become available.

Sharing Emulators Among Teams

Hero is straightforward to install and configure. Administrator can simply install the software on an available Linux host, create queues, and edit a master configuration file that describes the topology characteristics of available emulators in the environment. For each emulator, administrators provide a name, specify the type of emulator (PalladiumZ1, ZeBu, etc..), and place emulators in groups. The configuration file also defines the list of runtime hosts (RTHOSTS) that comprise the cluster associated with each emulator.

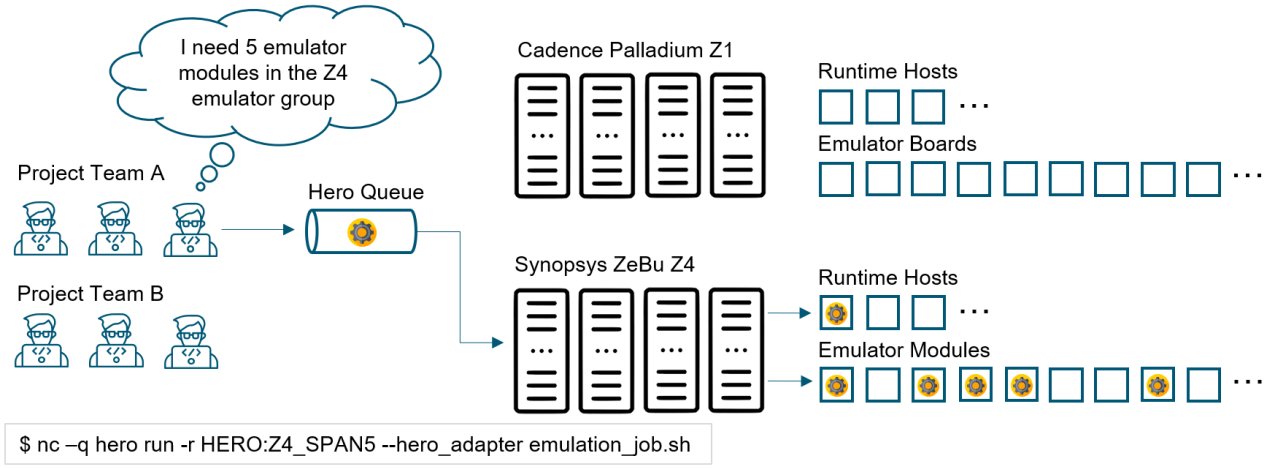

Figure 6 - Hero abstracts emulator resources making them sharable across teams

Project teams request emulator resources as illustrated in Figure 6, by submitting a job using a resource requirement. In the example above, five modules are required on a Synopsys ZeBu Z4 emulator. The “Z4” in the resource requirement string represents the name of the emulator group defined in the configuration file, and “SPAN5” expresses the number of modules or boards depending on the emulator technology. There may be multiple Z4 emulators in the environment, so users simply specify an emulator group and the number of modules required and Hero manages scheduling and allocation. Resources may be requested for either SA jobs or ICE jobs. Hero tracks the availability of runtime hosts and available emulator modules across multiple clusters and automatically determines the optimal placement for each new emulation workload. This high level of abstraction is what enables Hero to be emulator agnostic.

Hero enables emulator users to benefit from the same kinds of policy-based controls common in software verification environments, such as attaching different priorities to different emulation jobs, applying FairShare policies to manage sharing of emulator resources, and preemption ensuring the resources are available during business hours for interactive ICE activities.

Monitor Emulation Jobs in Real Time

The golden rule of process optimization is that you can’t manage what you can’t measure. Hero provides granular, realtime visibility to emulator resources as illustrated in Figure 7. This includes visibility to runtime host allocations and the boards and modules used across the various emulators. Hero allows boards that are down for maintenance to be de-configured and removed from the eligible pool of resources available for scheduling.

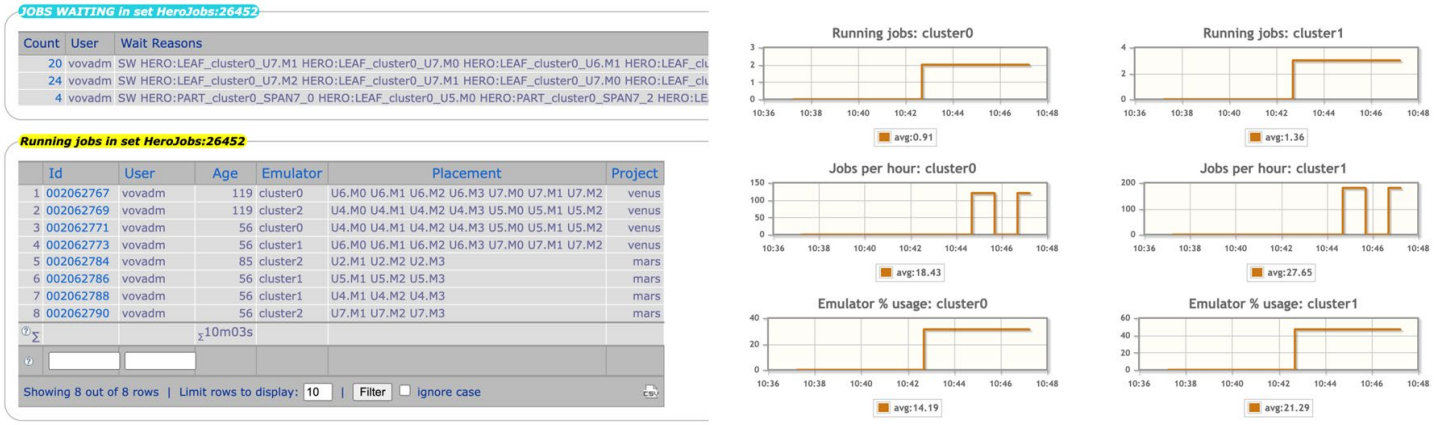

Figure 7 – Hero enables users to monitor jobs and emulator usage in real time

Users can also monitor queued and running jobs associated with various project teams and regressions runs, and see detailed information about job placement. The Hero GUI automatically presents key metrics in graphical form on a per-emulator group basis. It can also show aggregated statistics across all emulation platforms.

Visibility to job pending reasons helps users troubleshoot issues and minimize delays in getting emulation jobs compiled, loaded, and started. Hero also provides visibility to metrics such as job throughput per hour per emulator, wait times, and utilization metrics so that administrators can quickly spot issues and improve the overall efficiency of the environment.

Reservations and Preemption

When multiple project teams are sharing emulator resources, it's helpful to be able to reserve emulator resources in advance. For example, a project team may wish to reserve a specific number of emulator leaf resources (boards or modules) for a future date. The challenge with reserving a set number of resources is that there is no guarantee that the boards/modules selected will be part of the same span.

To deal with the practical challenges associated with reserving resources on emulators, Hero 2.0 provides a new *hero_reserve* facility. This facility enables users to reserve named boards or modules on specific emulators and ensure they are available at a future date. Users can also use this facility to search for a particular set of resources (e.g., a specific $group $span combination) and rely on Hero to identify times when the necessary resources will be available.

Hero 2.0 makes reserving emulator capacity similar to reserving a meeting room. Users can express the resources required, the duration, and optionally specify a time window in which the reservation is needed. Assuming a reservation window can be found that meets the resource requirements, Hero will return a reservation ID. The reservation ID can then be used to schedule a future job against the reservation. Users may have multiple advanced reservations.

Figure 8 illustrates reservations made on a particular date. A specific set of resources on a Synopsys ZeBu cluster (cluster0) are reserved for 2 hours at 6:00 pm. Another set of emulator modules on a separate cluster has back-to-back reservations between 6:00 pm and midnight.

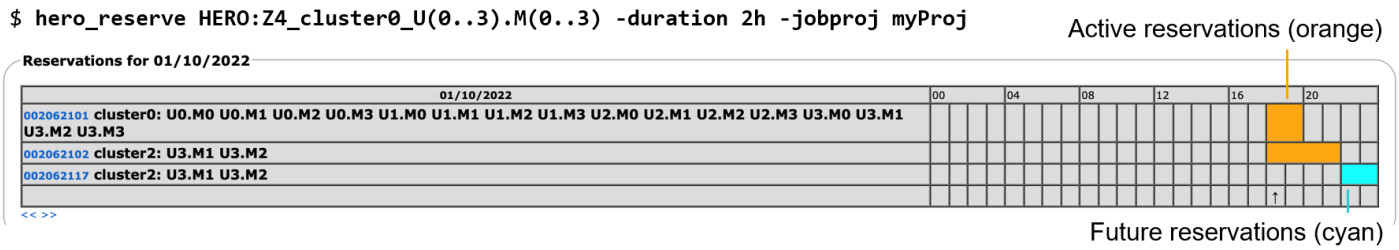

Figure 8 - Hero 2.0 supports calendar-style reservations, enabling teams to reserve resources in advance

A common use case for resource reservations is dealing with interactive ICE jobs. Suppose emulators are tied up running many SA jobs around the clock. In that case, it's helpful to reserve capacity for interactive jobs during business hours. Using reservations, engineers can ensure that the emulator's boards are pre-loaded with the correct model. They can then show up at an appointed time and launch an interactive job against the reserved resource, as illustrated in Figure 9.

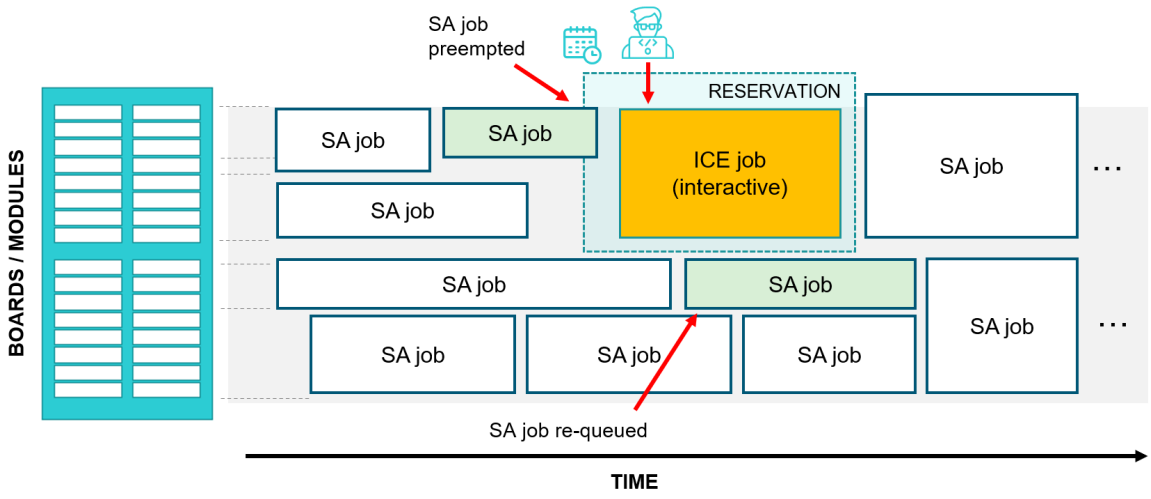

Figure 9 – Policy-based preemption of SA jobs to accommodate interactive ICE job

Hero avoids preempting jobs if possible. If nobody shows up to claim a reservation, the SA job occupying the resources will continue to run. Suppose this user shows up and starts the ICE job referencing the correct reservation ID. In that case, the job can either be re-queued on an equivalent set of resources so it can run later, or it can be terminated. This combination of reservations with preemption is helpful because it helps improve two of the metrics discussed earlier:

- By enabling resources to be reserved in advance, organizations can improve the number of DEBUG/ICE turnarounds per week, ensuring that engineers do not sit idle
- Similarly, scheduling resources helps reduce the time required for an interactive job to get started

Regression Tests and Job Chaining

While practices vary by design team, a standard usage model for software-centric tests and simulation acceleration is illustrated in Figure 10. The time required to load a design into an emulator is often much greater than the runtime for an individual test. As a result, engineers tend to group regression tests into separate “sets” where a particular team manages each set. This is an efficient way of using the emulator since it maximizes utilization and avoids needing to reload designs for each design team. Design engineers typically know how long each test and test set is expected to run based on experience.

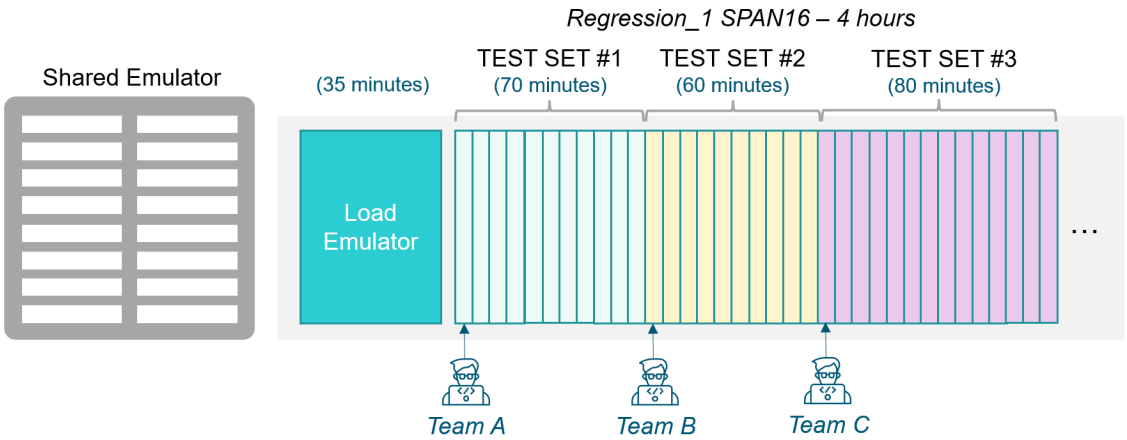

Figure 10 – Design teams often group regression tests into test sets to avoid re-loading designs

Reporting features in Hero can help improve the accuracy of these runtime estimates. Since engineers can estimate the time required for a series of regression tests, appropriate resources can be reserved, maximizing the use of emulators.

Hero 2.0 incorporates new functionality to support this practice. SA tests do not always behave well against loaded designs. Hero provides runaway mitigation mechanisms to abort tests if they exceed a particular duration automatically. This avoids resources being wasted due to tests that have hung or failed. This is a valuable feature for overnight SA runs when there may be nobody monitoring the environment. Hero supports primitives such as *autokill* to kill a task that has exceeded its expected runtime.

Similarly, administrators can specify a *maxLife* parameter that avoids accepting new tests after a time limit is expired. The total time for the regression can also be capped. These features, in combination, ensure that design teams do not monopolize precious emulator resources.

New in Hero 2.0 is “per test visibility.” Users of Altair Accelerator are familiar with the idea of using color codes to monitor the status of jobs. Hero 2.0 employs standard Altair Accelerator color codes so that engineers can quickly identify emulator jobs that are running, queued, completed, failed, or de-queued for policy reasons (e.g., a job may be preempted). With this information, engineers can quickly zero in on failed tests and incorporate needed design changes before the next set of regression tests.

Emulation Pipelining

Compiling a new design and preparing it to run on an emulator can take several days. Ideally, engineers would like to be able to submit test jobs in anticipation of a future design being available. They also need the ability to run tests against a current version of a design and cut over when a newly compiled design is loaded and becomes available.

Hero 2.0 supports the notion of pipelining compilation and simulation acceleration runs. The design loads and SA jobs can be triggered automatically after a long-running compilation workflow completes. Hero can handle a variety of distributed parallel flows used for compiling large Netlists, including Cadence IXCOM/HDL-ICE and VXE/WXE. By supporting compilation workflows and automatically loading them into the emulator, Hero helps keep emulator resources fully utilized, and design teams can work against the latest builds.

Reporting

Hero provides real-time monitoring and rich reporting features that provide better visibility into regressions. This helps engineers and site administrators understand how emulator resources are used. By monitoring and reporting on user, project, and team usage, administrators can devise strategies to use resources more efficiently. Hero can track emulator board/module utilization as well as job counts. Key parameters about each job can be logged to CSV files or a PostgreSQL database to facilitate historical reporting. Customers can create custom dashboards and visualizations using their preferred BI tools. Reporting in Hero can be used with a customer’s in-house reporting databases.

Additionally, Hero provides an interface to Altair® Monitor®, typically used to monitor license checkouts. When used with Hero, Altair Monitor provides additional visualizations such as heatmaps and efficiency charts, as illustrated in Figure 11.

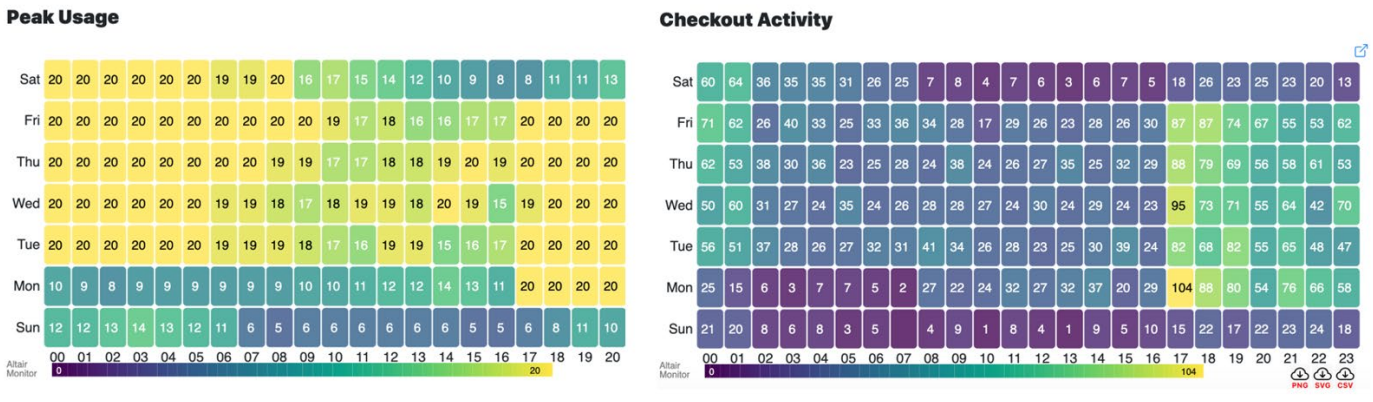

Figure 11 Altair Hero can be used with Altair Monitor to provide additional visualizations

A common use case is to run the same job repeatedly in successive regressions as designs change or conditions are modified. Hero provides a “longitudinal” reporting capability using its built-in jobs database, enabling verification teams to filter on a particular group of tests or individual tests and and compare the results and job metrics across multiple versions of the design image. This is yet another way that Hero can help organizations use emulation resources more effectively.

Compelling Benefits

Altair Hero is an enterprise job scheduler designed for hardware emulation environments. It is vendor-agnostic and capable of managing job scheduling requirements for the Cadence Palladium, Mentor Graphics Veloce, and Synopsys ZeBu product families. Hero manages emulation jobs end-to-end, including model compilation, automated emulator selection, and running regressions.

With the features described above, Hero 2.0 can help design firms significantly improve their return on emulator investments. With better utilization, design teams can perform more thorough verification and, in some cases, reduce time-to-market. Some benefits of Hero are that teams can:

- Share emulation resources optimally across teams
- Mix SA and ICE jobs to keep emulators busy 7x24
- Pipeline compilation and emulation jobs to maximize productivity and throughput
- Monitor emulator usage characteristics to understand utilization and improve sharing policies
- Improve visibility to regressions and individual tests, making engineers more productive

Summary

Semiconductor companies face daunting challenges related to chip design and verification. Challenges include increased competition, larger, more complex designs, time-to-market pressures, and limited budgets for infrastructure and tools. Increasingly, design firms are turning to hardware-assisted verification technologies such as emulators to improve verification efficiency and reduce the time required for regression tests.

Altair Hero is unique in the industry. It brings the same rich scheduling and workload management features commonly used in software-based verification to multiple hardware-assisted verification platforms. It is the only scheduler designed to optimize resources across multiple emulators and hardware-assisted verification platforms.

Learning More

To learn more about Altair's end-to-end hardware emulation resource optimizer, Altair Hero, visit www.altair.com/hero.

To learn more about Altair Accelerator and other industry-leading solutions for EDA workload management, visit www.altair.com/accelerator.