MEETING JOB SCHEDULING CHALLENGES FOR ORGANIZATIONS OF ALL SIZES

Altair / August 18, 2020

Introduction

Every semiconductor design group uses some sort of job scheduler, whether it is selected by a corporate IT department or by the group itself. At a high level, the function of a job scheduler is simple: Be aware of what is in the queue and what hardware and software license resources are available and make good decisions about what jobs to schedule when. In practice there are a several subtle issues that are just as important, including job mix, prioritization, and ease of support.

While every company is unique, the issues surrounding job scheduling are often related to company size or, more accurately, the size of the compute farm and design teams.

Small companies with limited resources (100-500 cores) tend to focus on using their job scheduler to solve a single problem, such as maximizing use of precious simulation licenses. In this case, job schedulers are typically managed by the designers themselves using a commercial/freeware solution or even relying on internally developed scripts. With limited resources, small companies are keenly aware of the resource tradeoff between investing effort in the scheduler versus doing the “real work.” Bringing an expensive enterprise-class job scheduler into the organization would be both cost- and resource-prohibitive. The most important feature in a job scheduler for this type of organization is making optimal use of a limited number of software licenses, since buying more is rarely an option. Small companies rely heavily on all the exotic features of a scheduler to maximize software license usage.

A medium-sized company (500-3,000 cores) depends more on the visibility and transparency of what is being scheduled since there are likely multiple design teams competing for the same resources. The administration and management of job schedulers typically falls on IT resources shared with several other groups. With multiple teams competing for the same resources, it becomes essential to be able to customize the job scheduler to accommodate varying requirements and have the flexibility and visibility to ensure that resource allocations are consistent with company priorities.

Departments in large organizations (more than 3,000 cores) have a different set of issues, as their available resources are used for a wide range of purposes. Large organizations are typically supported by a sizable IT department managing a single, general-purpose job scheduler. A semiconductor design team within a large entity may find itself at odds with the internally supported solution since their specific job mix might require more queue slots and higher dispatch rates than the job scheduler can handle. This is typically the case with library characterization, design verification, and large tape-out jobs. For this group the most important feature may simply be the ability to remove their problem jobs

Common Scheduling Problems

Despite the differences between use cases for the organizations mentioned above, there is a lot of commonality in scheduling workloads and thus in the type of job scheduler that can address their needs.

Software License Utilization – The biggest problem facing any organization looking to adopt a scheduler is maximizing utilization of their compute resources and software licenses. Semiconductor design is unique because the cost of required software licenses is many times higher than the cost of the hardware on which the software runs, so it's important to select a job scheduler that ensures maximum license utilization.

Complexity – One of the most common problems is the complexity of the job scheduler and the expertise required to administer the solution. For example, maintaining a complex scheduler that requires IT expertise will not suit a small or medium-sized company since it takes time away from design resources.

Adaptability – Job schedulers need the correct feature set to handle the typical mix of jobs a semiconductor design group produces. This is typically a varied mix consisting of a large number of small jobs that run for short periods of time, as in the case of design verification or library characterization, along with a limited number of extremely large jobs that run for hours or even days (e.g., during the final runs of tape-out). It's particularly important that small, easy-to-schedule jobs do not cause excessive wait times for large jobs. There is also a need for a provision to remove any roadblocks when a lower-priority, long-running job is consuming a critical resource. Having the right mix of features that facilitate setting and managing job priorities and organization policies is an essential requirement for a job scheduler in the semiconductor design environment.

Scalability – If a job scheduler is too slow or cannot scale to a large number of jobs, it invariably penalizes small jobs, as they either run too slowly or, in the worst case, cannot even be queued up without being manually batched to run. This encourages designers to combine small jobs into larger ones, prohibiting them from being executed in parallel and occupying resources such as software licenses for a long time, ultimately having a negative impact on large jobs as well.

Visibility – Transparency is another important consideration for job schedulers. If the scheduler's queue and resource allocations are too opaque, design groups that are the true users will not be able to adapt their job submission behavior based on the current state of the project, queue, and resource availability. Some aspects of design, such as functional verification or fault grading, are never complete. However, it makes little sense to flood the compute farm with many of these jobs when critical or interactive jobs are waiting. In contrast, it makes more sense to look at availability and see what other jobs are competing for resources before making job submission decisions or self-imposed job limits. All organizations follow this methodology to some extent, running large jobs overnight while keeping some interactive resources free during the day.

Altair Accelerator™: Job Scheduling for All Organizations

Altair Accelerator is a high-throughput, enterprise-grade job scheduler designed to meet the complex demands of semiconductor design, EDA, and high-performance computing (HPC). It's a highly adaptable solution capable of managing compute infrastructures from small, dedicated server farms to complex, distributed environments.

Scalability and Performance

Accelerator's efficient memory model allows it to keep track of job counts in excess of a million at any given time. Unlike cycle-based solutions where jobs are submitted at a fixed time (e.g., every minute or so), Accelerator's event-driven architecture allows it to dispatch jobs at any moment available resources are identified. Dispatch latency is only a few milliseconds. These two factors eliminate the attractiveness of batching up small jobs into large ones, reducing latency until the last of a collection of small jobs completes, while increasing the utilization of the server farm by running as many small jobs at the same time as the hardware can accommodate.

Figure 1 illustrates the elapsed time impact of cycle-based scheduler dispatch latency on the turnaround time for 100,000 jobs running on 500 cores at an average of 5 minutes per job. The results clearly illustrate Accelerator's performance advantage over alternative cycle-based solutions with a turnaround time advantage ranging from 23 minutes to ~2.5 hours for a single iteration. This turnaround time advantage has a compounding affect when the total number of jobs and expected number of iterations are taken into consideration for a typical semiconductor design environment.

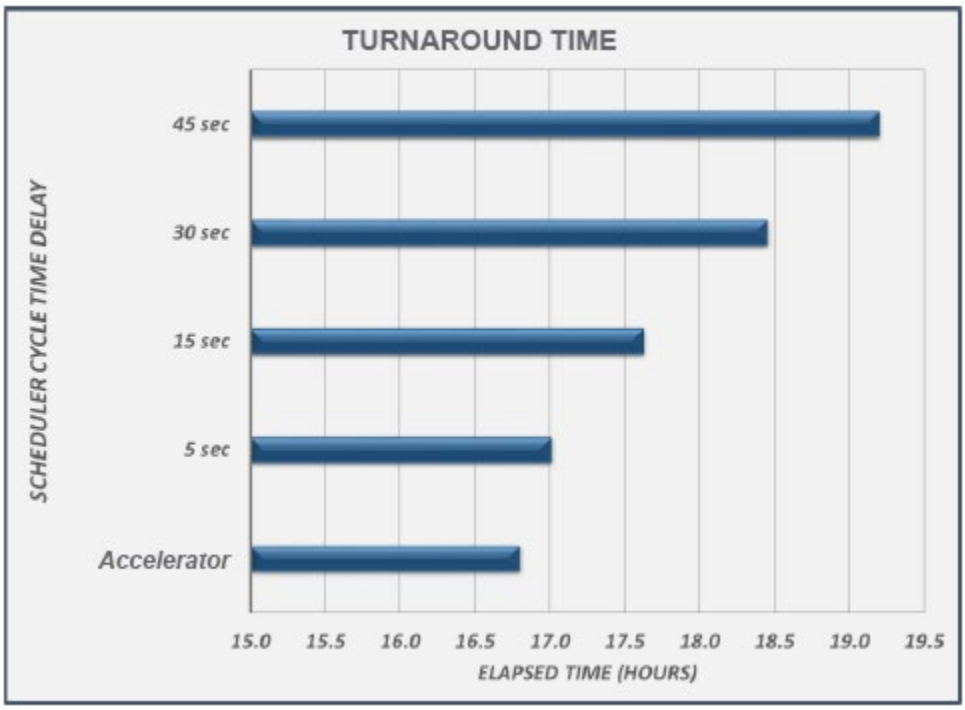


Figure 1: Turnaround Time Impact of Job Scheduler Dispatch Latency

Software License Utilization

Accelerator is a software-license-aware job scheduler with full visibility into available and in-use licenses, along with a full understanding of licenses required by queued jobs. For example, Accelerator will not dispatch a job knowing that the required software license is not yet available, since that would prohibit another job with an available license from using the allocated compute resources. Unlike other batch systems where knowledge of license availability is limited to a simple running job count, Accelerator keeps track of actual license availability for jobs both in-queue and out-of-queue. Accelerator also keeps track of whether jobs are actively using software licenses they've claimed and reallocate them to waiting jobs, a procedure known as license reconciliation. This is useful for composite jobs in which scarce licenses are only used for a portion of the job.

The resulting holistic view of both hardware and software resources enables efficient job scheduling. Figure 2 represents the license utilization of the simple workload mentioned above comparing various job dispatch delays. The results demonstrate the impact of the job scheduler architecture and its efficiency on overall software license utilization. In this scenario, Accelerator achieves a software license utilization of 98.4% compared to maximum license utilization ranging from 82.7% to 95.1% for comparable cycle-based implementations. At software license utilization of nearly 80% it becomes increasingly difficult to justify procurement of additional software licenses. Most organizations opt to increase their license utilization, which can be nearly impossible if the root cause is job scheduler dispatch latency.

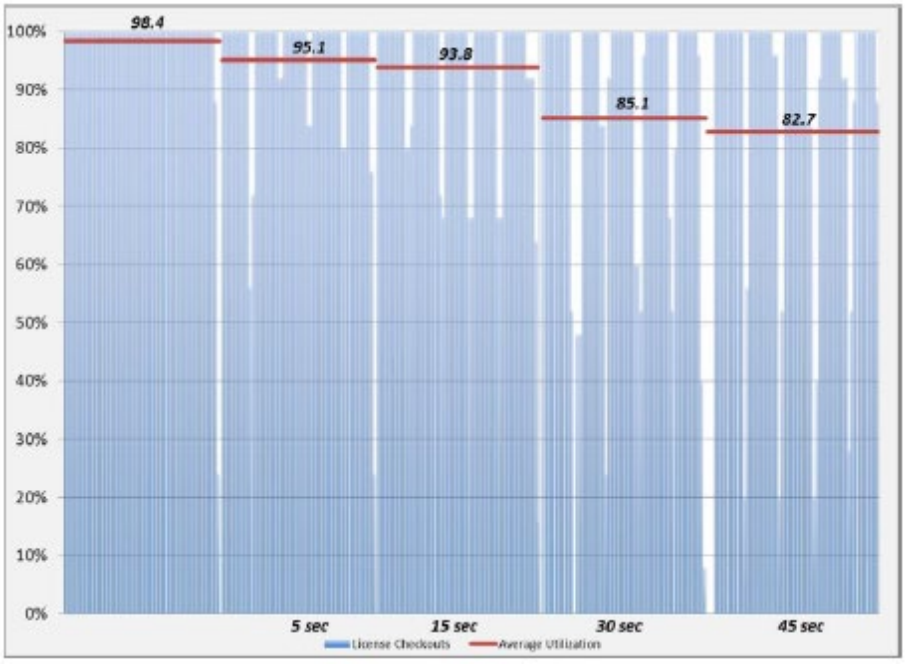
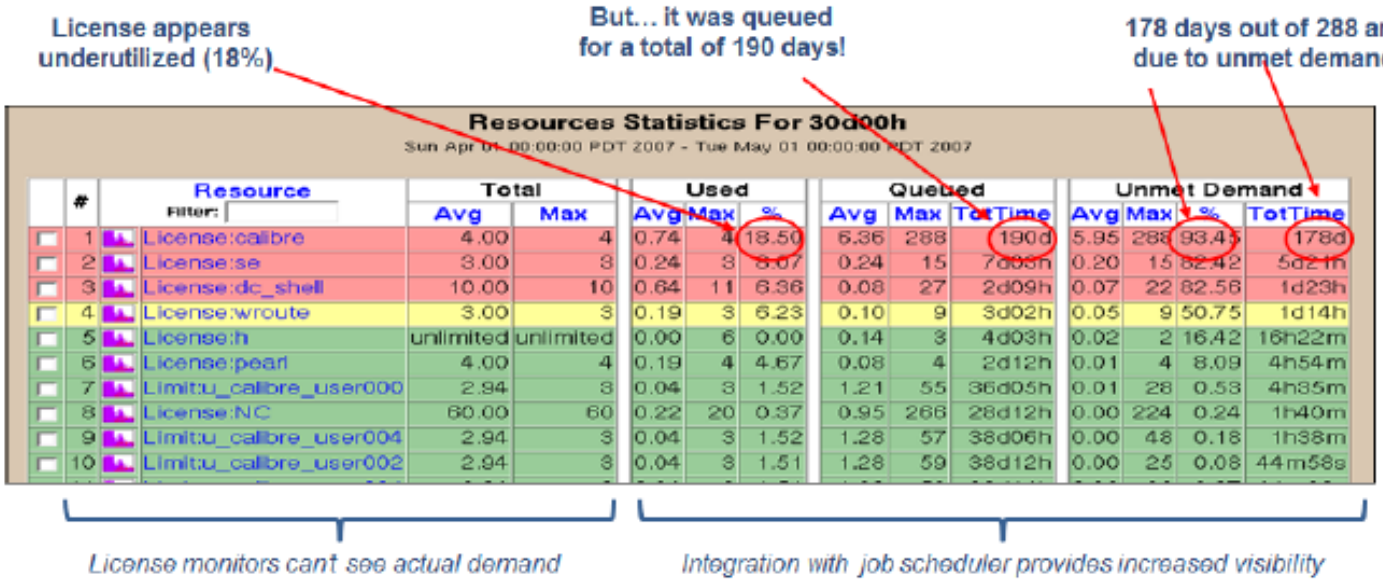


Figure 2: Job Scheduler Dispatch Delay Impact on Software License Utilization

Visibility

Accelerator has a single queue for all jobs, the simplest approach for users. Other schedulers end up with a proliferation of designated task- or resource-specific queues (e.g., tape-out queue), limiting end-user visibility and creating unnecessary confusion.

Accelerator has several mechanisms for displaying both running jobs and jobs waiting in the queue. This provides end users with a high level of visibility into the status of their jobs and allows system administrators to make changes to meet organizational priorities such as changing job priorities or manually preempting a job.



#	Resource	Total Avg	Total Max	Used Avg	Used Max	Used %	Queued Avg	Queued Max	Queued TotTime	Unmet Demand Avg	Unmet Demand Max	Unmet Demand %	Unmet Demand TotTime
1	License:calibre	4.00	4	0.74	4	18.50	6.36	288	190d	5.95	288	93.45	178d
2	License:se	3.00	3	0.24	3	8.07	0.24	15	7d05h	0.20	15	82.42	5d21h
3	License:dc_shell	10.00	10	0.64	11	6.36	0.08	27	2d09h	0.07	22	82.56	1d23h
4	License:wroute	3.00	3	0.19	3	6.23	0.10	9	3d02h	0.05	9	50.75	1d14h
5	License:h	unlimited	unlimited	0.00	6	0.00	0.14	3	4d03h	0.02	2	16.42	16h22m
6	License:pearl	4.00	4	0.19	4	4.67	0.08	4	2d12h	0.01	4	8.09	4h54m
7	Limit:u_calibre_user000	2.94	3	0.04	3	1.52	1.21	55	36d05h	0.01	28	0.53	4h35m
8	License:NC	60.00	60	0.22	20	0.37	0.95	266	28d12h	0.00	224	0.24	1h40m
9	Limit:u_calibre_user004	2.94	3	0.04	3	1.52	1.28	57	38d06h	0.00	48	0.18	1h38m
10	Limit:u_calibre_user002	2.94	3	0.04	3	1.51	1.28	59	38d12h	0.00	25	0.08	44m58s

Figure 3: Accelerator Gives Users Added Visibility

Adaptability

Job schedulers need to be equipped with tools for allocating resources according to policies. Accelerator handles this by fairshare, which allocates shares of the resources hierarchically, a share for each group and a share for each subgroup, down to the level of individuals. Within each level, it is possible to change the allocated shares quickly to ensure changing group requirements are reflected in the allotted shares.

Fairshare looks back through a window of time and attempts to ensure that everyone gets their fair proportion of total resources, in line with their shares during that period. When there is resource contention, the resulting resource shares are allocated according to a predetermined policy. Unlike a simple priority system, fairshare ensures that even low-priority jobs get their share of resources. It is equally critical that users have full visibility into what share of the compute resources their jobs are getting as it is for management to have the ability to set priorities as project objectives change or as design moves through the natural cycle toward tape-out. For sharing to work well in most organizations it needs to be visible and seen to reflect project and organizational priorities.

However, fair does not necessarily mean equal, as lower priority jobs should get fewer resources but should not be stalled or delayed forever. Figure 4 illustrates a scenario in which a single user is running a total of five jobs with varying weights ranging from 30 to 200 assigned to each of the tasks. In this example, “mars” is assigned a weight of 200 out of a total of 500, or 40% target share, but in reality it has only received a 16.75% share. As a result, “mars” is assigned a rank value of “0” as the highest-priority job to get access to compute resources, whereas during the same period “pluto” has managed to receive an actual share of 46.90% in comparison to its assigned target of 20%, hence being assigned the lowest rank of 4.

Current Status of FairShare for active groups

			Running		Share			Jobs	
	Group Name	Queued	N	%	Weights	Target	Actual	in window	Rank
1	/proj/mercury.taylor	96		0.00%	103/30/100	6.00%	7.04%	4	1
2	/proj/mars.taylor	85	8	8.57%	103/200/100	40.00%	16.75%	15	0
3	/proj/venus.taylor	91	5	14.29%	103/70/100	14.00%	7.37%	9	2
4	/proj/earth.taylor	87	5	20.00%	103/100/100	20.00%	21.94%	13	3
5	/proj/pluto.taylor	69	17	57.14%	103/100/100	20.00%	46.90%	31	4

Showing 5 out of 5 rows | Limit rows displayed: 20

Figure 4: Accelerator Provides Full Visibility into Fairshare Settings

Another important consideration facing today’s design organizations is that difficult-to-schedule jobs, those requiring many cores, specialized hardware, large amounts of RAM and scarce license resources, can be blocked by lower-priority but easier-to-schedule jobs. There are two ways in which this can happen. First, a job that requires multiple cores may wait forever if the job scheduler aggressively schedules smaller jobs on every core immediately as they are freed up. Second, a job that requires a specific resource (such as a core attached to an emulator) may be blocked by a job running on that core while it did not require any emulation resources. Accelerator addresses both issues with a reservation concept, noticing that the problem is occurring and leaving cores unscheduled for a time, waiting for enough to be freed to dispatch the large job. Reservation also provides a mechanism to ensure that jobs that do not require specialized resources do not initiate if jobs that require those specific resources are waiting in the queue.

Preemption is a powerful feature of Accelerator. This capability allows rules to be set in advance or via manual intervention to prevent long-running jobs that have already started from blocking higher-priority jobs. Accelerator features three different methods of preemption:

- Cancel and requeue: A lower-priority job is killed and put back in the queue to restart later when resources are available.
- Suspend and restart: Allows a lower-priority job to be suspended while its resources are made available to a pending higher-priority job. Upon completion of the high-priority job, the suspended job is restarted to continue.

- Soft preemption: The ability to reserve both hardware and software resources for waiting jobs.

Summary

Accelerator is a versatile high-performance, fully featured scheduler designed with EDA in mind, meeting the requirements and challenges of small, medium, and large companies.

Small companies can greatly benefit from the end-user visibility resulting from Accelerator's single-queue architecture and leverage its fairshare and preemption capabilities to maximize resource utilization without IT administration overhead.

Medium-sized companies with multiple design teams sharing a common job scheduler can leverage Accelerator's ability to easily handle a heterogeneous job mix, along with the combination of fairshare priority, reservation, and preemption to ensure that even the most diverse mixture of jobs can be efficiently scheduled with competitive job latency. Accelerator provides the visibility necessary to gain an understanding of what jobs are waiting and why.

Unlimited queue size and fast dispatch rate make Accelerator an attractive solution for large companies, eliminating the need to batch up smaller jobs. Accelerator will efficiently schedule small jobs in and around large jobs while ensuring that large, high-priority jobs are not unnecessarily delayed. Reservation and preemption allow the largest jobs to run in a timely manner. The mixture of speed, simplicity, visibility, and controllability delivers a departmental view of available resources and allows tuning them to the department's specific needs. In this scenario, Accelerator works on top of, or alongside, the existing IT/corporate job scheduler while offloading certain departmental workloads, eliminating bottlenecks and improving user experience.

Altair Accelerator's high-performance, event-based architecture combine with a rich set of policy management features and single-queue transparency, making it a smart choice for organizations of all sizes.

