Verification with multi-core parallel simulations: Have you found your sweet spot yet?

Rohit K Jain, Mentor Graphics (510-354-7407, rohit_jain@mentor.com)
Shobana Sudhakar, Mentor Graphics (503-685-1889, shobana_sudhakar@mentor.com)

I. INTRODUCTION

Despite the fact that parallel simulation technology is supported by almost all industry-leading simulator tools and the number of cores available in the machine servers is trending upward, the idea of multi-core simulations has not caught on in the user community. Successful multi-core parallel simulations depend on a variety of related design factors, which can be difficult to understand and sort out. Widespread analysis of different design scenarios have raised interesting observations about users' perception and understanding of the technology, and also have helped make clear where it makes the most sense to use this technology and where it doesn't. With suitable design applications, it is possible to significantly save verification cycles. On the other hand, frustrating results can arise if multi-core simulation technology is used in unsuitable scenarios. The key to success here, and the topic this paper explores, is knowing where to use the multi-core technology.

The multi-core parallel simulation and results described in this paper pertains to design-level partitioning, which is achieved by partitioning the design to be simulated on multiple cores. Application-level partitioning is not discussed in this paper.

The authors previously described [1] the factors which impact a design's suitability for multi-core simulations, and how the user can modify/build their design with these factors in mind to make their design suitable for multi-core simulations.

This paper focuses on explaining the scenarios and flows that are best suited for multi-core technology. We will also explain what design types and flows should be avoided with multi-core.

Section 2 describes design types that are natural candidates for an efficient multi-core simulation flow, while Section 3 describes the kinds of designs that are not suitable. Further, Section 4 describes suitable application flows for multi-core simulations and Section 5 describes the flows where multi-core cannot be used efficiently. All the sections are supplemented by data from real-world customer designs, gathered using Questasim MC2 technology.

This paper is aimed at verification engineers looking to improve the productivity of their verification flow and to understand where multi-core simulations can provide maximum benefit.

II. SUITABLE DESIGN TYPES

Certain kinds of designs are natural candidates for an efficient multi-core parallel simulation due to the structure of the design and its inherent parallel simulation activity, and thereby meeting the criteria of balanced partition load, high degree of concurrent simulation activity, and low inter-partition communication [1].

The authors' previous work [1] describes design qualification criteria derived from the factors for achieving good performance speedups with multi-core simulations and validated with QuestaSim MC2 flow. Typically, designs that meet the qualification criteria show successful results with multi-core simulations.

While it is still required to qualify such designs to ensure they meet the multi-core parallel simulation criteria, they are generally more likely to meet the requirements and hence prove to be suitable designs for multi-core simulations.

Let's look at different design scenarios and the results that were extracted from real customer designs. All the results shown in this paper are collected using QuestaSim MC2 technology.

a. DFT (Design For Test) Simulations

Multi-core technology can be applied on multiple DFT (Design For Test) simulations with successful results and significant simulation time speedup. Design candidates for this vary from full SoC to sub-blocks verification; they are gate-level synthesized netlists with test logic inserted with different test methods, and with or without timing annotation. Runtimes of these designs are usually multi-hours long.

When applied on designs with serial load or MBIST DFT simulations, multi-core simulation provides almost close to linear speed up with respect to number of parallel host system cores used. Table-2a shows the results from a few such designs. Using four-six cores on various designs, the speedup over single-core simulation ranges from 1.8x to 4.65x.

Serial load scan chains are a common factor among these designs and the reason behind the speedup with multi-core simulation. This is because irrespective of the nature and functionality of the design, serial load scan chain produces parallel, balanced and distributed activity throughout the design.

Table-2a: Multi-core simulation data from various customer DFT designs

	Design type	Single-core sim runtime	Number of cores	Speedup with Questa MC2
Design1	SoC with MBIST test	1 hour	5	4.65x
Design2	SoC with serial scan test	12 hours	6	3x
Design3	Network SoC design with DFT	16 hours	4	3.8x
Design4	SoC sub-block GLS with DFT	4 hours	4	2.5x
Design5	Serial scan ATPG test (with SDF timing)	2 hours	4	1.8x

b. FPGA and FPGA Fabric designs

FPGA designs are another good multi-core simulation candidate. Designs used for analysis are either Verilog or VHDL, and are SoC, sub-blocks or GLS. Most of the designs have long simulation runtimes. These designs have very nice, clean and balanced partitions for multi-core, which is a major factor in successfully using multi-core simulation.

Table-2b shows the data collected from couple of such designs; Four-five parallel cores were used and the speedup ranged from 1.9x to 4.5x. Results show that FPGA-based designs are often good candidates for multi-core simulations.

Table-2b: Multi-core simulation data from various real-world FPGA and Fabric designs

	Design type	Single-core sim runtime	Number of cores	Speedup with Questa MC2
Design6	FPGA Fabric GLS	-	4	4.5x
Design7	TDD LTE FPGA design	7 hours	4	2.3x
Design8	FPGA VHDL	30 minutes	4	1.9x
Design9	SoC-FPGA Verilog	28 hours	5	3x
Design10	FPGA chip design	10 hours	4	3.6x

c. *Multi-core Processor, SoC or Server designs*

The increased complexity of systems-on-chip (SoC) designs with multiple active processors/cores/blocks has warranted exploiting all available processor cores in host systems in order to increase simulation speed without sacrificing accuracy. Multiple single-core simulations are usually used instead of (or in addition to) scalable simulation techniques such as abstract simulations at a higher level that loses cycle accuracy.

One of the major disadvantages of simulating a multi-core system is the decrease in simulation speed with increase in number of processing elements (or cores) to simulate, due to the increased tax on the simulation host system's shared resources, caches and memory interface. This decrease can be somewhat mitigated by taking advantage of the parallelism of the host system, and splitting the SoC design so that cores that are active in parallel are simulated on parallel host system CPUs.

It is important to understand the underlying parallel nature of the simulation. A design with inherent parallelism is characterized by the complex functions that are broken down into a series of small tasks that can be performance-independent of each other. This concurrency helps achieve major simulation time improvements with multi-core simulations, where active cores are simulated in parallel in separate host system CPUs with minimal communication between them. [1]

Table-2c shows the data from multi-core processor-based designs. These designs are varied, including Verilog or VHDL, RTL or GLS, with and without SDF annotation. Using three-five parallel cores, these designs show a 1.3x to 3x speedup over single-core simulation runtimes. What makes these designs suitable is the presence of multiple processor cores, which causes very good partitioning for multi-core simulations. With tests that activate and test all the processor cores, the design generates good parallel simulation activity.

One of the designs analyzed here, a high bandwidth memory design with a DRAM cell matrix, is split into three partitions due to the test exercising only two of the eight cell matrices. A 1.3x speedup is achieved with three cores using Questa MC2 technology. Tests exercising more DRAM cell matrices could potentially see higher speedup factors.

Table-2c: Multi-core simulation data from real-world multi-processor/core designs

	Design type	Single-core sim runtime	Number of cores	Speedup with Questa MC2
Design11	Multi-core processor GLS (with SDF timing)	53 hours	4	2.3x
Design12	VHDL RTL SoC	1 hour to 4.5 days	4	2.4x
Design13	Microprocessor, Verilog RTL	1 hour	5	3x
Design14	High-bandwidth memory with DRAM cell matrix; stimuli in VECTOR file format	5 hours	3	1.3x
Design15	ASIC Server design	-	4	2.5x

d. *Image, Video or Graphics processing designs*

The earlier section shows that multi-core processor or SoC designs are suitable candidates for multi-core simulation. Image-processing and video-processing designs share there are suitable for multi-core simulations for much the same reason: they often have multiple parallel processing cores. Thus, image, graphics or video processing designs can also be a good candidates for multi-core simulations.

Such designs have a tendency to generate parallel simulation activities in multiple different blocks, often due to pipelining, multiple processing cores or multiple frame processing within the design. Table-2d shows the results

collected from image/graphics/video processing designs, which also have very long simulation times. Using four to six parallel cores produce a speedup of 2x to 3x.

One of the designs analyzed for this section was set up to use multi-core simulation technology with signal logging for debug and got similar speedup as without logging. However multi-core reduced the latency of the debug run and proved very useful for debug cycles. We will discuss more about it in Section IV(b).

We also show one design example, which does not produce very attractive gains (1.2x with 4 cores). This was a small design (with simulation times less than an hour) and does not have factors (parallel activity and balanced partition) suitable for multicore. This example is included to show that although graphics-processing designs often make for good candidates, it's impossible to generalize when it comes to these designs or indeed any others. Initial work to qualify designs for multi-core suitability must always be done [1].

Table-2d: Multi-core simulation data from real-world Image, Graphics and Video processing designs

	Design type	Single-core sim runtime	Number of cores	Speedup with Questa MC2
Design16	ASIC video processing design	6.25 hours	7	2.5x
Design17	Graphics design	7 hours	4	2.3x
Design18	Graphics design with debug logging	27 hours	4	2.7x
Design19	Video Processing design	< 1 hour	4	1.2x
Design20	Image processing design, Verilog RTL	2 hours	5	2x

III. NON-SUITABLE DESIGN TYPES

Now that we have discussed design scenarios suitable for multi-core simulations, let's take a look what types of designs should be avoided.

a. Gate-level designs with no design hierarchy

Although a variety of gate-level synthesized netlist designs (SoC or DFT designs) are good candidates for multi-core simulations, designs which have no structural hierarchy or a flat design hierarchy are not good candidates for multi-core simulations. Such GLS designs usually contain huge netlist structures at the same level of hierarchy in, or flat just under, the testbench. This makes it difficult to partition the design for multi-core use. Even if the user is able to partition the design, the potentially high number of inter-partition connections and excessive inter-partition communication can offset any multi-core simulation. This also violates two of the required design factors for multi-core simulations: balanced load partition and low communication [1].

b. Designs with heavy hierarchical references across the design

Designs, which use a large number of hierarchical references between the testbench and DUT, and also within multiple hierarchy levels of DUT itself, do not make for suitable design candidates for multi-core simulation. Although such designs can be partitioned, the excessive number of hierarchical references across the design results in very heavy inter-partition communication. This communication overhead can offset any speedup achieved by using multi-core simulation, and in some cases (very heavy communication), can also result in negative speedup.

Table-3b shows one such example. This design is a DFT scan-chain design with SDF annotation. We have discussed earlier that DFT scan-chain designs are good candidates for multi-core simulations. However, in this case, the testbench accesses nets throughout the DUT using hierarchical references. Inter-partition communication was so high that using four partition cores caused a 14.5x slowdown compared to single-core simulation.

Another example is designs with heavy UVM backdoor access. These typically access high number of registers from the DUT in testbench, or vice-versa, and are not good multi-core simulation candidates.

Table-3b: Multi-core simulation data from customer designs with heavy inter-design hierarchical references

	Design type	Single-core sim runtime	Number of cores	Speedup with Questa MC2
Design21	Scan chain test with SDF	2.5 hours	4	-14.5x

c. *Parallel load DFT designs*

In section II(a), we discussed why DFT designs with serial scan chains are good candidates for multi-core simulations. On the contrary, DFT designs with parallel test load are not suitable candidates for multi-core simulations. The probable cause: insufficient continuously parallel and balanced activity in the different scan-chains. There is still some parallel activity in multiple partitions, so some gains can be achieved using multi-core simulations, but it may not be enough to justify using additional resources and additional tool licenses.

Table-3c shows results from such designs. A multi-media design running Memory BIST DFT simulations with parallel load showed only 1.24x speedup with four cores. Similarly, a large SoC GLS design partitioned into four cores gave only a 1.3x improvement in runtime due to insufficient balanced activity in the different scan chains.

Table-3c: Multi-core simulation data from real-world DFT designs with parallel load tests

	Design type	Single-core sim runtime	Number of cores	Speedup with Questa MC2
Design22	Multi-media MBIST design with parallel load DFT	51 hours	4	1.24x
Design23	GLS SoC DFT	10.5 hours	4	1.3x

d. *Designs with heavy serial simulation activity*

Serializer-Deserializer (SerDes) network designs and single core SoC with heavy sequential peripherals are also unsuitable. Both such design applications tend to have heavy serial simulation activity in the design. It was also observed that such designs tend to have heavy testbench activity. So, even though such designs can be partitioned and run with multi-core simulation technology, the unbalanced partitioning and the serial nature of simulation activity in the partitions prevent noticeable speedup using multi-core [1].

Table-3d shows results from such designs with heavy serial activity. Even if the runtime for Design24 is increased to run more frames or process more data, the characteristics of the design that cause unbalanced partitioning and heavy serial activity are still valid. This makes this kind of design unsuitable for multi-core simulation.
The data from Design25 in the table shows a negative speedup with multi-core simulations due to both unbalanced partitioning and heavy sequential activity.

Table-3d: Multi-core simulation data from customer designs with heavy serial simulation activity

	Design type	Single-core sim runtime	Number of cores	Speedup with Questa MC2
Design24	SerDes	< 10 minutes	3	1.0x
Design25	OVM based design	1 hour	2	-1.2x

e. *Designs with heavy encrypted IP/blocks*

Designs with heavy encrypted regions or IP blocks, where most of the simulation activity is contained to within the encrypted IP, are not good candidates for multi-core simulations. Such designs produce unbalanced partitions with at least one of the partitions being very heavy compared to other partitions, since the design cannot be partitioned through the encrypted block. In such designs, the entire encrypted block(s) are placed in a single partition and can cause unbalanced partition load and non-concurrent simulation activity.

Table-3e shows results from such designs.

Table-3e: Multi-core simulation data from real-world customer designs with encrypted IP blocks

	Design type	Single-core sim runtime	Number of cores	Speedup with Questa MC2
Design26	Design with encrypted 3rd party IP	80 minutes	2	1.13x
Design27	VHDL image processing design with encrypted ASIC library	3 hours	4	1.27x

IV. SUITABLE TEST/APPLICATION FLOWS

So far, we have discussed suitability of different design types for multi-core simulation. In this section we will discuss some of the test/application flows that can be used to qualify designs for multi-core simulation.

a. *Designs with long simulation runtime*

It is very common to find long simulation runtimes in functional verification scenarios such as late stage gate-level/post-synthesis simulations with timing annotation, netlist simulation with test logic insertion to perform device testing, FPGA scan-chain tests and system-level tests, etc. Multi-core simulations are ideal for such simulations that consume dozens of hours, a few days or even weeks for even a single simulation run.

In such scenarios, it is more valuable, important and efficient to get simulation results as soon as possible, as opposed to conforming to a 100% resource (core/machine) utilization. Multi-core simulations can dramatically reduce the latency of the results, and although multi-core simulations do not provide very good core utilization, they are still attractive and usually achieve results faster.

Questa MC2 technology was used to improve the runtime of a long-running ATPG simulation on a network SoC design, from nearly 16 hours to 4 hours, realizing close to 3.8x improvement using four cores. A Serial Scan DFT test on yet another SoC design improved from 12 hours on single-core to around 4 hours (3x speedup with six cores). Another case study is a graphics core design where Questa MC2 technology improved the simulation runtime from seven hours (no signal visibility) to only three hours using four cores (even with full design visibility), a 2.3x speedup.

Table-4a shows the results from these designs as well as additional customer designs with long simulation times. In each case, Questa MC2 generated results quickly and with several hours speedup.

Table-4a: Multi-core simulation data from real-world customer designs with long simulation runtimes

	Design type	Single-core sim runtime	Number of cores	Questa MC2 sim runtime	Speedup with Questa MC2
Design11	Multi-core processor GLS design (with SDF timing)	53 hours	4	23 hours	2.3x
Design17	Graphics design	7 hours	4	3 hours	2.3x
Design10	FPGA chip design	10 hours	4	< 3 hours	3.6x
Design3	Network SoC design with DFT	16 hours	4	~4 hours	3.8x
Design2	SoC with serial scan test	12 hours	6	4 hours	3x

b. *Long debug turn-around times*

It is well-known that verification engineers spend a lot of time in debugging tasks. Recent industry surveys indicate that ASIC/IC verification engineers spend 37% of their time in debug [2].

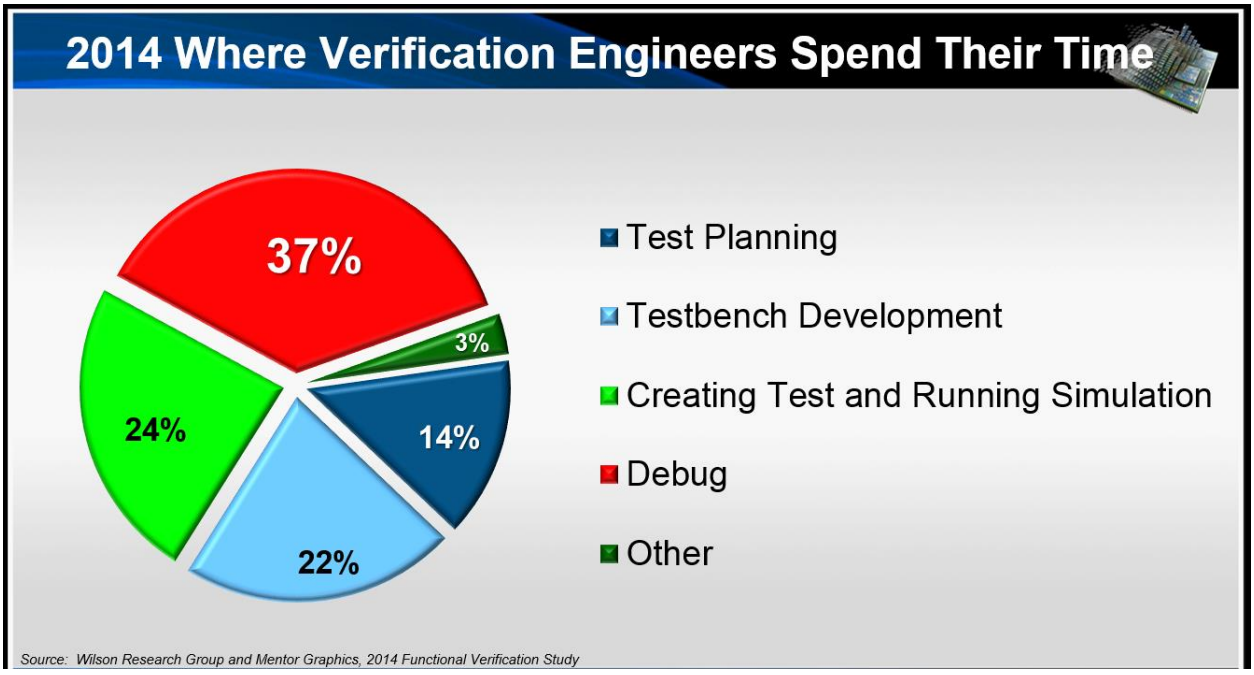

Figure 1- Mean time spent by ASIC/IC verification engineers in different tasks

While debug will never go away, technologies such as multi-core simulations can help shorten the debug cycle. If the design is a good candidate, a multi-core simulation allow the verification engineer doing the debugging to see massive savings in the time waiting for the simulation to get to a point where interesting things start to occur.

For example, if multi-core simulations can help reduce runtimes by 2X, a simulation that requires a runtime of 10 hours to generate waveforms, logs or other debug approaches, can be completed in 5 hours. Thus productivity could jump to two debug cycles in the same time it took for one debug cycle previously. Note that the savings are multiplied for as many debug cycles needed for a single design or test validation.

Questa MC2 technology was utilized in a graphics design, reducing the debug turnaround time from 27 hours to 10 hours, with full design visibility and signal logging. Another FPGA LTE design had a 2.3x speedup with four cores, with full design visibility and logging to Questasim's native WLF database. Table-4b lists the results from using Questa MC2 to reduce debug turnaround time on several customer design environments.

Table-4b: Multi-core simulation results from real-world customer designs with long debug turn-around cycles

	Design type	Single-core sim runtime	Number of cores	Questa MC2 sim runtime	Speedup with Questa MC2
Design18	Graphics design	27 hours	4	10 hours	2.7x
Design7	TDD LTE FPGA design	7 hours	4	3 hours	2.3x

QuestaSim MC2 technology for multi-core parallel simulations also supports logging design signals in Questa's native WLF format as well as in Verdi FSDB format. A native merge or a simple virtual merge can help combine the debug databases generated from each design partition into a single design-wide database for seamless design structure exploration and signal connectivity tracing.

c. *Large designs with huge memory footprint*

The growing size of today's SoC designs is stressing the available compute resources in terms of the server/machine memory (RAM). It can be quite expensive to upgrade the memory of systems or buy new, large machines to be able to handle such large designs. Instead, it has been observed that verification engineers typically stub out certain blocks in the design for everyday tests and regressions, keeping very few late-stage full-SoC tests to be run in simulators or in highly-expensive emulators.

Machines with very high memory requirements are usually very expensive compared to machines with low memory requirements. As design sizes increase, it puts more pressure on design teams to make high memory machines available to be able to do the verification.

QuestaSim MC2 technology supports partitioning a design to run a simulation on multiple computers, in addition to multiple cores on the same machine. Instead of using a very expensive high memory machine, a network of cheaper machines with smaller memories can be used to run the single simulation with the design partitioned across the different host machines and running in parallel.

There can be an overhead due to the communication between the machines and the network latency. However, if the design is a suitable candidate for this kind of simulation, the speedup seen from the parallel simulations can mitigate and even overcome any overhead slack. Multi-machine parallel simulation is best suited for scenarios where having the ability to run a large design using existing inexpensive resources is more important than running the design faster as a single-core simulation.

V. NON-SUITABLE TEST/APPLICATION FLOWS

This section describes the various reasons why certain simulations are not suitable for multi-core simulation flow and reiterates the design qualification criteria for an efficient multi-core flow [1].

a. *Short/bulk tests, Regression throughput*

Designs/tests with very short runtime or regression suites composed of multiple short-duration tests are not suitable for multi-core simulations. Latency is not an issue for such scenarios since the multitude of machines in a grid can solve latency problems. Typically, multi-core simulations do not produce a linear speedup factor compared to the number of cores used. So, even if multi-core simulation provides a good amount of speedup, it is still not attractive due to low resource utilization.

Superior resource utilization and regression throughput can be achieved using multiple single-core simulation jobs on multiple machines or multiple grid slots on same machine.

As an example, Questa MC2 technology gave only a cumulative 1.2x speedup across a sanity regression suite of 15 tests (of a real-world video processing design) – most of them short simulations – and the total throughput gains did not outweigh the multi-core communication overhead between partitions.

Table-5a: Multi-core simulation runtime from a regression suite of short tests

	Design type	Single-core runtimes of individual tests	Number of cores	Speedup with Questa MC2
Design19	Video Processor	7.5 – 21 minutes	4	1.2x

b. *Long simulations with unbalanced and non-concurrent activity*

Earlier results show that designs with long running simulations are good candidates for multi-core simulations, but long runtimes are not the only qualifying criteria to be successful with multi-core simulations. It is still important to qualify that a design meets with other required factors (concurrency, load and communication) [1].

Table-5b shows data from designs which resulted in very poor to negative speedups. These designs have unbalanced partitioning or sequential activity, or sometimes both.

After a design has been qualified for long runtimes, other factors like balanced partition load, should be analyzed through initial partitioning results and dynamic simulation load across the various blocks of design. These factors are apparent from the simulation profile report. Questa MC2 can do automatic partitioning of any given design. Questa MC2 also has useful utilities that can estimate the "rank" of a partitioning result, which can be used as a deciding factor on the best number into which a design can be split, even without running the simulation.

Additional knowledge from the design team can be used to augment the analysis for potential parallel and concurrent activity in the design. Only if a design meets all these initial qualification criteria should you proceed to apply multi-core simulations.

Table-5b: Multi-core simulation data from real-world designs with unbalanced simulation activity

	Design type	Single-core sim runtime	Number of cores	Speedup with Questa MC2
Design25	OVM based design	1 hour	2	-1.2x
Design28	Networking design	77 minutes	4	1.2x
Design29	Video design	2.5 hours	4	-1.8x

VI. CONCLUSION

We have discussed various design types such as DFT designs, FPGA designs, multi-processor designs and image/video processing designs as good candidates for multi-core simulation. We have also shown the results collected from multiple designs for each of these design types to assert our observations on why we think such designs are suited for multi-core application.

We have also discussed design types which are not suitable for multi-core application, such as GLS designs without hierarchy, designs with excessive hierarchical references, parallel load DFT designs, designs with serial activity and designs with large encrypted blocks. Such designs produce very low to negative speedup with multi-core simulation with respect to single-core simulation.

We described a few different design flows — long simulation time, debug turn-around, and high memory footprint — that are good applications for multi-core flows. Using multi-core technology on designs with long simulation time or for long debug turnaround time can help reduce latency and shrink time spent during verification. Using Questa MC2 on designs with a large memory footprint can help complete the verification using existing low-cost machines and avoid spending money on expensive high-memory machines.

We also looked at design flows such as regression tests with short/bulk tests and long running simulations with unbalanced activity that are not suitable for multi-core. Improvements in the regression throughput cannot be achieved using multi-core technology. Better solutions such as running single-core simulations on multiple grid machines with multiple cores can be used to improve regression throughput.

Suitable design types and flows for the multi-core simulation are not limited to the ones described in this paper. There are many other situations beyond those we have described here where multi-core simulations can prove useful. However the fairly standard scenarios and flows in this paper have been validated with real-life customer designs and thus may help guide users with similar scenarios applying multi-core technology more efficiently.

VII. REFERENCES

[1] The Need for Speed: Understanding design factors that make multi-core parallel simulations efficient, Shobana Sudhakar and Rohit K Jain, DVCON 2013

[2] Wilson Survey Group and Mentor Graphics 2014 Functional Verification study