The Need for Speed: Understanding design factors that make multi-core parallel simulations efficient

Shobana Sudhakar
Design & Verification Technology
Mentor Graphics
Wilsonville, OR
shobana_sudhakar@mentor.com

Rohit K Jain
Design & Verification Technology
Mentor Graphics
Fremont, CA
rohit_jain@mentor.com

Abstract

Running a parallel simulation may be as easy as flipping on a switch with the progressive and maturing solutions available today, but do people really take full advantage of the technology? It is true that in some scenarios the overhead of communication and synchronization needed for parallel simulation can negate any substantial performance gains. However, there are scenarios where deploying the parallel simulation technology can provide tremendous benefit. A long running simulation that exercises large blocks of a design concurrently and independently is one good example.

Designers need to be aware of the factors that can inhibit the advantages of parallel simulations, even in these best-case scenarios; the main factor being inflexibility due to the way designs are modeled today. This paper focuses on these factors and is an effort to educate on best design principles and practices to maximize the advantage of simulation with parallel computing. The discussion also extends to the three main fundamental features of parallel simulations, viz., load balancing, concurrency and communication. Designers need to understand how their designs run in simulation with these factors to ensure they get the maximum out of parallel simulations.

This paper will also discuss the applicability and trade-offs of the parallel simulations technology in various user scenarios, for example, overnight grid regressions, weeklong runs and debug turnaround times. It is important that designers understand these trade-offs, so that they can selectively use this technology in scenarios which can provide maximum benefit to them and increase their productivity.

Introduction

Parallel computing is no longer a new concept in digital simulation. The industry leading simulators already have solutions that can take advantage of the tremendous advancements in multi-core technology that is available in today's computing hardware. What has not happened, however, is widespread use and adoption of this technology in the user community. Additionally, the semiconductor industry's forecasts indicate an upward trend on the number of cores available on future compute servers. This paper discusses keys for how users can think about their design in order to take advantage of these multi-core compute resources using parallel simulation. The design factors discussed in this paper are not restricted to any single EDA simulator; rather it provides guidelines to understand what makes multi-core simulations a success, no matter what simulator is used.

Motivation behind the paper

Several papers have been written to discuss the common myths surrounding parallel simulations and explain the actual realities. Primarily, two common myths that have been blown away[1] shed significant light on simulation speed-ups that can be achieved using parallel simulations.

a. The increase in processor speed of multiprocessor computers do not necessarily result in better parallel simulation performance.
b. Larger designs do not automatically qualify as good candidates for parallel simulations that can result in great speed-ups.

QuestaSim MC^2 (Multi-core Multi-computer) technology has seen widespread use and deployments in the recent past and the motivation behind this paper is to answer some of the most interesting questions raised by our customers:

- Do multi-core/parallel simulations even work?
- When do parallel simulations work? When do they not work well?
- What can be done to make it work?
- What should be used to speed-up existing regression suites – a distributed grid job or parallel simulations?

The intent of this paper is to educate customers and guide them to understand what makes a design multi-core friendly. This can help customers write designs and testbenches to be more suited for parallel simulations.

Cases of success and failures of QuestaSim MC^2 deployments and the lessons learned from them form the basis of our analysis and substantiate our suggestions in this paper.

This paper will also discuss the applicability and trade-offs of the parallel simulations technology in various user scenarios, for example, overnight grid regressions, weeklong runs and debug turnaround times. It is important that designers understand these trade-offs, so that they can selectively use this technology in scenarios which can provide maximum benefit to them and increase their productivity.

Design factors that affect multi-core parallel simulations performance

Partitioning decisions affect parallel simulations performance and speedup. The following three sub-sections discuss the main factors that affect the performance of multi-core simulations.

Load balancing

Load-balanced activity on all partitions is very important for maximizing the throughput of the parallel simulation, and it is imperative to design with load balancing in mind. Sometimes, designers only activate and test one block at a time because testing all blocks together makes simulation very time consuming. With multi-core simulation in mind, it may be possible to activate all parallel blocks in the design and take advantage of multi-core simulation to reduce overall simulation time.

Example 1:[2]
Very good load balancing inside DUT, if partitioned at *block** level:

```
# testbench      100%
#  dut           91.1%
#    block1     32.6%
#    block2     22.7%
#    block3     22.6%
#    block4     21.0%
```

Example 2:[2]
Very uneven load balancing, with high TB time, very low DUT time and possibly sequential high-level monitor/drivers:

```
# testbench    44.9
#    dut        6.3
#    interface   5.5
# package      15.7
# monitor      14.0
# driver        8.8
```

Concurrency

A design with inherent parallelism and lesser sequential behavior is characterized by complex functions that are broken down into a series of small tasks that can be performed independent of each other and in parallel. This concurrency helps a

parallel simulation where the parallel blocks of the design do not wait idly for the other blocks and can proceed with the independent computations. In addition, a high degree of concurrency ensures minimal communication between blocks and the need to sync at fewer synchronization points.

Designs that have large blocks with independent parallel activity, such as multi-core SoC with busy cores and independent functions, are natural candidates for parallel simulations and may provide significant speedups depending on the partitioning. On the other hand, designs such as Ethernet with large serial compute/processing are not good candidates and may even perform poorly as a multi-core simulation due to added communication overhead between partitions.

Example 3:[3]
Partition p3 is sitting idle most of the time, and not concurrent while other partitions are running. Such behavior leads to sequential execution of design and decrease improvement with multi-core.

```
          |p0     |p1     |p2     |p3     |
Total Time|37.49s |37.49s |37.49s |37.49s |
Idle Time |0.00s  | 2.29s |0.54s  |27.00s |
```

Example 4:[3]
None of the partitions are sitting idle and executing concurrently.

```
          |p0     |p1     |p2     |p3     |
Total Time|310.86s|310.86s|310.86s|310.85s|
Idle Time |  0.00s|  0.00s|  0.00s|  0.00s|
```

Sometimes, designs show good load balanced behavior, but it is also important to have this load concurrent. If each partitions is spending 25% of the time, but they are active one after another, then even though design is well load balanced, multi-core simulation will not give any benefit due to no concurrency between partitions.

Communication

The inter-partition communication (IPC) between blocks can add considerable overhead to a parallel simulation and can impact simulation performance significantly and even produce negative results.

If the design partitioning causes two partitions to communicate at a very high rate for data transfer, the communication overhead in this case may cause a negative impact on the simulation and it may be better to combine the design units that communicate heavily into a single partition to minimize the overhead.

It may be possible to identify blocks in the design to either keep together or separate in order for communication overhead to be kept at a minimum during parallel simulation. QuestaSim MC^2 technology has built-in tools to help analyze this traffic and provide useful information to the user to help avoid high communication.

Example 5:[3]
Designs with flat hierarchy or partitioning at very lower hierarchy will results in high number of communication ports, and hence increased traffic:

```
p3-p0: Instances:67, In_ports:10332,
Out_ports:6142
p2-p0: Instances:68, In_ports:10332,
Out_ports:6123
p1-p0: Instances:68, In_ports:10222,
Out_ports:6050
```

Example 6:[3]
Communication statistics for ports from partitions 'p0' to 'p1', lower count indicates lower communication:

```
instance_name |port_name | count       |
tb.inst1.inst2|x         |176          |
tb.inst1.inst2|y         | 60          |
tb.inst1.inst2|z         | 59          |
tb.inst1.inst2|a         | 59          |
tb.inst1.inst2|b         | 58          |
```

Limit on speedup

Amdahl's Law dictates the maximum speedup possible on multi-core simulations performance.

Amdahl's Law:

$$\frac{1}{(1 - P) + P/N}$$

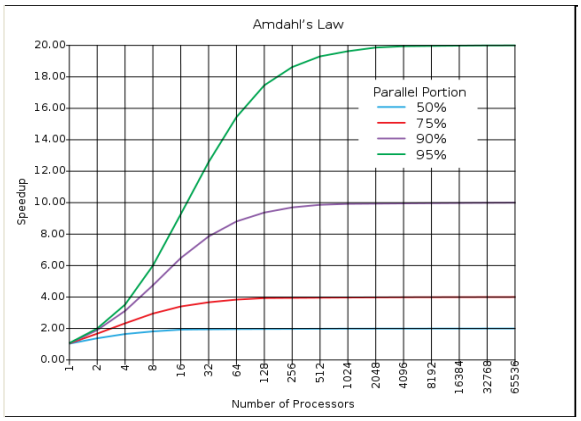

Figure 1: Amdahl's Law dictates the upper limit on speedups that can be achieved through multi-core simulations depending on the fraction of parallel portion

The speedup of a program using multiple processors in parallel computing is limited by the time needed for the sequential fraction of the program. For example, if a program needs 20 hours using a single processor core, and a particular portion of 1 hour cannot be parallelized, while the remaining promising portion of 19 hours (95%) can be parallelized, then regardless of how many processors we devote to a parallelized execution of this program, the minimum execution time cannot be less than that critical 1 hour. Hence the speed up is limited up to 20x.

That means even if a user is able to get ideal balance of load, concurrency and communication; ultimate speed up that can be obtained through multi-core simulation is constrained by Amdahl's law, and will vary based on design's characteristics.

Results

As enumerated in the previous sections of this paper, the performance gain in simulation using parallel partitions is highly dependent on the balance of the load of the design during simulation, concurrency of the partitions and the overhead of the communication to keep the partitions in sync. Below is a set of results from multiple customers and designs that demonstrate this variability.

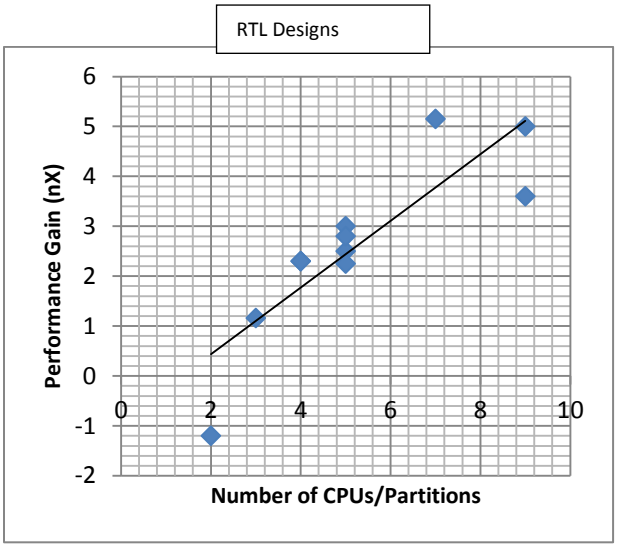

Figure 2: Results of RTL customer designs charting various performance gains vs. number of parallel partitions used

Figure 2 shows data points of 11 RTL designs of various combinations of VHDL and Verilog where the original simulation times ranged from 25 minutes to 8.5 hours. Using parallel simulation, the performance gain ranged from 1.2x slower to 5.15x faster. It is interesting to note that the design that was 1.2x slower was a test design that was primarily made up of a SystemVerilog testbench that was complex and a "fake" DUT that did not contain very much functionality. The profiling of the design showed that over 60% of the time in simulation was spent in the testbench. This heavy class based testbench which cannot be partitioned is why this design/test is not a good fit for parallel simulation. The trending curve is showing that on average, the

performance gain is approximately ½ the number of partitions/CPUs used.

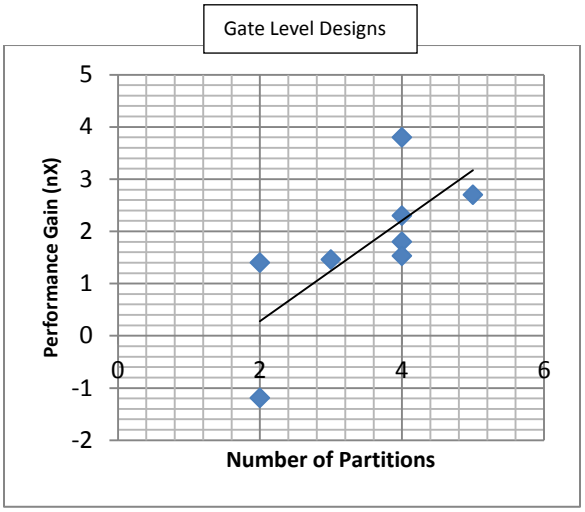

Figure 3: Results of Gate Level customer designs charting various performance gains vs number of parallel partitions used

Figure 3 shows data points of 8 Gate Level designs, primarily Verilog, where the original simulation times ranged from 17 minutes to 53 hours. Customers are finding that Gate Level simulations tend to be an easier fit for parallel simulations as the design activity is more balanced after synthesis. As seen in the RTL simulation results, there is a result in which the parallel simulation is slower than a normal simulation. In this case, the profile showed a very unbalanced simulation where over 80% of the simulation time was spent in a small block so it was not a good fit because of the balance. Again, on average, the performance gain trend is approximately ½ the number of CPU Cores.

Commercial Multi-Core technology

Multiple commercial simulation providers have been focused on exploiting multi-cores for improving simulation performance for many years now with the many major EDA vendors having products with these capabilities. The advances made recently are primarily around making this technology easier for users to deploy and reducing restrictions so that it is as simple as running their normal simulations. The goal is to see easy performance improvements, when used on designs that fit in with above described factors.

The most important feature for the users of these tools is the ability to have multiple ways to create the designs partitions – manual, semi-automatic or fully automatic. The partitioning mechanism must be very flexible such that it can be edited or modified by the users after it has been generated and it must naturally fit into the users' simulation use model and flow.

If a user has multiple languages in their simulation flow then it is also important that the multi-core implementation support all HDL languages or a mix of them, and various features such as coverage, assertions, SDF, etc.

Users also need to understand the tools' limitations across the partitions as that will have a direct effect on the balance of partitions and therefore the resulting performance.

Additionally, it is important to have not only a static automatic partitioning capability but have the ability to consider dynamic effects of simulation taken into account during partitioning in order to ensure the balance of runtime simulation activity across the partitions. This is especially important as the user will easily understand the natural static partitioning of the design hierarchy but will need help from the simulation tool in understanding where it is spending its time and how that can be balanced.

Once the balanced partitions are found, multi-core tools should also provide flexible methods to control the cross partition synchronization at different levels of granularity. This is important as the user may need to both tune the performance of the simulation and/or debug any design mismatches caused by race conditions across the partition boundaries.

In addition to automating partitioning and simulation, all of the commercially available multi-core technology tools provide various levels of analysis reports to help provide feedback to users on whether their design is suitable or not for multi-core simulation. It is important that this information be used to help users in qualifying suitable designs and making an early decision whether or not it will be

beneficial to spend time on running multi-core simulation. The same reports need to also be able to be used as an analysis tool to help understand multi-core post-simulation performance results. They need to be both detailed but also easy to read in order to provide feedback to the user to tune the partition file and help in obtaining balance factors that can improve multi-core simulation performance.

Applicability of parallel simulations technology

Parallel simulations may not be the best answer to some scenarios and it is very important for designers to understand where to use the technology and where not to. The main factors for achieving good performance speedups with parallel simulations in general also holds true for QuestaSim MC^2 simulations technology and the deployments have exposed design qualification criteria which can lead to successful multi-core simulations

- Big designs with long simulation times of more than an hour are a good fit.
- Designs that can be partitioned well with balanced activity in each partition can provide good multi-core simulation speedups.
- Flat gate-level netlists and designs with very low to no hierarchies are not good candidates for multi-core simulations. Such designs do not partition well and generate excessive amount of communication at lower level.
- Caution should be taken with designs that contain a lot of PLI/DPI/VPI/FLI usage. In some scenarios, design may not partition well due to access to cross-partition contents. Individual partitions should be thought of as mostly 'local' simulations with no global access.
- The design should be race-free as much as possible since multi-core simulations may expose races, due to reordering of events, and that may result in simulation mismatches against single-core simulations.

Overnight regressions

If the overnight regression suite consists of a large number of tests with small simulation time, it is not recommended to use multi-core simulations to achieve speedup. Short tests may result in worse performance with multi-core due to synchronization overhead.

In such scenario, total throughput of an overnight regression run can be improved by submitting multiple simulation jobs in parallel to a distributed grid.

Weeklong runs

A single test that takes multiple hours/days to complete can be an ideal candidate for multi-core simulations, provided it also adheres to the other qualifying criteria of a balanced load, communication and design concurrency. When design runs are too long, it becomes important to get results as early as possible, so as to find functional issues, fix them and reiterate sooner to reduce overall design cycle and increase productivity. Take an example of a design or test that takes 6 days to finish and get results. If you could even run it 2x faster, and get results in 3 days, that would be very productive.

It is also possible to have a mix of multi-core and single-core simulations in a regression suite depending on the length of the test and the performance speed up produced by multi-core simulation, which can result in optimal throughput.

Conclusion

Parallel simulations may not be an easy silver bullet for performance, but it does show dramatic performance results on suited designs. It also shows worst results when run on badly suited designs for parallel simulation. Important part is to qualify the designs which can work well with multi-core simulation, or understand the factors which affect multi-core simulation and consider them during design planning.

With improvements in QuestaSim's MC^2 technology, it has become very easy for users to setup and run

multi-core simulation and see results. QuestaSim also provides various analysis tools for users to help in qualifying good designs and analyzing various design factors to improve performance further.

References

[1] Parallel Logic Simulation: Myth or Reality?, Kai-Hui Chang and Chris Browy, IEEE Computer Society, April 2012

[2] Results from QuestaSim MC2 evaluations /deployments on real customer designs.

[3] Results from QuestaSim MC2 analysis tool reports from actual evaluations/deployments.