SHORT-RUNNING JOBS CAN HELP OPTIMIZE YOUR RESOURCE UTILIZATION

Altair / August 18, 2020

Introduction

Semiconductor companies typically run jobs by queuing them up, then using a job scheduler to dispatch them onto available cores in server farms while pulling EDA tool licenses from a license server. There are two primary goals facing organizations, and they're somewhat at odds with each other: First, maximizing the utilization of server farms and software licenses and second, running jobs with minimal latency so users aren't delayed.

The easiest way to satisfy the requirements of high utilization and low latency is to maximize the number of short-duration jobs. Just as it is easier to fill a bucket with sand than it is with large rocks, short jobs give the scheduler increased flexibility in what jobs to run and when. Short jobs will not block or occupy a resource for long periods and are therefore not likely to impede the progress of higher-priority jobs arriving in the queue.

Job Scheduling Challenges in Today's Environment

Many job schedulers are not tuned to handle large numbers of short jobs because their maximum queue size is limited. In today's semiconductor design environment, there are inevitably tasks that generate huge numbers of short jobs. In practice, if short jobs are encouraged, the queue can quickly grow to millions of jobs or more.

In addition, the dispatch rate of many of today's schedulers is too low, a byproduct of their cycle-based architecture, which limits them to dispatching jobs at a regular interval such as a minute boundary. If a job runs for just a few seconds, the scheduler will not notice and the available hardware and software licenses will sit idle until the next cycle boundary.

While there are many other important facets of job schedulers (e.g., crash recovery and ease of installation), this white paper is focused on the challenge of scheduling large numbers of short-duration jobs.

Using short jobs might sound like an impossible task, but somewhere between 30% and 50% of all jobs in a typical semiconductor organization run for less than 1 minute, and over 50% complete in under 5 minutes. Fewer than 10% of jobs run for 4 hours or more. The graphs below are actual job mixes from real semiconductor development organizations, one an IP company and one an SoC company (Figure 1).

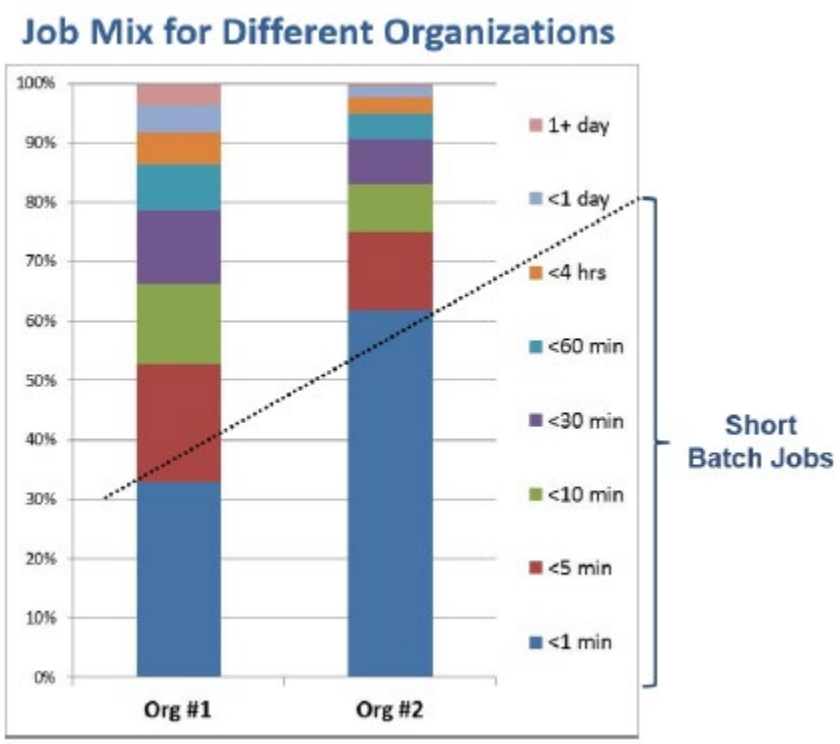

Figure 1: EDA Job Mix for Different Organizations

This is looking across an entire organization. Some engineering groups, such as physical design, do not generate many short jobs, but other parts of the organization, such as design verification and library characterization, do. This emphasizes the importance of running the entire job mix from a single queue so the scheduler has both short-running and long-running jobs to trade off against each other.

In practice, many semiconductor organizations adapt to their existing scheduler's inability to efficiently handle short jobs by batching up small jobs into larger ones, which poses additional challenges:

- Jobs may wait longer to start because the scheduler penalizes them for being long-running jobs.
- Jobs take longer to run because there is limited or no opportunity to run smaller jobs in parallel.
- Overall server farm throughput is reduced since there may be thousands of cores but only dozens of jobs to be run.

For instance, if a large workload of jobs, each taking a minute, is run on 60 servers, then a single job finishes approximately every second. This means that machines and licenses free up fast for later high-priority jobs, even those that might require many cores simultaneously, such as running place and route on a modern EDA tool.

If the scheduler has the capability to reserve machines for large jobs requiring multiple servers, even a job that requires 10 servers will normally only wait about 10 seconds for them to become available. Without reservation, 10 servers will never free up at the same time since the scheduler would immediately launch a new job — so, paradoxically, running many short jobs makes scheduling large jobs easier.

Altar Accelerator™: Performance and Scalability

Altair Accelerator is an agile, fully featured scheduler optimized for today's EDA workloads. The most important difference between Accelerator and other popular schedulers is its event-driven architecture, which allows it to schedule a new job immediately when compute resources and software licenses become available.

Traditional cycle-based schedulers operate differently. They have a heartbeat, typically a minute, when they wake up and see what resources have been freed up and what new jobs have arrived in the queue, dispatching jobs in accordance with available resources at the cycle boundary. If the heartbeat is a minute, there will be an average delay of 30 seconds before a job is launched.

Figure 2 illustrates the elapsed time impact of cycle-based scheduler dispatch latency on the turnaround time of 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 ~20 minutes to ~2.5 hours for a single iteration. This turnaround time advantage has a compounding effect when the total number of jobs and expected number of iterations for a typical semiconductor design environment are considered.

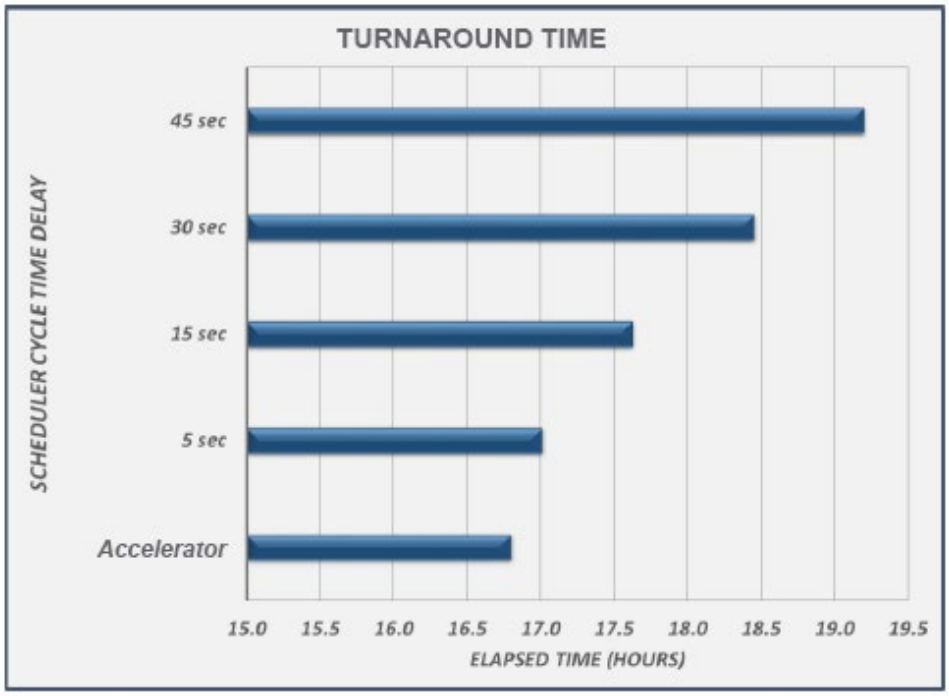

Figure 2: Turnaround Time Impact of Job Scheduler Dispatch Latency

The difference between event-driven and cycle-based schedulers is minimal for large jobs, as a delay of a minute is insignificant when compared to a job that lasts several hours. However, consider the case when the same workload is split into many short jobs. In this scenario, it is not only scheduler efficiency and its ability to keep the server farm loaded that's important, but also the delay associated with scheduler latency and dispatch time. When each short job is delayed 30 seconds or more there is a compounding effect since it is not only that the current job that has to wait, but also any later jobs that use the same resources.

Another consideration for running short jobs is that there are simply a lot more jobs to run if the same workload is performed in short jobs rather than large ones. In this scenario, scheduler overhead becomes a critical factor since it adds up quickly with a large number of jobs. The scheduler's memory footprint and dispatch latency are other important considerations because they impact resource consumption and scalability.

Accelerator was designed to be a versatile, fully featured, high-performance scheduler for EDA job mixes. Its small memory footprint and low dispatch latency enable it to handle over 2M jobs on a 32-bit server and over 5M on a 64-bit server. Its event-driven architecture allows it to respond immediately when resources such as server or software licenses are freed up.

Benefits for End Users

Two communities of users greatly benefit from Accelerator's features.

For chip designers who create the jobs and wait for them to complete, Accelerator significantly reduces the delay from the time a job is created to getting results back, improving designer productivity and user experience. Designers are no longer under pressure to batch up a lot of work and create big workloads as they would with cycle-base schedulers. Furthermore, Accelerator is fast enough that interactive users do not see the difference between running locally or on a server farm.

The second benefit is at the enterprise and IT level. Accelerator achieves higher utilization of hardware and software license resources than competing schedulers, leading to less need for additional hardware and software licenses, or faster turnaround for the same investment.

These two benefits mean designers don't need to spend their time trying to "beat the scheduler" by doing unusual things such as adapting their methodology to that of the scheduler. The only action needed is to break jobs up into shorter jobs.

Two features of Accelerator that help with this are controllability and observability. Management needs to be able to dynamically set and alter policy about which jobs are most critical, then see that the changes are effective.

It's important that every user can see that scheduling is being done fairly and understand what is delaying their jobs. It is much more efficient for an organization if a single queue of jobs is scheduled on all available hardware rather than compensating for scheduler limitations with multiple queues, dedicated pools of hardware, and dedicated software licenses. Inevitably these dedicated resources turn out to be heavily underutilized, resulting in excess cost for the organization in both dollars and delays.

As an example, a company was unable to fully utilize their software licenses with their existing scheduler, maxing out at around 80%. Accelerator maximized their software license utilization using the exact same compute resources and software licenses, getting close to 100%. As the company acquired more hardware and software licenses, they continued to achieve near 100% utilization on their investment (Figure 3).

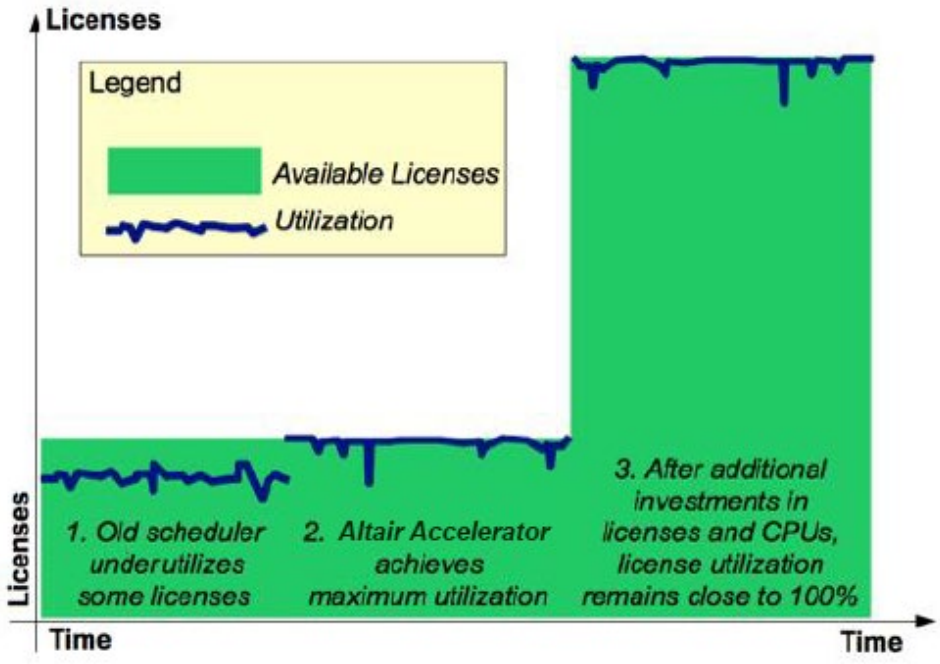

Figure 3: Accelerator Capacity Utilization

Another customer did an analysis of their existing scheduler versus Accelerator for dispatching a large workload of 300,000 jobs. The existing scheduler ran the workload in about 39 hours. Accelerator ran the same workload on the same hardware in about 10.5 hours. The theoretical limit is 9.5 hours (Figure 4). In this case, Accelerator only required a quarter of the compute resources and software licenses the existing scheduler demanded, resulting in significant financial savings.

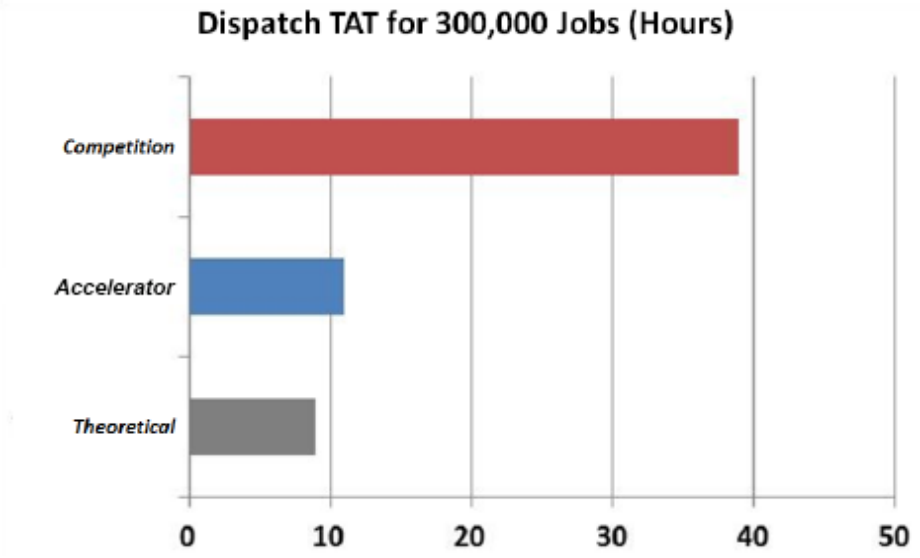

Figure 4: Accelerator vs. Competition TAT Advantage

Finally, consider the example of characterizing a cell library that has 500 cells at 3 voltages and 16 process corners. That is a total of 24,000 jobs, all of them short. If each job is delayed by 45 seconds due to cycle-based scheduling, it accounts for 300 hours of additional server time. The amount of delay this will introduce depends on the number of servers allocated to the project. Assuming a customer is using 20 servers, the dispatch delay alone quickly adds an additional 15 hours to turnaround time.

Large jobs that are processed on the same hardware can easily be scheduled fairly so that if a big, long-running place-and-route job is launched, the reservation system can quickly acquire the necessary hardware software licenses and launch the job in a timely manner.

While the advantages mentioned above apply to most cases, there are corner cases where customers may not fully realize the advantage mentioned above. These include:

- Access to compute farms with large core count: Compute resources may already have an excellent job turnover rate. Additional small jobs formed by breaking up larger ones will have limited improvement that needs to be offset against the cost of job refactoring and scheduler overhead.
- Software license checkout time: Software license checkout takes finite time that can become significant for short jobs. The added load on the license daemon should be considered in this case.
- Uniform workload may be easier to optimize: For example, a design verification workload may benefit from being run on a separate cluster. The costs of having a separate cluster may be mitigated by DV's ability to use all resources all the time.

Summary

Altair Accelerator is a fully featured, high-performance scheduler specifically designed to handle EDA job mixes in the most efficient way, maximizing use of compute resources and software licenses while minimizing latency.

Most jobs in semiconductor development groups are short or can be made short by breaking up large jobs that have been bundled together. In that environment, Accelerator runs a representative job mix in about one quarter of the time needed by cycle-based schedulers.

