SIX SMARTER SCHEDULING TECHNIQUES FOR OPTIMIZING EDA PRODUCTIVITY

Stuart Taylor – Altair / December 9, 2021

Introduction

Semiconductor firms rely on software tools for all phases of the chip design process, from system-level design to logic simulation and physical layout. Ensuring that designs are error-free requires extensive verification before fabrication. Engineers need to use hardware and software resources as efficiently as possible. Given the enormous investment in tools, design talent, and infrastructure, even minor improvements in server farm efficiency can significantly impact the bottom line. As a result, verification engineers and IT managers are constantly looking for new sources of competitive advantage.

Workload management plays a crucial role in helping design teams share limited resources, boost simulation throughput, and maximize productivity. In this paper, we discuss six valuable techniques to help improve design center productivity:

1. Monitoring, measuring, and optimizing the EDA infrastructure
2. Employing high-throughput scheduling
3. Adopting a license-first scheduling policy
4. Mapping and optimizing design flows and simulations
5. Optimizing hardware emulation workloads
6. Leveraging the cloud to augment on-premises resources

By adopting these techniques and products in the Altair® Accelerator™ portfolio, organizations can realize higher levels of efficiency and performance and dramatically improve productivity with smarter workload scheduling.

Customer Challenges: Why the Need for Advanced Optimization Techniques in Semiconductor Design

There are few industries more competitive than electronics design and manufacturing. Semiconductor designers are under constant pressure to deliver faster, cheaper, more reliable designs. They also need to deliver new features and functionality with each product generation and beat competitors to market.

Challenges include increasing design complexity and higher gate counts, multiple projects with overlapping deadlines, and maximizing the use of expensive software tools. Some common challenges faced by semiconductor design teams are illustrated in Figure 1.

 © Altair Engineering, Inc. All Rights Reserved.

Figure 1 – Common challenges in EDA environments

Verification is the process of ensuring that semiconductor designs perform and function as expected. Modern devices increasingly involve system-on-a-chip (SoC) implementations that can have hundreds of millions or even billions of gates. Verification becomes exponentially more complex as the number of gates (and thus the number of states) increases. Engineers run extensive verification and regression tests as they iterate on designs. Given the high cost of committing a design to silicon, designs must be error-free before tape-out.

Never Enough Simulation Capacity

Verification requirements are outstripping the capacity of simulation environments. Engineers could once rely on new, faster processors to boost simulation throughput. However, performance gains from Moore's law have largely stalled. For the past decade, engineers have seen only modest increases in single-threaded processor performance.[1] To increase verification capacity, customers need to take an "all of the above" approach taking advantage of innovation in other areas, including:

- New software tools to better manage and optimize verification coverage
- Increased use of parallelism at both the level of workflows and individual tools
- New scheduling techniques to optimize the use of infrastructure and software licenses
- The use of special hardware to accelerate simulations in the form of emulators and FPGAs

Diverse EDA Workloads

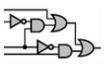
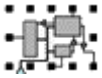
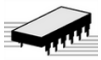
The high-level chip design process drives the mix of EDA workloads. The tools used will typically vary depending on the project. Table 1 shows typical design phases along with the types of tools used at each phase[2]. Design efforts tend to be highly iterative. Engineers implement features in a design, verify them through simulation, make refinements as necessary, and rerun simulations.

[1] See Moore's law has stalled. Where does processing go from here? Redsharknews.com, August 2021

[2] Table 1 is inspired by the Integrated circuit design page on Wikidpedia. This list has been expanded to include additional verification techniques and examples of software tools used at each stage. This is only a partial list of tools. Also, some of the tools are used in multiple design phases.

Some verification jobs are compute-intensive and time-consuming. For example, register-transfer level (RTL) simulations involve simulating the logical behavior of a device, often as it executes code. RTL simulations can run at 10K-100K simulated cycles per second and take hours or days to execute. Gate level simulation (GLS) can be even slower, running at just 10-100 simulated cycles per second and taking days or even weeks to complete depending on the size of the design. Running verification jobs quickly and efficiently is critical to meeting time-to-market goals.

Table 1 – The high-level semiconductor design process

	High-level Phase	Description	Typical Applications
	System Specification	Requirements analysis, die size estimates, and functional analysis	Cross-functional teams evaluate customer needs, market opportunity, and design feasibility
	Architectural/System-level Design	Functional specification expressed in software	Altair® Activate®, SystemC, SystemVerilog, Simulink, etc.
ENTITY test port a: in; end ENTITY;	Functional Design & Logic Design	A register-transfer level (RTL) model is specified in a high-level design language HDL – Verilog, SystemVerilog, or VHDL	RTL Simulation: Synopsys® VCS®, Cadence® Xcelium™ Parallel Logic Simulation, and Questa®
	Circuit Design	RTL models are synthesized into gate-level models and then into a circuit-level design	Altair® PollEx™, Cadence® Genus™, Synopsys® Design Compiler® NXT, Cadence® Spectre® Simulation Platform, and Synopsys® PrimeSim™ HSPICE®
	Formal Verification	Mathematical methods provide a supplementary way of verifying device functionality	Cadence® JasperGold® Formal Verification Platform, Synopsys® VC Formal™, Mentor® Questa® Formal Verification
	Emulation	For some designs, emulators simulate device behavior using programmable hardware	Cadence® Paladium® Z1/Z2, Mentor® Veloce®, Synopsys® ZeBu®
	Physical Design	High-level circuit design – floor-planning, clock-insertion, placement, and routing	Cadence® Virtuoso®, Cadence Innovus™, Synopsys® IC Compiler™ II, Synopsys® Fusion Compiler™, Mentor® Calibre InRoute™
DRC LVS ERC	Physical Verification and Signoff	Extensive verification related to power, signal integrity, timing, IR/EM	Cadence® Physical Verification System (PVS), Mentor® Calibre™, Synopsys® IC Validator™, Ansys® Redhawk-SC™
	Fabrication	Manufacturers may have in-house fabrication facilities or rely on third-party foundries	Fabrication facilities employ various in-house and commercial tools
	Packaging, Testing, Production	Fabrication facilities manufacture, package, and test devices for clients post-design	Fabricators test wafers, sometimes "binning" parts based on results – wafer final test (WFT) or electronic die sort (EDS)

Diverse Compute Requirements

Server farm administrators need to deal with the challenge of multiple designers, each facing their own project deadlines, running the tools described above on a shared compute infrastructure. Different tools run best on different types of hardware.

Digital simulation is typically the most common simulation workload. For digital simulation, engineers tend to favor hardware that delivers high single-threaded performance. Simulation servers typically feature high clock speeds, large L3 caches, and high memory bandwidth to speed processing and minimize license checkout time. Some tools, such as Mentor® Analog FastSPICE™ with its AFS MCP operating mode, can exploit multi-core and multi-node parallelism.[3] Servers with higher core counts and fast interconnects may be preferred for analog simulations.

Placement and routing applications used at the physical design stage typically require large amounts of physical memory. Some workloads may involve emulation, where jobs run on highly specialized FPGA-based emulators programmed to emulate the functionality of a particular design. Because of these diverse requirements, server farms often have different servers dedicated to specific applications, as shown in Figure 2. Administrators need to schedule workloads on the infrastructure where they run most efficiently to maximize overall throughput.

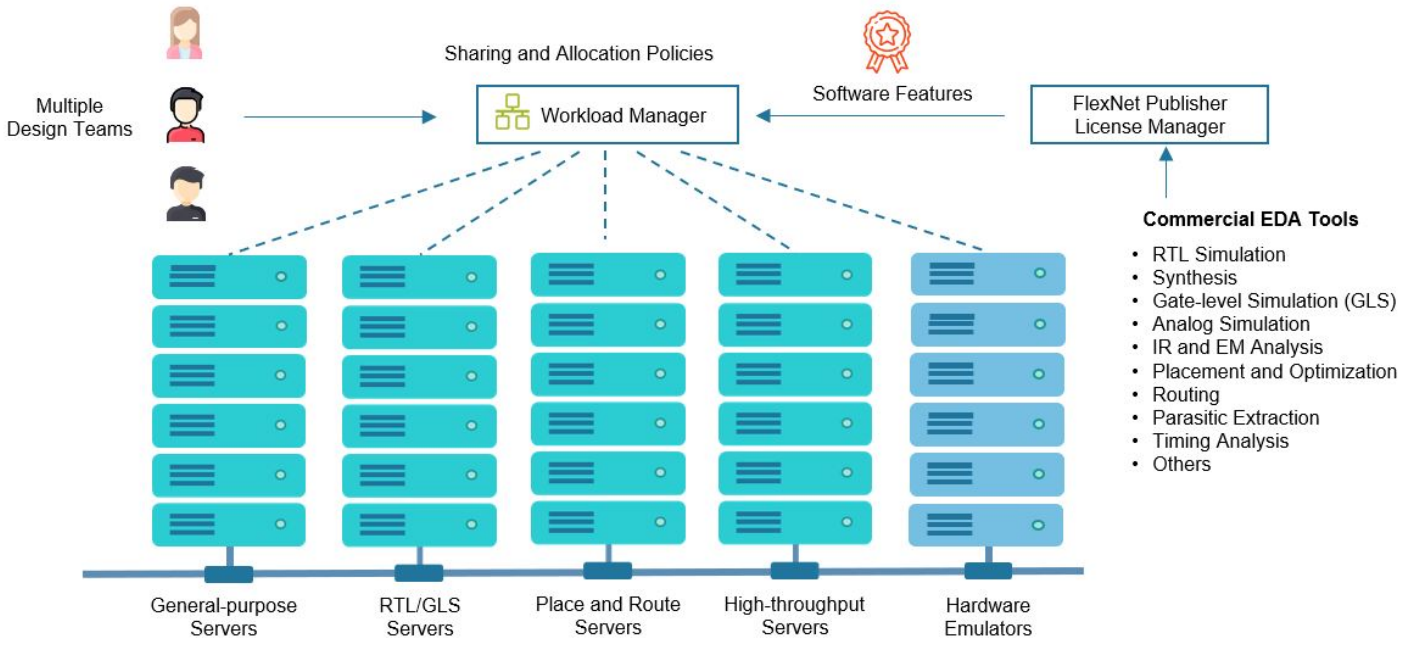


Figure 2 – Server farms have multiple types of servers optimized for different EDA workloads

Workload Management Plays a Key Role

For all these simulations, workload management plays a critical role in improving throughput and productivity. Workload managers determine how infrastructure resources and software licenses are shared among project teams and dispatched to server farm resources. They consider details such as resource sharing policies, license and asset ownership, workload priorities, project deadlines, resource reservations, and more. Focusing on workload management is often the most productive path forward for design teams to squeeze more performance from existing hardware and software assets.

[3] See Planet Analog article - Analog FastSPICE Multi-Core Parallel solution offers up to 50x higher performance

Six Techniques to Improve the Effectiveness of Your EDA Environment

Below we describe six techniques and best practices to help organizations get more out of their EDA verification environments. Implementing these practices and solutions can help improve job throughput, resource utilization, and overall productivity. In some cases, these techniques can also help reduce costs by avoiding overspending on software licenses and maximizing the use of on-premises and cloud-based infrastructure.

1. Monitor, Measure, and Optimize

A truism in business is that "if you can't measure it, you can't manage it." To improve EDA efficiency, understanding utilization is an excellent place to start. Users should begin by running monitoring tools to get a baseline understanding of key metrics related to license and infrastructure utilization. Understanding workload patterns, resources in short supply, and where bottlenecks are occurring can often lead to simple optimizations. EDA administrators should ask the following questions:

- Are some simulations queueing for excessive periods while assets sit idle?
- Are licenses and infrastructure being allocated to the most critical projects?
- Could in-demand license features be shifted to faster node types to reduce checkout times?
- Can bottlenecks be alleviated by shifting some non-critical jobs or workflows to different times?

When measuring utilization, it is essential not to confuse "slot utilization" with actual resource utilization. For example, a workload manager may report asset utilization in the range of 85-90%. However, this metric depends on the resource requirements associated with each job and may not reflect the actual resources used. For example, an application may request 4GB of memory and two cores from the scheduler but only use 1GB and one core. Ideally, monitoring and reporting tools should have visibility to workload manager-level metrics and actual hardware and license resource utilization for each EDA job. Altair provides a complete portfolio of tools to help electronics manufacturers monitor and measure how license and infrastructure resources are being used and get more out of their infrastructure investments.

Altair® Monitor™

Deploying Monitor is an excellent way to understand license and infrastructure utilization and identify and remove bottlenecks. Monitor provides users and administrators with real-time insight into software license availability, usage, job status, and more. Monitor uses current and historical data to help optimize license spending and utilization. It can be used with other Altair tools to provide an up-to-date view of how software and infrastructure assets are used. Armed with this information, administrators can tune scheduling policies to maximize utilization and improve overall productivity.

Altair Mistral™

Mistral is a lightweight application monitoring tool used in EDA server farm environments to monitor CPU, memory, and I/O usage. Using Mistral, administrators can get the most out of their server farm environments by quickly identifying rogue jobs or storage bottlenecks affecting throughput. Better visibility to how applications use resources such as CPU and memory helps organizations fine-tune resource requirements in the scheduler, enabling more efficient operations. Mistral provides telemetry related to resource usage to help organizations plan capacity more effectively. It also helps organizations decide on the optimal compute environment for different types of EDA applications.

Altair Breeze™

Breeze is another valuable tool in EDA environments that helps solve software deployment problems. While Mistral provides I/O profiling in a low-impact manner and can be applied to jobs on mass, Breeze gives detailed analysis of all I/O activity and is typically used selectively. Verification engineers and server farm administrators can use Breeze to profile application I/O and ensure that files are stored in the right place. For example, suppose Breeze determines that a licensed application performs significant file I/O while licenses are checked out. In that case, I/O may be redirected to fast local memory-based file systems or solid-state drives rather than using slower NFS file systems. This type of optimization speeds throughput and helps organizations use expensive software license features more efficiently.

The detailed nature of Breeze traces is also helpful in enabling hybrid cloud deployments by automatically detecting what files, programs, and libraries are accessed by each application. Armed with this information, administrators are better positioned to understand data dependencies and the best applications for deployment in cloud environments.

2. Maximize Productivity with High-throughput Scheduling

It is not uncommon for EDA sites to run workflows comprised of thousands or even millions of jobs per day. As processors and simulators get faster, the overhead associated with scheduling becomes a more significant concern. For each job dispatched, schedulers need to consider a variety of factors, including:

- Job resource and license requirements
- Time window policies for queues or hosts
- Availability of eligible hosts and dynamic load conditions
- Job dependencies
- Sharing policies and the relative priorities of jobs and workflows

Evaluating these constraints and placing jobs involves significant overhead. For EDA workflows, managing job dependencies at scale is critical. A challenge with many traditional workload managers is that they provide only rudimentary support for dependency management. We describe dependency management in detail later in this paper when discussing mapping and optimizing design flows and simulations.

Another critical consideration for schedulers is that not all jobs and workflows are created equal. In environments with limited license features, organizations cannot afford low-priority jobs consuming previous licenses while business-critical workflows stall. Schedulers need capabilities such as job preemption and resource reservations so that urgent jobs can take precedence over less critical workloads.

The internal design of the scheduler can also affect throughput. Most workload managers take a "cycle-based" approach to scheduling, evaluating job placement at pre-set intervals. Cycle intervals are typically 5 seconds or more depending on the size of the environment and volume of jobs. In addition to placing workloads, the scheduler also needs to respond to requests from various clients, including command-line utilities and web interfaces. Suppose the scheduling cycle is short (meaning that workloads are scheduled frequently). In that case, users may see messages such as "batch system not responding" on a busy cluster or experience timeouts. With traditional schedulers, administrators need to manage this tradeoff between scheduling frequency and cluster responsiveness. Cycle times are frequently tuned higher in busy environments to ensure that the scheduler is responsive to user commands.

A simplified example in Figure 3 illustrates this challenge. Users continuously submit jobs (or flows comprised of multiple job steps) at various times, represented by the arrows along the top of the diagram. The scheduler seeks to allocate resources and start jobs on optimal server types as quickly as possible. With a cycle-based scheduler, newly submitted jobs cannot be evaluated until the beginning of the next cycle. This leads to delays and resource underutilization, as illustrated in the top half of Figure 3.

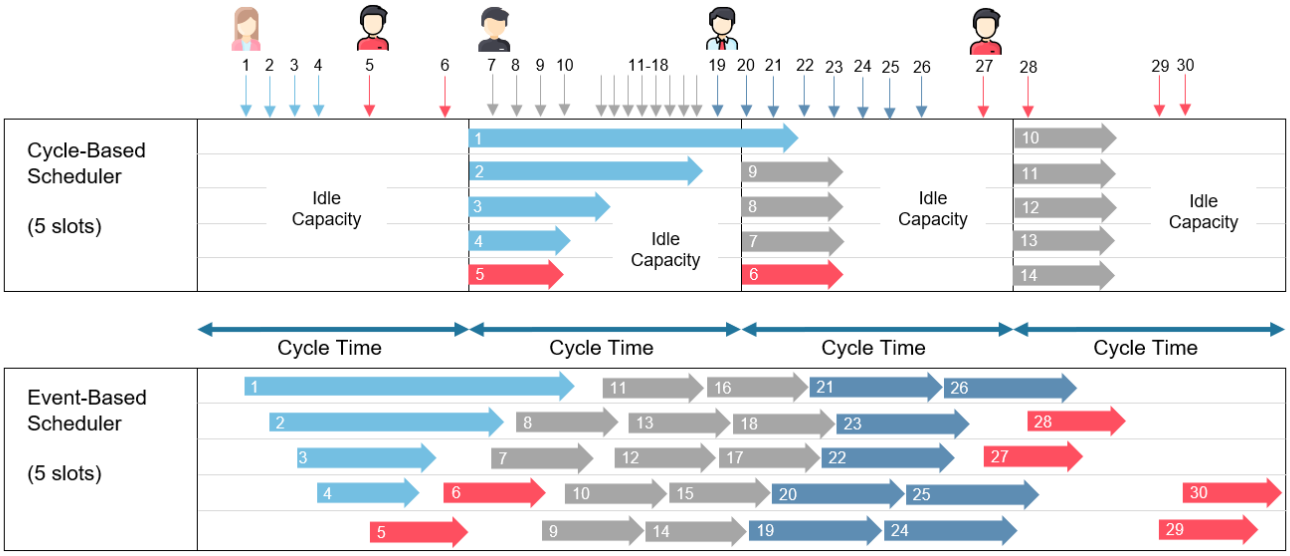


Figure 3 – Event-based schedulers dispatch jobs immediately, resulting in high throughput and utilization

While some cycle-based schedulers have optimizations that allow jobs submitted by the same users with identical resource requirements to reuse allocations without the cost of an additional scheduling cycle, there are still fundamental limitations. Some cycle-based schedulers have optimizations that allow jobs submitted by the same users with identical resource requirements to reuse allocations without the cost of an additional scheduling cycle. Cycle-based schedulers still have fundamental limitations, however. In contrast, event-based schedulers (illustrated in the lower portion of Figure 3) can respond in milliseconds and place jobs without waiting until the start of the next cycle. An event-based scheduler reduces job pending time, yielding higher throughput and delivering better cluster utilization by minimizing idle capacity.

Altair Accelerator

Accelerator is a 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. Accelerator provides complete visibility into what is 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 Accelerator help ensure 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.

With configurable alerts and notifications, users and cluster administrators can be notified of various exception conditions at configurable intervals to improve administrator productivity. Accelerator can provide alerts in the case of long-running jobs, non-responsive hosts, licenses checked out for excessive periods, and license expirations.

For environments needing even higher throughput, customers can optionally deploy Altair® Accelerator™ Plus, a hierarchical scheduler architected to offload the base scheduler. Accelerator Plus provides individual users with what is essentially a "personal scheduler," enabling them to queue short jobs independently. From the main scheduler's perspective, the workload is dynamically converted into fewer, longer-running jobs — an easier option for a scheduler managing high core counts. Perhaps more importantly, the queue master is relieved of common but onerous tasks such as job submission, job queries, and reporting. Furthermore, an Accelerator Plus session can front multiple base schedulers, allowing for even higher scaling or blending of different hardware resources. For high job counts with similar resource needs, **Accelerator Plus can help organizations realize a dramatic 6-10x throughput improvement** without changes to the workloads themselves.

3. Adopt a License-first Scheduling Policy

Since licenses are typically expensive and in high demand, optimizing license usage is critical. Ideally, organizations want to keep licenses busy 24x7. While some batch jobs and multi-step workflows can run around the clock, others cannot. Some tools that require licenses are run interactively. Other licensed applications may run outside the control of a workload manager (referred to as "out-of-queue"), making license tracking and optimization more difficult. In some cases, licenses may be reserved in advance. While this is often necessary and well-intentioned, it can be a disaster from a utilization standpoint. Figure 4 shows varying levels of license utilization.

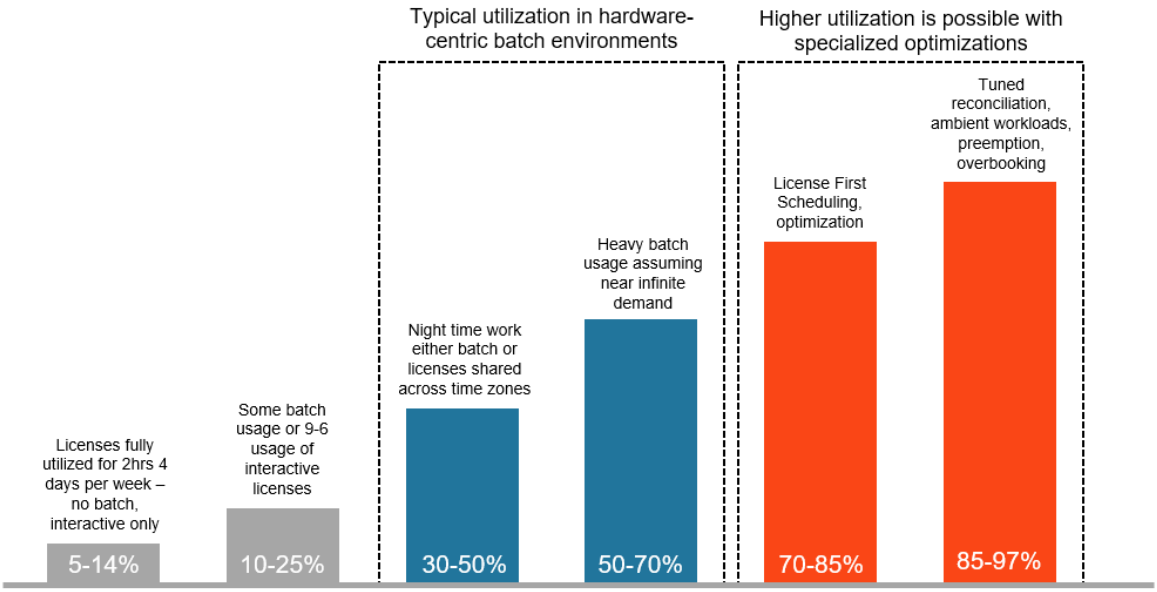


Figure 4 – Achieving optimal license utilization is challenging

Unoptimized environments can exhibit poor license utilization in the range of **5-25%**. By implementing batch queueing and attempting to use licenses around the clock (by running simulations overnight or sharing licenses across time zones), utilization can range from **30-70%,** assuming sufficient workload demand. Organizations can achieve **70%** or better utilization with additional optimizations described below, translating into significantly higher throughput and cost efficiency.

License Scheduling

When dispatching EDA jobs, the scheduler needs to take license availability into account. Commercial EDA tools typically connect to a FlexLM server at the start of execution and attempt to check out a license. If a license is not available, the job will fail with an error. Therefore, the scheduler must ensure that license features are available in sufficient quantities before scheduling and starting a licensed job. Some common strategies for handling licenses resources are described below:

- **Track license availability using a counter.** The most straightforward approach to license management is to treat license features as "consumable resources" using a simple counter to meter allocation. While this solution may work in some ideal cases, it is impractical with most real workloads. The scheduler's view of license availability can quickly diverge from reality without visibility to the actual state of checked-out licenses. Real-world Issues include out-of-queue usage, applications that check out licenses at multiple execution stages, and failed jobs that hold onto licenses for excessive periods.
- **Check license availability before dispatching jobs.** This approach is better than a simple counter, but it too has issues. License daemons can be slow to respond, often taking 3-15 seconds. For short-running jobs, this is an unacceptable amount of overhead and can lead to poor utilization. Also, previously dispatched jobs may be slow in checking out licenses, resulting in FlexLM overstating the number of licenses available. In this case, a license may appear to be available. However, jobs can fail once dispatched due to a lack of available licenses.
- **Periodically check license availability with a configurable "linger" period.** A more sophisticated approach periodically checks license availability and dispatches jobs against that availability with a resource limit that lingers. In other words, when a job is dispatched, the scheduler views the license as allocated for a configurable time even though the application may not have checked out the license yet. This approach is better than the previous two, but it is hard in practice to configure this linger period optimally. If the linger period is too short, the scheduler can incorrectly assume that an application has checked out a license when it has not, resulting in jobs failing due to unavailable licenses. Conversely, if the linger period is too long, the scheduler will assume that licenses are in use even though they are available, resulting in jobs that should be eligible to run pending and license underutilization. In practice, it is difficult to get more than 50-70% license utilization using this approach.

License Matching

To address these complexities, Accelerator employs license matching to help organizations achieve better license utilization. Accelerator maintains an internal table, as illustrated in Figure 5, which matches license checkouts reported by FlexLM to jobs known to the scheduler. License matching considers the number of total licenses, license checkout information reported by FlexLM, and workload-related information from the scheduler. The scheduler can identify out-of-queue checkouts (checkouts without a corresponding job), definite matches (Matched), and cases where a job has requested a license but the license is not yet used (Requested/Not Used).

Resource User				VOV Info					License Mgr. Info				
#	User	Host	Matching Status	Project	Job Id	Job Status	Age[1]	Ask[2]	Got[3]	License Server	Handle	Age[4]	Match Type
1	taylor	sdc-c6qa-1	Out-of-queue(0)						1	6306 lm	507903235	15m31s	none
2	taylor	sdc-u14s-1	Matched(1)		517121082	RETRACING	1m47s	1	1	6306 lm	507903237	26s	sure
3	taylor	sdc-u14s-2	Matched(1)		517121086	RETRACING	1m48s	1	1	6306 lm	507903232	26s	sure
4	brown	sdc-c6qa-1	Matched(1)		517121074	RETRACING	1m47s	1	1	6306 lm	507903234	26s	sure
5	taylor	sdc-u15s-1	Matched(1)		517121078	RETRACING	1m45s	1	1	6306 lm	507903230	26s	sure
6	brown	sdc-u15s-2	Matched(1)		517121096	RETRACING	1m47s	1	1	6306 lm	507903239	26s	sure
7	brown	sdc-u16s-1	Matched(1)		517121080	RETRACING	1m47s	1	1	6306 lm	507903236	26s	sure
8	taylor	sdc-u16s-3	Requested/NotUsed(1)		517121084	RETRACING	1m47s	1					
9	brown	sdc-c7qa-1	Requested/NotUsed(1)		517121088	RETRACING	1m48s	1					
10	taylor	sdc-c7qa-1	Requested/NotUsed(1)		517121072	RETRACING	1m47s	1					
11	taylor	sdc-u16s-3	Requested/NotUsed(1)		517121076	RETRACING	1m47s	1					
12	taylor	sdc-c6qa-1	Requested/NotUsed(1)		517121090	RETRACING	1m47s	1					
13	brown	testdev-04	Requested/NotUsed(1)		517121094	RETRACING	1m48s	1					

Figure 5 – License matching with Accelerator and Monitor

With license matching, the scheduler maintains an up-to-date view of how licenses are being used while checking license availability only periodically. If a site has a total of 30 modelsim licenses, and seven licenses are checked out, as shown in Figure 5, a simple license scheduling scheme that polled FlexLM would conclude that 23 licenses were available. However, by considering FlexLM and workload data together, Accelerator knows that six jobs have already started requiring licenses but have not yet checked them out. Hence, the number of available licenses is 17 (23 minus the six jobs yet to check out their allocated licenses). License matching can be complicated under the covers. A few examples are provided below:

- **Sure match vs. best match** – In some cases, a single licensed job will run on a host. In this case, we can be sure that we have a correct match (i.e., a "sure" match) because both the resource manager and FlexLM report only one licensed job on that host. In other cases, multiple licensed jobs may be dispatched to the same host in quick succession. FlexLM may report multiple licenses checked out to the same host, but it will not be clear which license corresponds to which job. In this case, the license matching algorithm will need to guess what job corresponds to what license handle (i.e., a "best" match). 100% accuracy does not matter in this case. What is important is that enough licenses are allocated to support the workload.
- **Old match** – The workload manager may know that a job has finished, but the license manager data may still show the license as checked out. This may occur because the license server has not been polled recently, and the table maintained by the license monitor contains old data. This case is referred to as an "old match." In this case, the scheduler will recognize that it is safe to reuse the license feature immediately.
- **Too early to tell** – The opposite of the "old" case is where the scheduler dispatches a licensed job, and the license manager data still shows the license as available. It is premature to mark the job as "requested not used" until Monitor provides the following update from the license manager. Hence, the license match algorithm marks these jobs as "Too early to tell."

License Matching Drives High Throughput

License matching together with an event-based scheduler can deliver higher throughput and utilization than other approaches. An example is provided in Figure 6 comparing a scheduler that polls FlexLM before each job submission with the license matching scheme algorithm used by Accelerator.[4]

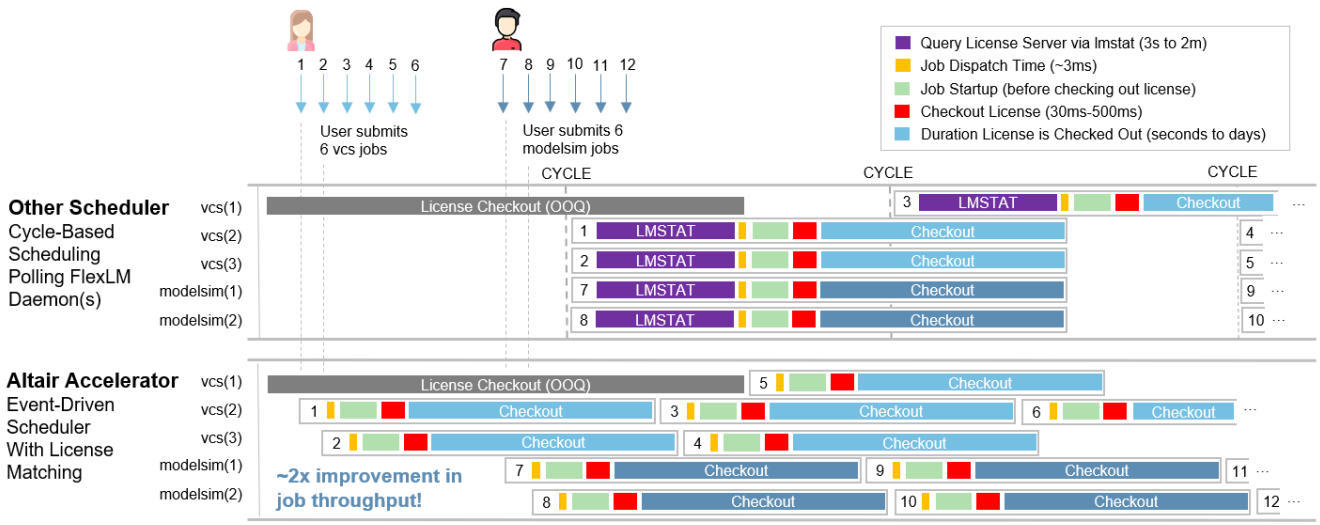


Figure 6 – License matching combined with event-driven scheduling delivers better job throughput and utilization

[4] Please note: Figure 6 is an oversimplification. Polling the license daemon via the lmstat command occurs prior to the job being scheduled. We've drawn it this way for simplicity to indicate that the job cannot be dispatched until lmstat verifies that a license is available.

Polling the FlexLM license manager daemon via the lmstat command (shown in purple) can take anywhere from 3 seconds to 2 minutes, depending on the size of the environment and how busy it is.[5] With a simple polling approach, jobs can only start after lmstat confirms license availability.

Once the job starts running, it may do some preamble work (shown in green in Figure 6), such as loading data files before actually checking out the appropriate license. In the case of a cycle-based scheduler polling the FlexLM daemons, licenses tend to be underutilized. Throughput is low because time is wasted either waiting for lmstat to respond or waiting for the start of the following scheduling cycle, as shown in the top half of Figure 6.

By maintaining an internal license-matching table, Accelerator can dispatch jobs immediately without waiting for lmstat to respond. Throughput is further enhanced by event-driven scheduling described earlier that allows jobs to start immediately without waiting for the next scheduling cycle. In this example, Accelerator completes roughly ten jobs (not including one job submitted out of queue) in the time that a competing scheduler completes five jobs – a **2x throughput advantage**.

Actual improvements in production environments will depend on many factors. However, Altair's license-first scheduling optimizations have been shown to improve license utilization by roughly 20% in many cases, resulting in overall license utilization in the range of 70-85%. With further optimizations enabled by Accelerator, including tuned reconciliation, ambient workloads, and preemption overbooking not discussed here, organizations can reach utilization as high as 85%-97%.

4. Map and Optimize Design Flows and Simulations

So far, our discussion has focused mainly on batch and interactive jobs. Many processes in semiconductor design are repetitive and involve multiple discrete steps where the next step depends on the output of the previous step. These multi-step design flows are common in logic design, synthesis, placement, and routing. A typical design environment may have thousands of different design workflows that can take hours or even weeks to complete in some cases.

For example, a basic physical block design may involve ~2,000 jobs in a flow graph, with perhaps 20 steps executing in parallel. There may be ~300,000 jobs in a flow graph for regression testing, and parallelism is typically limited only by software licenses and infrastructure resources.

Managing workflows manually is not an option. Problems that cause workflows to fail can significantly impact schedules, so customers need automated solutions to make workflows resilient. Some organizations use custom scripts or turn to dependency management tools such as make, commonly used in software build environments.[6]

These solutions are less than ideal, however:

- Makefiles and custom scripts are challenging to maintain and debug.
- Scripts are often maintained by one or two key individuals, leaving an organization vulnerable if these employees or contractors leave and take critical knowledge with them.
- Different aspects of the design cycle involve different tools and expertise. Customers may find themselves with many discrete makefiles and scripts authored by different people.
- With scripted workflows, administrators and site managers have no visibility into flow execution, making it difficult to monitor progress and identify and fix issues.

Fortunately, solutions exist that make it easy to create, manage, and monitor large-scale EDA workflows with complex dependencies.

---

[5] Lmstat is a command-line utility included in FlexNet Publisher that can be used to query a FlexLM daemon about license availability.

[6] Make is a popular build automation tool included with most Linux distributions. Make automatically builds executables and libraries from source code based on rules expressed in a Makefile.

Altair® FlowTracer™

FlowTracer is an advanced design flow development and execution platform that provides users with unique flow visualization and troubleshooting capabilities for greater productivity. FlowTracer provides flow visualization, analyzes flows, and identifies inherent parallelism to optimize the use of compute resources and speed up workflow execution. FlowTracer tracks inputs and outputs of individual workflow steps, automatically detecting changes as they occur. FlowTracer provides:

- Tracing to capture, manage, and execute flows automatically
- A color-coded GUI that can display millions of jobs, enabling issues to be quickly identified, debugged, and resolved
- Dependency awareness, including the ability to continue running workflows from the point of failure without the need to restart an entire workflow
- Shared visibility to workflows enterprise-wide, enabling design teams to collaborate on complex designs more effectively

The use of FlowTracer is illustrated in Figure 7. With FlowTracer, engineers can easily visualize the progress of scalable workflows using a variety of views and zoom in on regions of a large flow. Workflow nodes include files (shown as ovals) and executable jobs steps (shown as rectangles) in the FlowTracer GUI. Completed job steps are shown in green, running steps in yellow, and any failed steps in red. Steps that still need to run and are considered invalid are shown in purple, and queued job steps are shown in blue.

If jobs fail, users can explore the reason for the failure using an integrated node editor. Using this interface, they can rectify problems and resume the flow from the point of failure without needing to rerun the entire workflow. Similarly, if workflow steps are queued for long periods and fail to run, these problems become evident to designers. Designers can drill into a stalled workflow step, determine why a job is pending (perhaps a resource requirement that cannot be met), and rectify the problem to expedite the flow.

FlowTracer enables designers to manage many large and complex flows interactively. For example, engineers might invalidate parts of the flow (to force re-execution) or exclude parts of a workflow not relevant to a result to save time and resources. Designers often make minor changes to steps in a workflow that will not affect computed results. For long-running flows, organizations cannot afford redundant calculations that tie up resources and licenses. In this case, engineers can use "barrier steps" in FlowTracer to prevent unnecessary re-computation, a critical capability for long-running regression tests.

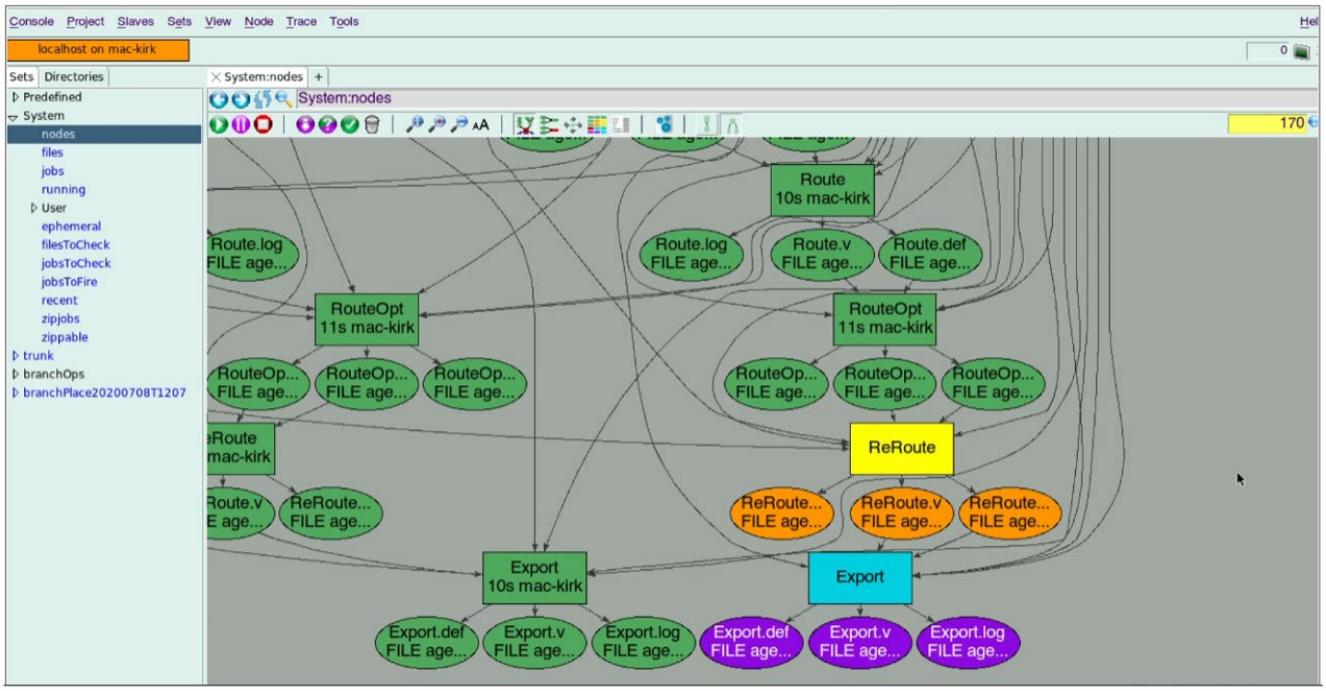


Figure 7 – FlowTracer helps visualize complex EDA workflows

FlowTracer's user-friendly views (grids, nodes, flow charts, and dependencies) allow complex workflows to be easily deployed and shared across organizations for flow standardization and increased collaboration. Flows become self-documenting and easier to understand and manage than makefiles or complex scripts.

Designers can also employ branches to execute steps conditionally or run multiple experiments in parallel to optimize designs and improve product quality. For example, reducing area and power in a semiconductor design is essential both for profitability and viability. Designers may use FlowTracer for floor planning activities, exploring multiple routing and placement strategies to determine which is optimal.

Workflows are critical in electronic design environments. Unlike other workload managers, where job dependency management and workflows are an afterthought, workflows and dependencies are integral to the design of Accelerator. With better workflow automation, organizations can create higher-quality designs, realize higher throughput and productivity, and avoid processing delays and errors impacting schedules.

5. Optimize Hardware Emulation Workloads

To verify large designs, some organizations turn to specialized hardware emulators. Emulators implement design logic using programmable FPGAs or custom processors. Leading platforms include Cadence® Palladium® Z1 and Z2 emulation (commonly used with Cadence® Protium® X2 for software validation), Mentor Graphics® Veloce®, and Synopsys® ZeBu®. Sharing hardware emulators among design teams brings unique challenges. For example, 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 to emulation jobs. The number of emulation boards assigned varies with the complexity of the design.

Hardware emulation can provide a dramatic performance advantage over software-based simulation techniques, often running between 500K to 2M simulated cycles per second (~1MHz). While slower than the target hardware, hardware emulators can verify functionality orders of magnitude faster than software-based logic simulations. This performance advantage makes emulation particularly valuable from a time-to-market perspective. Also, emulators and software validation platforms enable hardware and software to be developed and debugged concurrently without software engineers waiting for functional hardware prototypes, helping compress the design cycle even further.

Unlike software-based simulators, where workload managers are commonly used, verification engineers have traditionally managed emulation workloads manually. Engineers perform tasks such as compiling designs, assigning virtual devices, and deciding which physical boards to program to support each emulation task. While this manual approach may work 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, just as software licenses and server infrastructure are optimized in the case of software-based simulations.

Altair® Hero™ (Hardware Emulation Resource Optimizer)

Hero is an end-to-end enterprise job scheduler designed for hardware emulation environments. Hero is a vendor-agnostic solution 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 accelerated regressions. A single Hero job queue can dispatch emulation jobs to multiple emulator instances of the same or different architectures.

Hero brings new capabilities to hardware emulation environments:

- Policy management, including FairShare and preemption
- Soft reservations enable organizations and users to reserve blocks of time on the emulator
- A rich GUI enables users to see job status identify the root cause of any failing jobs
- Visibility into emulation-specific metrics to improve hardware asset optimization and organizational planning

Bringing resource-aware scheduling to hardware emulation environments can be a game-changer for design teams. With Hero, design teams can improve automation and share emulators more efficiently. By maximizing the use of expensive emulators, organizations can dramatically reduce the time required for regression testing and positively impact design schedules.

6. Leverage the Cloud to Augment On-premises Resources

While most design firms operate on-premises server farms, organizations increasingly augment local capacity with cloud resources. This is especially true during the busy period before tape-out. Before tape-out, organizations may temporarily acquire additional licenses to increase simulation capacity. The temporary use of cloud resources to augment on-premises capacity during busy periods is called "cloud bursting." Cloud bursting can be "coarse-grained," where supplementary cloud resources are brought online for periods of days or weeks. It can also be "fine-grained," where cloud instances are dynamically provisioned as required.

Increased use of the cloud is being enabled by continuous improvements in cloud functionality and security along with new software tools that make it easier to tap cloud resources. While cloud computing can help improve capacity during busy periods, verification engineers still face practical challenges implementing bursting solutions. Among these challenges are:

- **Data handling and movement** – Ensuring that required datasets are available locally or in the cloud and pre-staging data on cloud storage, including cost-effective object stores such as Amazon S3 or Azure Blob Storage.
- **Managing cloud-based file systems** – Deploying NFS servers based on block storage, cloud-native file systems such as Amazon EFS, or third-party distributed file systems such as WekaIO.
- **Software licensing** – Serving software licenses to cloud-based instances from on-premises license servers via VPNs.
- **Managing cloud spending** – Ensuring that cloud resources are fully utilized and do not sit idle once provisioned.

Accelerator and Hybrid Clouds

Accelerator makes it straightforward to leverage public cloud resources and realize an elastic EDA server farm infrastructure. Accelerator's cloud-friendly features include:

- **Zero configuration execution hosts** – The Accelerator daemons on execution hosts can come and go without reconfiguring the scheduler on the master host.[7] The daemons self-discover resources (memory, cores, etc.), making the architecture ideal for dynamic cloud environments where instance types can change.
- **Optional shared file system** – Accelerator doesn't require a shared file system between master and slave nodes, providing flexibility and simplifying cluster deployment and management.
- **Preemption and execution host turnover** – As workload requirements change, preemption and execution host turnover allow new cloud instances to be deployed optimally sized to new workloads for more cost-efficient operation.
- **Container support** – The Accelerator daemon can advertise properties such as hasdocker or hassingularity, enabling transparent execution with a user's preferred container framework.
- **Elastic agents and daemons** – Cloud-based clusters can scale dynamically based on varying workload demands.
- **Breeze** – Breeze is particularly useful in hybrid cloud environments. By tracing I/O activities, users can understand file dependencies and performance characteristics to determine what workloads are candidates for execution in the cloud and which files need to be replicated or moved.

[7] The term VOV describes a project to build a Tcl-based system to automate the chip design process at UC Berkely in 1991, developed by Andrea Casotto. This technology is the basis of Altair Accelerator. See: The VOV Story: TCL in EDA and Religious Wars.

There are multiple ways to deploy Accelerator in cloud and hybrid cloud environments. Approaches include naïve bursting scenarios (where instances are deployed manually or via scripts and bound to a local cluster via a VPN connection) or operating separate, dedicated cloud-based clusters. Rapid Scaling is an Accelerator feature that enables cloud instances to be provisioned, started, stopped, and de-provisioned automatically, subject to policy and changing workload demand.

Accelerator Rapid Scaling

Accelerator can distinguish between workloads waiting for licenses from those waiting for hardware. It will only scale cloud resources when sufficient software licenses are available. By auto-scaling resources, administrators avoid pitfalls such as under-provisioning (leading to poor throughput and license utilization) and overprovisioning (leading to lingering cloud instances and overspending on resources).

Accelerator Rapid Scaling can shut down instances when they have been idle for a configurable period and can use both On-Demand and Spot cloud pricing to reduce costs further.[8] The scheduler can auto-select the most efficient instance type depending on the workload to maximize throughput and license use efficiency. Administrators can provide alternate instance types if the preferred instance type is no longer available in their region or availability zone. In production settings, Accelerator Rapid Scaling has been shown to reduce cloud spending by 50% or more.[9]

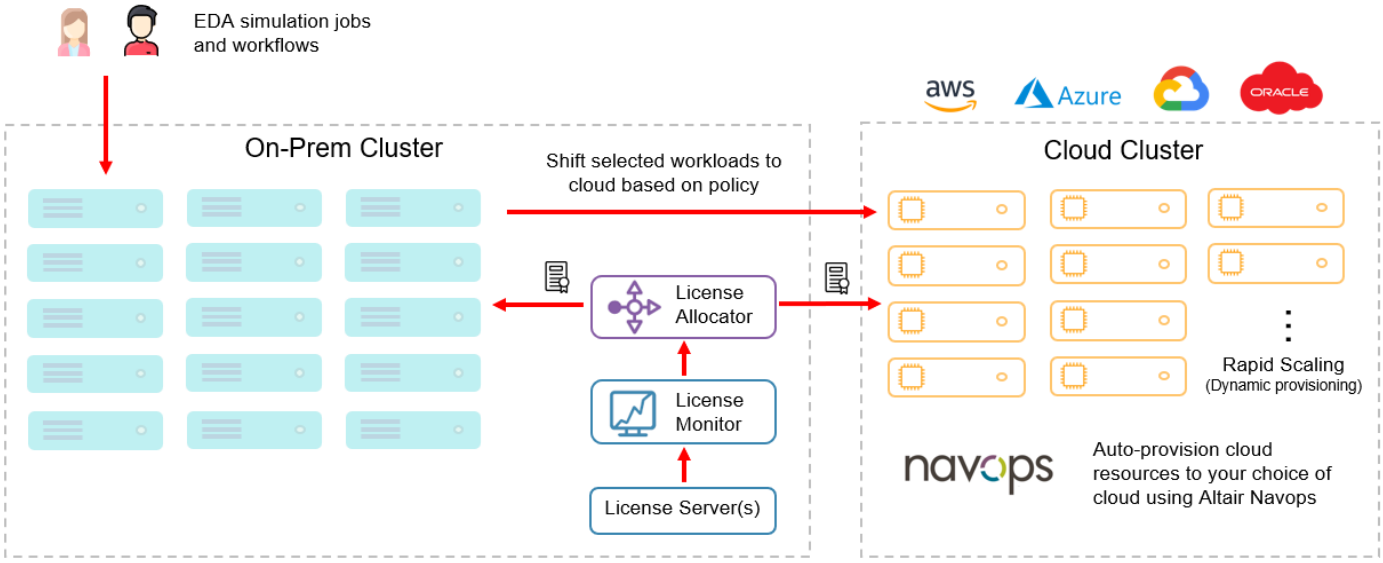


Figure 8 - Tapping hybrid cloud resources with Accelerator Rapid Scaling or Altair NavOps

Figure 8 illustrates a typical Accelerator hybrid cloud deployment. In this scenario, the on-premises cluster is augmented by a separate Accelerator cluster in the cloud with Rapid Scaling. Replicating an on-prem compute environment in the cloud can be a massive undertaking, especially from a data management and file system perspective. A more conservative approach takes selected workloads and allows them to be run in the cloud; this allows data provisioning and tool/flow validation efforts to be more focused. Accelerator provides a consistent user interface in both environments, allowing existing tooling and scripts to be leveraged. A key aspect of this hybrid model is the ability to share licenses from the on-prem queue with the cloud-based queue: Altair® Allocator™ provides this capability. The FlexLM daemons serving license features run on-premises along with Monitor to track license utilization. Allocator

[8] Amazon claims that EC2 Spot Instances can be up to 90% less expensive than on-demand instances for selected workloads.

[9] See the Altair Annapurna Labs customer case study. Rapid Chip Design in the Cloud.

facilitates distributing licenses between nodes on local and cloud-resident clusters just as it would in an on-premises environment with multiple clusters. A more complex model using Accelerator Plus to unify both queues can also be employed.

Altair® NavOps®

Another valuable tool for deploying and managing hybrid cloud environments is NavOps, a modern tool that helps organizations quickly deploy and scale clustered server farm environments across multiple clouds. It leverages powerful automations to manage clusters in a "no-ops" fashion and helps customers right-size deployments by auto-selecting optimal instance types depending on the EDA workload. NavOps is production-proven at an extreme scale, having dynamically provisioned cloud environments comprised of more than one million vCPUs to accelerate engineering simulations.[10]

NavOps is application-, resource-, and budget-aware and provides real-time insights into workloads and spending with complete visibility into HPC cloud resources. By combining sophisticated automation with cloud spend management, NavOps can help boost efficiency, reduce cloud costs, and improve time-to-results, ultimately improving revenue and profitability.

A Complete Suite of EDA Workload Management Solutions

Altair provides a complete suite of EDA workload management solutions to help design firms implement the six optimizations described above. Customers can start using Altair Accelerator for high-throughput scheduling of EDA workloads and gradually introduce additional workload management capabilities as requirements evolve. The full suite of Altair EDA workload management solutions is shown in Figure 9.

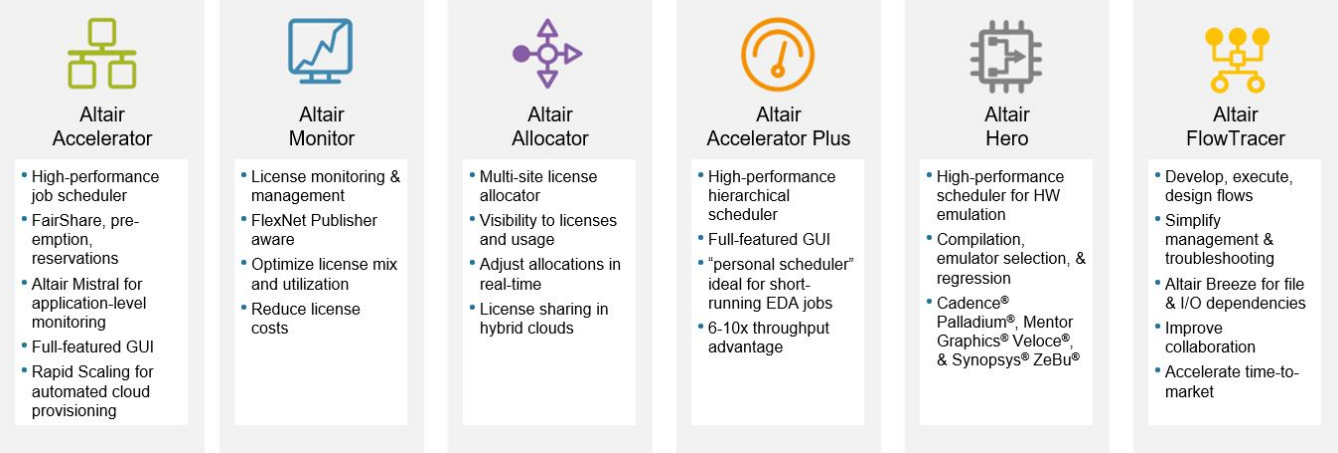


Figure 9 - Altair provides a full suite of EDA workload management tools

Realizing a More Effective EDA Environment

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. Improvements in processor speed have slowed, causing organizations to look for new ways to improve efficiency.

Workload management is a crucial area for optimization. Even marginal improvements in workload throughput and resource utilization can drive significant productivity improvements. Altair can help organizations improve efficiency in six valuable ways:

[10] See the article Mission Is Possible: Tips on Building a Million Core Cluster.

1. **Monitor, measure, and optimize workloads.** Administrators can improve sharing policies and express resource requirements more precisely by monitoring license and resource allocations with Altair Monitor. Improved monitoring leads to better utilization and helps ensure that critical project deadlines are met.
2. **Implement a high-throughput scheduler.** By leveraging Accelerator and its high-throughput, event-based scheduler, design teams realize reduced scheduling latency and faster job turnaround leading to better productivity. By running jobs faster and more efficiently, they also gain simulation capacity leading to higher-quality, more thoroughly tested products.
3. **Employ license-first scheduling.** EDA tool licenses are the most valuable commodity in most verification environments. However, maximizing license utilization is hard. Even sophisticated organizations may achieve only 50-70% license utilization. By using Monitor and Allocator with advanced license matching techniques, sites can dramatically improve license utilization and throughput, in some cases achieving utilization of 90% or higher.
4. **Map and optimize design flows and simulations.** While multi-step workflows are common in EDA environments, many schedulers support only rudimentary job dependency management. FlowTracer captures flows, provides visualization, identifies inherent parallelism opportunities to optimize resource usage, and enables collaboration around workflow execution.
5. **Improve hardware emulation efficiency.** Most EDA firms use workload management, but ironically the most expensive hardware assets in the data center, hardware emulators, are usually managed manually. Hero brings advanced scheduling and resource sharing to leading hardware emulation platforms. This provides greater flexibility and control and enables organizations to get more productivity from their hardware emulation investments.
6. **Leverage the cloud to augment on-premises resources.** While using cloud resources during busy periods sounds like a good idea, the devil is in the details. Accelerator, Monitor, and Allocator provide a complete solution for hybrid cloud deployments. Organizations can easily tap their choice of clouds to improve capacity with efficient cloud auto-scaling and policy-based allocation of software licenses.

Learning More

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

Discover breakthrough optimization opportunities in your own infrastructure with the help of Altair's semiconductor experts. Get started today.