United States District Court
For the Northern District of California

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN JOSE DIVISION

DYNETIX DESIGN SOLUTIONS INC., a California corporation, Plaintiff/Counter-defendant, v. SYNOPSYS INC., a Delaware corporation, Defendants/Counter-claimant.	Case No.: C 11-5973 PSG **FINAL CLAIM CONSTRUCTION ORDER**

In this patent infringement suit, Plaintiff and counter-defendant Dynetix Design Solutions, Inc. ("Dynetix") alleges that Defendant and counter-claimant Synopsys, Inc. ("Synopsys") infringes U.S. Patent No. 6,466,898 ("the '898 patent"). Synopsys counterclaims that Dynetix infringes U.S. Patent No. 5,706,473 ("the '473 patent") and U.S. Patent No. 5,784,593 ("the '593 patent"). The parties submitted eleven terms of the '898 patent for construction and stipulated to construction of one term. Regarding the '473 patent, the parties only asked the court to construe one term. In two separate claim construction hearings on October 10, 2012 and January 9, 2013, the court construed these terms from the bench and later summarized those rulings in an order.[1] The court provides its full reasoning behind those rulings below.

I. BACKGROUND

The '898 patent, entitled "Multithreaded, Mixed Hardware Description Languages Logic Simulation on Engineering Workstations," was filed on October 15, 2002. The '898 patent concerns the field of logic simulation of integrated circuits. Design of an integrated circuit begins with designers drafting the layout of the circuit in hardware description language ("HDL"). Designers can then use logic simulators, or electronic design automation ("EDA") tools,[2] to "verify the functional behavior and timing characteristics of their circuit designs on their engineering workstations before committing such designs to fabrication."[3]

The '898 patent discloses and claims three features that are relevant to claim construction: (1) mixed HDL simulation, (2) multithreaded logic simulation to achieve linear to super-linear speedup in performance, and (3) remote logic simulation. The first feature, mixed HDL simulation, is a method for simulating circuit designs written in multiple HDL languages (i.e. Verilog and VHDL) in a single program.[4] The second feature, multithreaded simulation, is a method that "enables the logic simulator provided by the invention to achieve a scalable performance (i.e., from linear to super-linear) according to the number of CPUs on the selected platform."[5] The third feature, remote logic simulation, "provides seamless access of network resources for HDL compilation and simulation" by installing a "server program" on a remote

[1] *See* Docket No. 559.

[2] *See* '898 patent, col. 1, ll. 1-3.

[3] *Id.* at col. 1, ll. 14-16.

[4] *See id.* at col. 4, ll. 13-15.

[5] *Id.* at col. 4, ll. 9-12.

workstation and a "graphical user interface (GUI) program" on a local workstation, which allows the user at the local workstation to communicate to remote servers.[6]

The '473 patent, filed on January 6, 1998, is entitled "Computer Model of a Finite State Machine Having Inputs, Outputs, Delayed Inputs, and Delayed Outputs."[7] It also relates to the field of circuit design and logic simulation and discloses and claims a "computer system having a computer model of a Finite State Machine (FSM)."[8]

II. LEGAL STANDARDS

Eight years after the Federal Circuit's seminal *Phillips* decision,[9] the canons of claim construction are now well-known – if not perfectly understood – by parties and courts alike. "To construe a claim term, the trial court must determine the meaning of any disputed words from the perspective of one of ordinary skill in the pertinent art at the time of filing."[10] This requires a careful review of the intrinsic record, comprised of the claim terms, written description, and prosecution history of the patent.[11] While claim terms "are generally given their ordinary and customary meaning," the claims themselves and the context in which the terms appear "provide substantial guidance as to the meaning of particular claim terms."[12] Indeed, a patent's specification "is always highly relevant to the claim construction analysis."[13] Claims "must be read in view of

[6] *Id.* at col. 4, ll. 20-32.

[7] '473 patent, col. 1, ll. 7-12.

[8] *Id.* at 1.

[9] *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312 (Fed. Cir. 2005) (en banc).

[10] *Chamberlain Group, Inc. v. Lear Corp.*, 516 F.3d 1331, 1335 (Fed. Cir. 2008).

[11] *Id.*; *Phillips*, 415 F.3d at 1312 (internal citations omitted).

[12] *Phillips*, 415 F.3d at 1312-15.

[13] *Id.*

the specification, of which they are part."[14] Although the patent's prosecution history "lacks the clarity of the specification and thus is less useful for claim construction purposes," it "can often inform the meaning of the claim language by demonstrating how the inventor understood the invention and whether the inventor limited the invention in the course of prosecution, making the claim scope narrower than it would otherwise be."[15] The court also has the discretion to consider extrinsic evidence, including dictionaries, learned treatises, and testimony from experts and inventors.[16] Such evidence, however, is "less significant than the intrinsic record in determining the legally operative meaning of claim language."[17]

III. ANALYSIS

A. The '898 Patent

A. 1.	**TERM**	**CONSTRUCTION**
	"Multithreaded simulation" *Claims 1, 2, 5-7, 19, 20-22, 23, 36, 39, 44, 45, 48, 53*	A "thread" is a process flow in a program that runs on a central processing unit ("CPU"). "Multithreaded simulation" means a simulation of circuit functionalities by executing multiple process flows concurrently on multiple CPUs.

Regarding this claim term, the parties primarily disagree about the definition of the internal term "thread." Dynetix argues that "thread" should be understood as "a process flow in a program that runs on a central processing unit."[18] Dynetix argues this construction is correct because the

[14] *Markman v. Westview Instruments, Inc.*, 52 F.3d 967, 979 (Fed. Cir. 1995) (en banc), *aff'd,* 517 U.S. 370 (1996). *See also Ultimax Cement Mfg. Corp v. CTS Cement Mfg. Corp.*, 587 F. 3d 1339, 1347 (Fed. Cir. 2009).

[15] *Phillips*, 415 F.3d at 1317 (internal quotations omitted).

[16] *See id.*

[17] *Id.* (internal quotations omitted).

[18] At first, Dynetix proposed the language "central processing unit" while Synopsys proposed simply "processing unit." Dynetix argued the addition of the word "central" was necessary

'898 patent specification has specifically defined this term. As a general rule, a clear definition set forth in the specification will prevail over extrinsic evidence.[19] The specification of the '898 patent, which explicitly defines "thread" in the context of "multithreaded simulation":

> EDA tools that employ multiple CPUs on a single workstation to accelerate their performance are said to be multithreaded. Specifically, a thread is a process flow in a program that runs on a CPU. If a program can have multiple threads executing on multiple CPUs concurrently, then it[] is a multithreaded application[].[20]

Synopsys proposes that the construction include the phrase "a thread is different from a process." Synopsys argues that a "thread" is part of a "process," offering excerpts of the specification as support. For example, the description of the invention states that "[i]n general, the thread manipulation overheads increase exponentially as more threads are used in a process"[21] and "[t]o reduce the number of threads employed in a process upon with n CPUs are available on a platform, the simulator will allocate exactly n threads."[22] Synopsys argues that these references to both "thread" and "process" within the same sentence/idea demonstrates that the two are mutually exclusive.

The court finds Dynetix's arguments to be more persuasive. Dynetix's proposed construction is identical to the definition of "thread" specifically set forth in the '898 patent specification. Synopsys has not persuaded the court that its proposed addition of the phrase "a thread is different from a process" is necessary. Even if the '898 patent notes that a thread can be

because the specification uses the term "central processing unit," *see, e.g.*, '898 patent col. 2, ll. 41-42, and the embodied invention does not include simple data processors like those found in desk calculators and tabulating machines. In its opposition, Synopsys noted that it did not object to adding the word "central" to the construction. *See* Docket No. 98 at 8 n. 2. The court therefore incorporates Dynetix's definition of "central processing unit."

[19] *See 3M Innovative Props. Co. v. Avery Dennison Corp.*, 350 F.3d 1365, 1371 (Fed. Cir. 2003) ("a definition of a claim term in the specification will prevail over a term's ordinary meaning if the patentee has acted as his own lexicographer and clearly set forth a different definition").

[20] '898 patent, col. 2, ll. 39-44.

[21] *Id.* at col. 17, ll. 39-40.

[22] *Id.* at col. 17, ll. 51-54.

part of a larger process, the thread itself may still be a process flow. No intrinsic or extrinsic evidence precludes this possibility. The IEEE Standard Dictionary of Electrical and Electronics Terms ("IEEE Dictionary"), for example, defines "thread" as "a single sequential flow of control within a process."[23] This definition can be reconciled with the fact that a thread could be a process flow within a larger multi-part process. Synopsys' proposed construction therefore unduly limits the term "thread" by mandating that it can never be a process.

A. 2.	**TERM**	**CONSTRUCTION**
	"To achieve linear to super-linear scalable performance speedup/simulation" *Claims 1, 2, 5-7, 36, 39, 44, 45, 48, 53*	The terms "linear" and "super-linear" describe the speedup that a parallel simulation will achieve when performing hardware containing one or more processing units. A simulation is "linear" if the speedup that is achieved is equal to the number of available processing units. For example, a simulation that runs two times as fast on hardware containing two processing units is "linear." Similarly, if the simulation runs four times as fast on four processing units, it is again "linear." A simulation that has a speedup greater than the number of processing units is "super-linear." For example, if a process executed on two processing units runes three times as fast as the same simulation on one processing unit, it is "super-linear." "Scalable performance" means there is a consistent increase in performance for each added processing unit.

Claim 1 employs this term in the following context:

1. …(c) automatically detecting the number of microprocessors (CPUs) available on the multiprocessor platform to create a master thread and a plurality of slave threads for concurrent execution of the multithreaded event-driven simulation of the design **to achieve linear to super-linear scalable performance speedup** as according to the number of CPUs on the multiprocessor platform.

[23] Docket No. 99, Ex. E at 11.

Dynetix urges the court to construe this term as "for the purpose of enhancing the simulation speed in a rate proportional to, or even faster than, the increase in the number of processors."[24] Dynetix argues that this is merely a "desired effect" or "purpose" of the claimed invention, but should not be read as a required element. This is proper because the specification makes clear that a multithreaded tool need only "consistently demonstrate[] linear to super-linear scalability on most test cases it processes" to be "classified as a linear/super-linear scalable tool."[25]

Synopsys disagrees, arguing that Dynetix ignores other portions of the specification requiring the invention to achieve "linear to super-linear scalability."[26] More fundamentally, during prosecution the patentee disclaimed simulators that do not achieve "linear to super-linear… speedup" by relying on that amendment to distinguish prior art. In response to an office action rejecting all of the original claims as anticipated by the prior art, the patentee added claim 1 which included the phrase "to achieve linear to super-linear scalable performance speedup" within the claim limitations.[27] In his response, the patentee argued that the various prior art references (Dunenloup, Bahra, Rompaey, Liao, and Davis) did not teach this particular limitation:

> Nor does [Dunenloup] make use of any multiprocessor platform to accelerate the performance of its system…
> Bahra does not teach concurrent use of any multiprocessor to speedup the performance of their system…
> There is no… teaching [in Rompaey] of making concurrent use of a multiprocessor platform to speedup their system performance…
> Davis system does not make use [of] any multiprocessor platform to speedup their system performance…

[24] Docket No. 83 at 8.

[25] '898 patent, col. 16, ll. 57-60.

[26] *See, e.g., id.* at 1 ("[t]his invention…uses special concurrent algorithms to accelerate the tool's performance on multiprocessor platforms to achieve linear to super-linear scalability on multiprocessor systems"); col. 4 ll. 9-12 ("[t]he algorithm enables the logic simulator provided by the invention to achieve a scalable performance (i.e. from linear to super-linear"); col. 17 ll. 17-19 ("[t]he simulator achieves linear and super-linear scalability by using special new concurrent methods that are described in the following sections").

[27] Docket No. 99, Ex. B at 34 (original claim 1 with no limiting language), Ex. D at 2 (amended claim 1 including the limiting language).

> Liao does not make use of any multiprocessor platform to speedup the execution of their system…[28]

Later in the response, the patentee further distinguishes the five prior art references mentioned above on the basis that they do not perform linear to super-linear scalable concurrent simulation:

> Dunenloup does not specify any means to perform super-linear scalable concurrent simulation of the system function.
> However, Bahra et al. does not teach any means to perform super-linear scalable concurrent simulation of the system function.
> Rompary does not perform super-linear scalable concurrent simulation function.
> Davis, however, does not perform any super-linear scalable concurrent simulation.
> Liao does not perform any super-linear scalable concurrent simulation.[29]

The court agrees with Synopsys that this term is a limitation, not merely a hoped-for result. It is well-established that a patentee's argument "that a prior art reference is distinguishable on a particular ground can serve as a disclaimer of claim scope."[30] Although clauses describing a particular result are sometimes viewed as nonlimiting, where clauses are material to patentability, courts have not hesitated to interpret them as required limitations.[31] Here, the patentee clearly distinguished prior art references on the basis that they do not achieve a "speedup" of system performance and do not "perform super-linear scalable concurrent simulation." These disclaimers show that the amendments to claim 1 were material to patentability, rendering the language at issue a required limitation.

[28] Docket No. 99, Ex. D at 18-19.

[29] *Id.* at 36-37.

[30] *American Piledriving Equipment, Inc. v. Geoquip, Inc.*, 637 F.3d 1324, 1336 (Fed. Cir. 2011).

[31] *See Hoffer v. Microsoft Corp.*, 405 F.3d 1326, 1330 (Fed. Cir. 2005) (holding that where the specification and prosecution history established that a "whereby" clause was "an integral part of the invention," the clause limited the invention); *C&C Jewelry Mfg., Inc. v. Trent West*, Case No. 09-1303 JF, 2010 WL 2681921 at *4-6 (N.D. Cal. July 6, 2010 (holding that the phrase "to provide a pleasing appearance" was a necessary limitation of the invention); *Fast Memory Erase, LLC v. Spansion, Inc.*, Case No. 3:08-cv-977, 2010 WL 363498, at *4 (N.D. Tex. Feb. 2, 2010) (construing the clause "whereby source leakage of the semiconductor device is reduced" to be a limiting term rather than merely an intended result because "the reduction of source leakage was material to patentability").

Having determined that the phrase limits claim 1, the court turns to the task of construing that term. Synopsys asserts that the term has been defined by the '898 patent specification:

> Specifically, when a user runs a multithreaded tool on a n CPU system (where n > 1), he would expect that the performance of the tool should improve by C * n times, where C is an empirical constant as follows:
>
> 0.5 > = C < = 1
>
> For example, if C is 0.75, then the expected speedup of a multithreaded tool on different configurations of a multiprocessor system are:
>
> | Number of CPUs | Speedup |
> |---|---|
> | 2 | 1.5 times |
> | 4 | 3.0 times |
> | 8 | 6.0 times |
>
> If the C factor of a multithreaded tool remains at 1.0 on different number of CPU configurations, the tool is said to demonstrate a linear scalability. If the C factor of the tool is less than 1.0, then it is said to demonstrate a sub-linear scalability. Finally, if the C factor of the tool is above 1.0, then it is said to demonstrate a super-linear scalability.[32]

In other words, if the ratio of speedup to the number of CPUs is equal to one, the tool exhibits linear scalability. If the ratio exceeds one, the tool demonstrates super-linear scalability. The phrase "linear to super-linear" scalable performance therefore requires the tool to exhibit at least a one-to-one ratio of speedup per added processing unit. Applying this concept to concrete examples, if a simulation runs twice as fast on a multithread tool using two CPUs as a tool using one CPU, the tool is linear. If that same tool instead employed four CPUs, it would run the simulation four times as fast. By contrast, if a multithread tool employing two CPUs instead of one ran a simulation three times as fast, the ratio of speedup to CPUs would be 1.5, which is greater than one and thus super-linear.

[32] '898 patent, col. 16, ll. 26-54.

Dynetix admits that "[i]t is true that Synopsys' definition uses the literal language[] found in the specification," but maintains that this "mathematical precision" is unnecessary.[33] Dynetix instead urges the court to adopt the generally-understood meaning of the word "linear," which according to the Merriam-Webster Online Dictionary can mean "resembling a straight line" or "having or being a response or output that is directly proportional to the input."[34]

The court finds Synopsys' proposed construction properly mirrors the specification. In defining not only "linear" but "super-linear" scalable performance, the patentee has acted as his own lexicographer, and that definition is far more precise than any outside information. Dynetix ignores the clear definition provided by the specification, instead advocating for extrinsic evidence to control the term's construction. The dictionary definition of "linear" provided by Dynetix conflicts with the specification's definition – a line can be "proportional" or can "resemble a straight line" but still have a slope of less than one, which under the specification's definition would be sub-linear. Further, although Dynetix complains that Synopsys' proposed construction is too complicated for a jury to understand,[35] the opposite is true. Synopsys applies the abstract concept to concrete examples in a way that the jury could comprehend.

The court modifies Synopsys' proposed construction of "scalable performance" to emphasize that it means there is a "*consistent* increase in performance for each added processing unit." Consistent increase in performance is emphasized by the table of test results excerpted above.[36] As the number of CPUs increases from two to four to eight, the speedup also increases by the same proportion each time. This demonstrates consistency in performance improvement.

[33] Docket No. 83 at 11.

[34] Docket No. 85, Ex. 6.

[35] *See O2 Micro Int'l Ltd. v. Beyond Innovation Tech. Co.*, 521 F.3d 1351, 1359 (Fed. Cir. 2008).

[36] *See* '898 patent, col. 16, ll. 26-54.

A. 3.	TERM	CONSTRUCTION
	"Achieving super-linear scalable simulation"	This term in the preamble limits claims 36 and 45.

This term, similar to the last, appears in the preambles of claims 36 and 45:

> 36. A method of **achieving super-linear scalable** Hardware Description Language simulation for a multithreaded event-driven simulation for a multithreaded event-driven simulation of an integrated circuit design on a multiprocessor platform, comprising steps of:
>
> 45. A program product of **achieving super-linear scalable** Hardware Description Language simulation for an event-driven logic simulation of a circuit design on a multiprocessor platform, comprising:

As this term is almost identical to the last, one might reasonably assume that construction of this term would closely track the previous construction. The main difference is that the previous term appears in the body of the claim, while this one appears in the preamble. But before adopting the same construction, the court must ask: does the preamble language limit the claim?

The law governing whether a preamble limits the claim is not always clear.[37] "Whether to treat a preamble term as a claim limitation is determined on the facts of each case in light of the claim as a whole and the invention described in the patent."[38] The general consensus is that the preamble may be construed as limiting if it "recites essential structure or steps, or if it is necessary to give life, meaning, and vitality to the claim."[39] Conversely, if the preamble merely "extoll[s] benefits or features of the claimed invention," it will not limit the claim scope.[40] The preamble

[37] *See Am. Med. Sys., Inc. v. Biolitec, Inc.*, 618 F.3d 1354, 1363 (Fed. Cir. 2010) (Dyk, J., dissenting) ("As the majority itself appears to recognize, we have not succeeded in articulating a clear and simple rule."); *Bell Commc'ns Research, Inc. v. Vitalink Commc'ns Corp.*, 55 F.3d 615, 620 (Fed. Cir. 1995) ("Much ink has, of course, been consumed in debates regarding when and to what extent claim preambles limit the scope of the claims in which they appear.").

[38] *Am. Med. Sys., Inc.*, 618 F.3d at 1358.

[39] *Id.*

[40] *Catalina Mktg. Int'l, Inc. v. Coolsavings.com, Inc.*, 289 F.3d 801, 808 (Fed. Cir. 2002).

also is not limiting if it is "duplicative" and simply "gives a descriptive name to the set of limitations in the body" such that deletion of the preamble would not affect the structure or steps of the invention.[41]

Dynetix claims that the phrase "to achieve" shows it is not essential to the invention but merely describes desired results. This contention runs contrary to the numerous references in the Abstract, Summary of the Invention, and other claims that make clear that the invention uses an algorithm to achieve linear to super-linear scalability.[42] Citing *Am. Med. Sys. Inc.*, Dynetix nevertheless persists that the preamble is not essential because removing the preamble language would not alter the structure or steps of the invention.[43] But unlike in *Am. Med. Sys. Inc.*, in claims 36 and 45 the preamble and the set of the limitations in the body do not appear to overlap – the rest of the claim does not address any kind of performance speedup. All this indicates that the preamble language describes an essential aspect of the claim that is not present in the bodies of claims 36 or 45.

Even if that were not enough, Synopsys presents evidence showing the preamble language was essential in distinguishing the prior art. "Clear reliance on the preamble during prosecution to distinguish the claimed invention from the prior art transforms the preamble into a claim limitation

[41] *Am. Med. Sys. Inc.*, 618 F.3d at 1358-59.

[42] *See* '898 patent at 1 ("This invention describes a multithread HDL logic simulator… it uses special concurrent algorithms to accelerate the tool's performance on multiprocessor platforms to achieve linear to super-linear scalability on multiprocessor systems"); *see id.* at col. 4 ll. 5-13 ("This invention describes a novel concurrent, multithreaded algorithm to accelerate the execution of logic simulation… The algorithm enables the logic simulator provided by the invention to achieve a scalable performance (i.e. from linear to super-linear) according to the number of CPUs on the selected platform").

[43] *See Am. Med. Sys., Inc.*, 618 F.3d at 1359 (holding that the preamble language "photoselective vaporization" was merely a "descriptive name for the invention [] fully set forth in the bodies of the claims").

because such reliance indicates use of the preamble to define, in part, the claimed invention."[44] Here, as noted previously, the examiner rejected all of the patentee's original claims (including the claims that eventually issued as claims 36 and 45) as anticipated by five prior art references.[45] The patentee then distinguished each of the prior art references on the grounds that the claimed invention performed "super-linear scalable concurrent simulation."[46] This means the preamble language was crucial to the claims' patentability and was "necessary to give life, meaning, and vitality to the" claims.[47]

In sum, despite the fact that the term appears in the claims' preambles, the court finds that the term "linear to super-linear scalability" limits claims 36 and 45.

A. 4.	**TERM**	**CONSTRUCTION**
	"Event queue" *'898 patent* *Claims 5, 36, 39, 44, 45, 48, 53*	An "event" in simulation is a task to be processed at a specified time resulting in a change of state. "Event queue" is a sequence of events held in temporary storage waiting to be processed.

The parties agree on the construction of the overarching term "event queue," but disagree about the internal definition of "event." As an example, this term appears in claim 36 in the following context:

> 36. …(b) minimizing thread interaction and synchronization by assigning a private heap memory, **event queue**, and fanout queue region for each of the master thread and slave threads at the beginning of the simulation to eliminate thread synchronization resulting from subsequent addition or deletion of signals or logic gate events during the simulation; and…

[44] *Catalina Mktg. Int'l, Inc.*, 289 F.3d at 808.

[45] *See* Docket No. 99, Ex. C.

[46] *See id.*, Ex. D.

[47] *Am. Med. Sys. Inc.*, 618 F.3d at 1358.

Dynetix asks the court to construe "event" broadly as "a task to be processed at a specified time and may be either a signal or logic event." To derive this interpretation, Dynetix points to both intrinsic and extrinsic evidence. The specification discloses an embodiment that "schedules events for these selected signals to change states according to the stimulus specification."[48] This demonstrates that an event is not coterminous with a change of state. Dynetix concludes an event can take form as either a signal or logic gate event because the specification describes an event queue as "[a] linked list [that] may contain both signal events and logic gate events."[49] Figure 21 also depicts an event queue consisting of different signal and gate events. The specification, however, never actually defines "event." To that end, Dynetix turns to an extrinsic source: the Merriam-Webster dictionary defines an event simply as an "occurrence."[50] Even though the Merriam-Webster dictionary definition makes no mention of "a task to be processed," Dynetix argues that the broad dictionary definition supports Dynetix's equally broad construction.

Synopsys complains that Dynetix's definition is overbroad. While words in a claim "are generally given their ordinary and customary meaning," that meaning must be construed in terms of what a person of ordinary skill in the art at the time of the invention would understand the term to be.[51] A technical term therefore must generally be construed "as having the meaning that it would be given by persons experienced in the field of the invention,"[52] which in this case would be the field of event-driven logic simulation. It is therefore inappropriate, says Synopsys, to adopt a broad definition of "event" as commonly understood outside of this specific context.

---

[48] '898 patent, col. 6, ll. 42-44.

[49] *Id.* at col. 6, ll. 46-48.

[50] Docket No. 86, Ex. 7.

[51] *Phillips*, 415 F.3d at 1303, 1312.

[52] *Hoechst Celanese Corp. v. BP Chemicals Ltd.*, 78 F.3d 1575, 1578 (Fed. Cir. 1996).

To be specific to the relevant field, Synopsys argues that the definition of "event" should incorporate "a change in state." The specification describes an event-driven simulation process:

> FIG. 3 illustrates the event-driven simulation process in which the input stimulus consists of new states to be assigned to the primary input and bi-directional signals (e.g. `11, 12, and B1 of FIGS. 2) of the design being simulated at various simulation time points. The simulator schedules events for these selected signals to change states according to the stimulus specification.[53]

This indicates that in the context of event-driven simulation and the claimed invention, the events cause signals to change states. The IEEE Dictionary definition further supports this claim. An "event" is defined as "an occurrence that causes a change of state in a simulation."

The court finds that both proposed constructions suffer from the same problem – including signal or logic events in the definition of "event" is redundant given the claims' later references in the same paragraph to signals or logic gate events.[54] It is unnecessary to define an event as including both signal and logic events because the claim itself later makes clear that both signals and logic gates may be events. Stripping away the redundant construction language, Dynetix and Synopsys' proposed constructions are more similar than not. They both agree that the specification is clear that the events are to be processed at a certain time.[55] Otherwise, while Synopsys is correct that an "event" *results* in a change of state, the event itself is not coterminous with a change in state. As Dynetix proposes, the event is a task to be processed that prompts the change in state. Accordingly, the court finds it appropriate to construe an "event" as "a task to be processed at a specified time resulting in a change of state."

[53] '898 patent, col. 6, ll. 38-44.

[54] *See, e.g., id.* at col. 28, ll. 42-43.

[55] *See id.* at col. 6, ll. 38-44 ("[t]he simulator schedules events for these selected signals to change states").

A. 5.	TERM	CONSTRUCTION
	"Common design database" *'898 patent* *Claims 1, 2, 5-7*	A "design database" is a database in which the simulation later compiles design files and stimulus files supplied by the user. The phrase "common design database" means a design database which contains various compiled design modules that may be coded in different design languages that are processed by the same multithreaded simulation engine.

Claim 1 requires "pre-examining each user-specified HDL source file and automatically invoking an appropriate HDL compiler to compile a design source file into a common design database."[56] The term also appears in a number of dependent claims.

Dynetix contends that the specification describes a "design database" as the formation of a design database: "The logic simulator compiles the design files and stimulus file supplied by the user into a database."[57] For the meaning of the modifier "common," Dynetix looks to another part of the specification:

> To accomplish the aforementioned purposes the simulator compiles VHDL and/or Verilog design files into a common database to which the event-driven and cycle-based logic simulation will be performed.[58]

This description means that design files written in different languages are pooled into the same database. Dynetix interprets "common" as this pooling of different design files in the same database, and nothing more. Dynetix argues that even though the phrase "common design database" is preceded by the article "a," that does not mean that there is only one such database in

[56] *Id.* at col. 23, ll. 13-16.

[57] *Id.* at col. 1, ll. 32-33.

[58] *Id.* at col. 10, ll. 22-25.

United States District Court
For the Northern District of California

the claimed invention; in a "comprising" claim, courts have interpreted "a" or "an" to mean "one or more" unless the patentee evinces a "clear intent" otherwise.[59]

Synopsys agrees that the "common design database" requires that the simulator compile design files into the same database, whether they are coded in Verilog or VHDL languages, but rejects Dynetix's contention that the phrase refers to "one or more" such databases. Synopsys argues not that the article "a" should be implied to mean one database only, which would be contrary to Federal Circuit case law. Instead, Synopsys argues that the word "common" means that there is only a single database that is simulated by the same simulation engine. In the specification, the patentee noted and criticized known products that use two different, interconnected simulation engines to simulate mixed language designs:

> A few EDA vendors provide a simulation backplane to interconnect a VHDL and a Verilog simulator, so that a user can simulate his VLSI design coded in both VHDL and Verilog. These products are not very popular as they are expensive (i.e., users need to purchase two separate simulators and the backplane) and inefficient in their performance.[60]

The specification goes on to tout the claimed invention's distinctions from those products:

> There is therefore an apparent need for a general-purpose, multithreaded logic simulator that supports both the VHDL and Verilog languages in a single program to perform both a[n] event-driven and a cycle-based logic simulation on a multiprocessor platform chosen by a user.[61]

Crucially, the specification goes on to explain that in the claimed invention, "[o]nce users' VHDL and/or Verilog designs have been compiled into the simulation database, they are being processed by the same multithreaded engine."[62]

---

[59] *See Baldwin Graphic Sys., Inc. v. Siebert, Inc.*, 512 F.3d 1338, 1342 (Fed. Cir. 2009) ("this court has repeatedly emphasized that an indefinite article 'a' or 'an' in patent parlance carries the meaning of 'one or more' in open-ended claims containing the transitional phrase 'comprising'").

[60] '898 patent, col. 2, ll. 57-63.

[61] *Id.* at col. 3, ll. 10-14.

[62] *Id.* at col. 10, ll. 22-41.

The court finds Dynetix's construction to be well-supported by the specification except for its assumption that the claim encompasses more than one "common design database." While Dynetix is correct that the article "a" does not limit the claim to one such database, the specification makes clear that "common" means one database that is processed by a single multithreaded engine. This is the advantage that was touted by the specification – that the claimed invention could process multiple languages with the same program. Even if that were not enough, the description of the invention in the specification is unequivocal that simulations are processed "on the same multithreaded engine."[63] As a result, it is appropriate to limit this claim to a single multithreaded engine.

A. 6.	**TERM**	**CONSTRUCTION**
	"To create a master thread and a plurality of slave threads" *Claims 1, 2, 5-7, 36, 39, 44, 45, 48, 53*	Creating one thread for each processor where the master thread is executed on one processor and each of the slave threads is executed on a separate remaining processor.

This term appears in the following contexts in claims 1, 36, and 45:

> 1. … (c) automatically detecting the number of microprocessors (CPUs) available on the multiprocessor platform **to create a master thread and a plurality of slave threads** for concurrent execution of the multithreaded event-driven simulation of the design to achieve linear to super-linear scalable performance speedup as according to the number of CPUs on the multiprocessor platform…
> 36. … (a) minimizing frequencies of thread creation and destruction by **creating a master thread and a plurality of slave threads**, based on the number of available CPUs on the multiprocessor platform, prior to the start of simulation…
> 45. … means to minimize frequencies of thread creation and destruction by **creating a master thread and a plurality of slave threads** based on the number of available CPUs on the multiprocessor platform, prior to the start of simulation …

[63] *Id.*

The parties stipulated to a term contained within this phrase: "master/slave thread" means that "in a multithreaded application, the thread that controls the execution of all other threads is the 'master thread;' the other threads that are controlled by the master thread are called 'slave threads.'"[64]

In light of the parties' stipulation to construction of "master/slave thread," Dynetix argues that there is no need for further construction. Alternatively, Dynetix proposes that the term be construed as "to create a master thread and two or more slave threads."

Synopsys disagrees, arguing for a more comprehensive construction reflecting the necessary one-to-one ratio between threads and CPUs. The "context in which a term is used in the asserted claim can be highly instructive."[65] Claims 1, 36, and 45 all note that the master thread and plurality of slave threads are created "as according to" or "based on" the "number of CPUs on the multiprocessor platform." This demonstrates that there is a required relationship between the number of CPUs on the multiprocessor platform and the number of threads.

The specification, Synopsys argues, further underscores the "one thread per CPU" principle in the claimed invention. The patentee first described common practice in the prior art as when there are n CPUs on a system, the simulator allocates n + 1 total threads:

> Contrary to the invention, it is common practice for the prior art simulators to allocate n + 1 threads (one master and n slave threads) on an n-CPU system. The master thread's main function is to manage the execution of the slave threads to perform simulation.

The patentee went on to distinguish the claimed invention from the prior art:

> In this invention, however, the master thread will spend only a minimum amount of time managing the slave threads, and it shares an equal amount of workload with the n-1 slave threads to carry out the simulation. The invention approach has the benefits for reducing the idle time the master thread needed to spend in waiting for the slave-threads to complete their tasks.

The Detailed Description of the Invention in the specification then states unequivocally that on an n-CPU system, the simulator allocates "exactly n threads":

[64] Docket No. 59 at 1.

[65] *Phillips*, 415 F.3d 1303, 1314.

> To reduce the number of threads employed in a process upon which n CPUs are available on a platform, the simulator will allocate exactly n threads (one master thread and n-1 slave-threads) at program start[] up. Each of these threads will be set as a real-time thread and be scheduled by the operating system directly to bind to a hardware CPU throughout the entire simulation run. No other threads will be created during the simulation run.[66]

Dynetix responds that the excerpt discussing the claimed invention vs. the prior art actually focuses on improvements of having the master thread spending only a minimum amount of time managing slave threads, but then sharing an equal amount of workload with the slave threads.[67] Dynetix also contends that "based on" does not necessarily mean there is a one-to-one ratio between CPUs and threads, but merely means that the number of CPUs is taken into consideration.

While Dynetix is correct that the claim language alone does not mandate a one-to-one ratio, claims must be read in light of the specification. Here, the specification in the Detailed Description of the Invention explains in no uncertain terms that on an n-CPU system, the simulator will allocate exactly n threads (one master thread and n-1 slave threads).[68] The specification also distinguished prior art which used n + 1 threads. If the one-to-one ratio described was of no import, there would be no reason for the specification to specifically draw this distinction.

A. 7.	**TERM**	**CONSTRUCTION**
	"Pre-examining each user-specified HDL source file" *Claims 1, 2, 5-7*	The simulator examining the content of each HDL source file to automatically detect its coded file language before compiling the source files.

The term can be found in claim 1:

> 1. … (a) pre-examining each user-specified HDL source file and automatically invoking an appropriate HDL compiler to compile a design source file into a **common design database**;

[66] '898 patent, col. 17, ll. 51-58.

[67] *Id.* at cols. 17-18, ll. 67-2.

[68] *See id.* at col. 17, ll. 51-56.

Synopsys believes the specification and claim language requires that the simulator perform the pre-examining feature described by the term. The specification states that "[w]hen the simulator compiles a user-specified HDL source file, it will pre-examine the file content to detect automatically the coded file language."[69] Synopsys therefore seeks to make clear that it is the simulator, not the user, which automatically examines the source file before it is compiled.

Dynetix does not believe the term contains any confusing jargon and therefore does not need to be construed. Even if it were to be construed, Dynetix argues that a simple construction of "examining each source file before compiling the source files" would suffice, without specifying who conducts the examination. Dynetix contends that because the claim language uses the word "automatically" to modify the second feature described by claim 1, or "invoking an appropriate HDL compiler to compile a design source file," the patentee clearly knew how to draft a claim to cover only an automatic process, but did not do so regarding the "pre-examining" feature. Dynetix further takes issue with the fact that Synopsys' definition limits examination to the "content" of the source file. Without citation, Dynetix argues that pre-examination could be "looking at the file extension to determine the type of HDL used, not necessarily reviewing the contents."[70] Lastly, Dynetix complains that Synopsys has unnecessarily "cherry-picked" the term – Synopsys asked to construe the term "pre-examining each user-specified [] source file," omitting the world "HDL," and has provided no legitimate reason for doing so.

Contrary to Dynetix's assertion, the term is less than clear and benefits from construction. It is clear from the claim context and specification that the simulator examines the file content to determine its coded file language, then uses a language-specific compiler to compile the design source file. Synopsys' proposed construction is appropriate to describe both the pre-examining

---

[69] *Id.* at col. 9, ll. 43-46.

[70] Docket No. 107 at 17.

process and when it takes place. As for Dynetix's accusation of cherry-picking, this is easily remedied by inserting "HDL" into Synopsys' proposed construction, as seen above.

A. 8.	**TERM**	**CONSTRUCTION**
	"Specify remote hosts" *Claims 19-23*	The user identifying remote computers by name.

This term appears in terms 19-23 ("the remote access claims"), of which claims 19 and 23 are independent, in the following contexts:

> 19. … installing and executing a graphical user interface program ("GUI") on the user's local host to **specify remote hosts** on which the HDL design compilation and simulation is to be performed …
> 23. … means to provide a graphical user interface program ("GUI") on the user's local host **to specify remote hosts** on which the HDL design compilation and simulation is to be performed …

Dynetix contends that the term can be understood by its plain and ordinary meaning, but if the court were to construe the term, it should be understood as "to specify the remote servers." There is no need for construction of the word "specify" because it simply means "to name or state explicitly or in detail."[71]

Synopsys on the other hand argues that it is necessary to note that the user identifies the remote computers by name. The specification supports this notion, stating that "[t]his UI program allows the users to specify a remote machine name, and a remote directory pathname, on which they desire their HDL compilation and/or simulation to be performed."[72] "Name" simply means the unique identification information that the user invokes to contact the remote server.[73]

[71] Docket No. 85, Ex. 6.

[72] '898 patent, col. 13, ll. 12-15.

[73] *See id.* at col. 13, ll. 61-66 ("When a user specifies a remote host name and a remote directory pathname… for compilation or simulation… the UI [] will contact the remote host RMI naming service to obtain a handle for the server by specifying the registered name of the server.").

The court agrees with Synopsys. Synopsys' construction finds support in the intrinsic evidence, and as Synopsys itself concedes "naming" can be the user inputting various identifying data. Even under the extrinsic dictionary definition put forth by Dynetix, the definition of "specify" includes "naming." Synopsys' proposed construction is consistent with that definition and does not unduly narrow claim scope.

A. 9.	**TERM**	**CONSTRUCTION**
	"Graphical user interface" or "GUI" *Claims 19, 23*	A computer user interface that allows interaction using graphical objects such as icons, images, and windows as opposed to merely a command line interface.

The term "graphical user interface," or "GUI," also appears in the remote access claims. The GUI is "on the user's local host" and is used "to specify remote hosts on which the HDL design compilation and simulation is to be performed."[74] It also "automatically activat[es] network connection… to send the user's commands from the user's local hosts to the remote hosts to be executed thereof."[75]

The parties agree that the graphical user interface allows the user to interact with the computer through an interface that displays icons, images, and other graphical objects. Where they differ is if a graphical user interface necessarily excludes a command line interface.

Synopsys maintains that a graphical user interface can never be a command line interface, and in fact, the two are opposites. The Dictionary of Computing defines the two interfaces as follows:

> **Graphical user interface (GUI):** An interface between a user and a computer system that makes use of input devices other than the keyboard and presentation techniques other than

[74] *Id.* at col. 26, ll. 8-11.

[75] *Id.* at col. 26, ll. 12-15.

> alphanumeric characters. Typical GUIS involve the use of *windows, *icons, *menus, and *pointing devices.[76]
> **Command-line interface (CLI):** An interactive system where user input is achieved through lines of text. The user learns these commands by consulting an online *help system or reference manual. Users familiar with the interface may use abbreviations or mnemonic commands to speed access and reduce the number of keystrokes required for a given command.[77]

Dynetix points out that the IEEE Dictionary of graphical user interface merely notes that it is "graphical in nature," and the "user *can* enter commands by using a mouse, icons and windows" but that this is by no means necessary.[78] Dynetix also argues that Figure 13 shows a text-based input system as a preferred embodiment.[79]

The court agrees with Synopsys that a graphical user interface cannot be a command-line interface. What these extrinsic definitions make clear is that the two are diametrically opposed. A graphical user interface allows the user to enter commands through graphical objects, whereas a command-line interface is text-based commands only. The intrinsic evidence is consistent with this finding. As Synopsys notes, Figure 13 in the specification shows a graphical user interface, not a command-line, because it has icons for "Add," "Delete," "OK," "Cancel," or "Help" which allow the user to execute commands. A command-line interface would not have those icons but would merely have a text-based input system for those commands. Accordingly, Synopsys' exclusion of command-line interface from the definition of graphical user interface is appropriate.

A. 10.	**TERM**	**CONSTRUCTION**
	"By at the beginning" *Claim 39*	The court finds this term may be construed by correcting the typographical error. The phrase shall be corrected to "by the beginning."

[76] *Oxford Dictionary of Computing* 215 (4th ed. 1996).

[77] *Id.* at 86.

[78] *IEEE Dictionary* 458 (6th ed. 1997).

[79] '898 patent, Fig. 13.

Claim 39 states:

> 39. The method of achieving super-linear scalable Hardware Description Language simulation according to claim 36 further comprises the step of scheduling the master thread and slave threads to bind to CPUs of the microprocessor by at the beginning of simulation to eliminate the time spent in scheduling threads for execution during subsequent simulation.

Both parties recognize that claim 39 contains a typographical error, but they dispute whether the court can correct that error. "It is well-settled law that [] a district court may correct an obvious error in a patent claim."[80] However, it is equally well-established that a district court may do so only if "(1) the correction is not subject to reasonable debate based on consideration of the claim language and the specification and (2) the prosecution history does not suggest a different interpretation of the claims."[81]

Synopsys asserts that the term cannot be corrected because it is subject to reasonable debate, and as a result is indefinite. Specifically, Synopsys identifies a reasonable dispute as to whether the step of scheduling the master and slave threads to bind to the CPUs should occur "by" the beginning of simulation or "at" the beginning of simulation. Other claims are of no help because some require binding "prior to the start of simulation" while others require this step "at the beginning of the simulation."[82] The specification, too, does not provide any direction as to whether the binding "by" or "at the start of simulation. It states that "[e]ach of these threads will be set as a real-time thread and be scheduled by the operating system directly to bind to a hardware CPU throughout the entire simulation run."[83]

[80] *CBT Flint Partners, LLC v. Return Path, Inc.*, 654 F.3d 1353, 1358 (Fed. Cir. 2011).

[81] *Id.*

[82] '898 patent, col. 28, ll. 35-36, 40.

[83] *Id.* at col. 17, 54-58.

Dynetix in turn asserts that it does not matter whether "by" or "at" is used because it would be obvious to a person skilled in the art that "the step of scheduling the master thread and the slave threads to bind to CPUs" occurs before actual simulation.[84] As the Federal Circuit noted, the relevant question is what a person skilled in the art would understand the phrase and claim scope to be.[85] If the court corrected the construction to "at the beginning," that would mean scheduling is a part of simulation but occurs "before the core steps of simulation."[86] If the court chose the phrase "by the beginning," that would mean that scheduling is not a part of simulation and occurs before it.[87] Dynetix calls for the court to adopt the former, citing the same specification as Synopsys at the excerpt, "the simulator will allocate exactly n threads… at program start[] up."[88]

After scrutinizing the claim language, the court agrees with Dynetix, but only in part. Synopsys fails to appreciate that it is the timing of "the step of *scheduling* the… threads to bind to CPUs," not merely the binding of threads to CPUs, which is at issue here. The binding of threads and CPUs plainly occurs "throughout the entire simulation run."[89] The scheduling of that binding process, however, occurs before simulation. The claim language unequivocally supports this – it states that the purpose of scheduling first is "to eliminate the time spent in scheduling threads for

[84] *See id.* at col. 17, ll. 51-57 (noting that n threads are allocated at program startup so that they will each be bound to a CPU throughout the entire simulation run).

[85] *CBT Flint Partners, LLC,* 654 F.3d at 1358 ("a person of skill in the art would find the claim to have the same scope and meaning under each of the three possible meanings that the court found reasonable").

[86] Docket No. 107 at 20.

[87] *See id.*

[88] *See* '898 patent, col. 17, ll. 52-54.

[89] *Id.* at col. 17, ll. 57.

execution during subsequent simulation."[90] The specification language quoted by both parties also supports that understanding.

While the court agrees that either interpretation would have the same claim scope as understood by a person skilled in the art, the court does not believe that "at the beginning" would be the more precise correction.[91] A jury member or other lay person who is not well-versed in the field might read "at the beginning" as possibly coinciding with the simulation, even though a person skilled in the art would not. No similar issue occurs by correcting the phrase to "by the beginning," which even to a lay person signifies that scheduling is completed by the time the simulator runs any simulation. Therefore, the court chooses to correct the claim to "by the beginning" to alleviate any potential misunderstanding.

A. 11.	**TERM**	**CONSTRUCTION**
	"Means to provide a graphical user interface program ('GUI') on the user's local hosts" *Claim 23*	No ruling or construction in light of the court's concern that this claim may be indefinite.

Claim 23 is a means-plus-function claim:

23. A program product of executing remote Hardware Description Language ("HDL") compilation and multi-threaded simulation of a circuit design employing a user's local and remote single-processor or multiprocessor hosts, comprising:
[…]
Means to provide a graphical user interface program ("GUI") on the user's local host to specify remote hosts on which the HDL design compilation and simulation is to be performed;
Means to automatically activate network connection by the GUI to the server program to send the user's commands from the user's local host to the remote hosts to be executed thereof…

[90] *Id.*

[91] The court originally agreed with Dynetix that the claim language should be corrected to "at the beginning." *See* Docket No. 559. The court has inherent authority to change its claim construction rulings, and it finds upon further reflection that adopting slightly different language would be more appropriate.

A means-plus-function claim must be construed as follows. First, the court identifies the function described by the claim.[92] Second, the court looks to the specification and identifies "the corresponding structure, material, or acts that perform that function."[93] "A structure in the specification qualifies as a corresponding structure if the specification or the prosecution history clearly links or associates that structure to the function recited in the claim."[94] "Even if the specification discloses a corresponding structure, the disclosure must be adequate" in showing what is meant by the claim language.[95]

Dynetix identifies the claimed function of the term at issue as "to provide a GUI on the user's local hosts in a remote simulation method of claim 19."[96] Dynetix argues that the structure is a "local host" that can run the user interface,[97] but does not identify a more specific structure that is described in the specification but not in the claim language. In Dynetix's view, however, someone skilled in the art would understand that the "local host" would have to be a "computer or equivalent machine."[98] It argues that the corresponding structure is therefore a "combination of computer monitors, graphic drivers, input devices such as keyboard, mouse, and the enabling software, allowing the user to receive, display, manipulate, and output information, graphics, and commands on the user's local hosts."

[92] *See HTC Corp. v. IPCom GmbH & Co., KG*, 667 F.3d 127, 1278 (Fed. Cir. 2012).

[93] *Id.*

[94] *Noah Sys., Inc. v. Intuit, Inc.*, 675 F.3d 1302, 1311 (Fed. Cir. 2012).

[95] *Id.* at 1311-12.

[96] Docket No. 83 at 24.

[97] '898 patent, col. 13, ll. 7-12 ("…the users need to install a server or a server program 62 provided by the invention on the respective machines networked with the Internet or Intranets. Once the server program 62 is installed, the users then run a UI (user interface) program 63 on their local hosts.").

[98] Docket No. 83 at 24.

Synopsys, finding this explanation insufficient, insists the claim is indefinite. Nothing in the specification explains the function of "providing a graphical user interface" as Dynetix has suggested. With no function adequately described, there is also no structure linked to that function. The "combination of computer monitors, graphic drivers, input devices such as keyboard and mouse and the enabling software, allowing the user to receive, display, manipulate, and output information, graphics, and commands on the user's local hosts" appears nowhere in the specification. As a result, there is no proper function or structure description in the specification and the claim is indefinite. Synopsys argues this was an attempt by the patentee to claim the method of claim 19 as an apparatus, but in purely functional terms and without any associated structure.[99] This the patentee cannot do.[100]

The court has serious concerns that this claim is indefinite because the specification contains no reference to the claimed function and any associated structure. Further, Dynetix has not identified any legitimate support for its contention that a person skilled in the art would necessarily interpret a "local host" to be a "computer or equivalent machine." Dynetix only cites the inventor, Terence Chan's bare assertion that "a person of ordinary skill in the art would surely understand [a 'local host'] to mean a computer or equivalent machine with some kind of input device such as a keyboard and/or mouse."[101] This is plainly insufficient; "[b]road conclusory statements offered by [one party's] experts are not evidence."[102] In ruling at the claim construction hearing, the court expressed that concern and invited Synopsys to bring a motion for summary judgment on indefiniteness of this claim. The parties did not do so, opting to take a different

[99] *See Med. Instrumentation & Diagnostics Corp. v. Elekta AB*, 344 F.3d 1205, 1211 (Fed. Cir. 2003).

[100] *See id.*

[101] Docket No. 84 ¶ 14.

[102] *Telemac Cellular Corp. v. Topp Telecom, Inc.*, 247 F.3d 1316, 1329 (Fed. Cir. 2001).

litigation route. As the parties never provided further briefing on this issue, the court does not decide this issue now.[103]

B. The '473 Patent

B. 1.	**TERM**	**CONSTRUCTION**
	"Finite state machine" *Claim 7*	A sequential circuit whose finite number of output values at a given instant depends on either the sequence of previous inputs, the current input, or both.

The term "finite state machine" is central to the '473 patent and appears in multiple claims, including independent claim 7:

> 7. A method of simulating a circuit on a computer system, said circuit including a computer model of a **Finite State Machine (FSM),** said computer system including a processor and memory, said processor being coupled to said memory, said method comprising:
> accessing a textual description of said computer model into a compiled computer model;
> compiling said textual description of said computer model into a compiled computer model;
> storing said compiled computer model in said memory;
> accessing said compiled computer model;
> accessing a first input value;
> accessing a first delayed input value;
> accessing a first delayed output value; and
> executing a simulation program on said processor causing said processor to access a first output value from said computer model using said first input value, said first delayed input value and said first delayed output value, said first output value corresponding to the output value of said circuit.[104]

Synopsys contends that the term is defined by the specification:

> Some sequential circuits, called Moore machines, have outputs that depend on a sequence of previous inputs. Other sequential circuits, called Mealy machines, have outputs that depend on the sequence of previous inputs and the current input. Each sequence of

[103] In any event, the court found that the remote access claims are no longer at issue for other reasons. *See* Docket No. 166 (denying Dynetix's request to amend infringement contentions to assert remote access claims against VCS Multicore on diligence grounds); Docket No. 362 at 12 (ruling that VCS Cloud, the only infringing device implicated by the remote access claims, does not infringe because it does not use a GUI).

[104] '473 patent, cols. 15-16, ll. 60-12.

> previous inputs causes the circuit to assume a specific state for the sequential circuit. Because sequential circuits have a finite number of these states, sequential circuits are referred to as finite state machines (FSMs).[105]

Factoring in this passage describing "Moore machines" and "Mealy machines," both of which could be finite state machines, Synopsys urges the court to construe "finite state machine" as "a circuit that has a finite number of output values at a given instant that are dependent on a sequence of previous inputs." Synopsys asserts that this proposed construction also is consistent with the claim language and the IEEE Dictionary, which defines "state" in the field of modeling and simulation as the "values assumed at a given instant by the variables that define the characteristics of a system, component, or simulation."[106] Flip-flops would meet this definition – they are defined by the IEEE Dictionary as "a circuit or device capable of assuming either of two stable states, and which can be made to switch states by applying the proper signal or combination of signals to its inputs."[107]

Dynetix, on the other hand, asks the court to interpret "finite state machine" as "a computational model of sequential circuits" with "outputs that depend on, among other things, the previous inputs." Dynetix further asks the court to exclude "simple devices such as flip-flops" from this definition. Looking at the same passage provided by Synopsys, Dynetix contends that Synopsys' definition is inaccurate because at least some sequential circuits (Mealy machines) have outputs depends on more than just previous inputs, but also current ones. Further, the specification shows that the model is intended to be a compilation of primitive devices. For example, a netlist computer model is described as "a number of interconnected primitive models" and each

[105] *Id.* at col. 1, ll. 17-26.

[106] Docket No. 133, Ex. C.

[107] *IEEE Dictionary* 443 (7th ed. 2000).

"primitive model" is described as "a model of a primitive digital device such as an AND gate, OR gate, or a D flip-flop, or a multiplexor."[108]

The court finds that the passage cited by both parties provides the definition for a finite state machine. What both parties miss in their briefing, however, is that both Moore machines and Mealy machines are included in the definition of finite state machines. Therefore, the construction of that term should include sequential circuits that depend on previous inputs, current inputs, or both.

As for Dynetix's proposed exclusion of flip flops from the definition of finite state machine, the court sees no reason to implement such an explicit exclusion. "Negative limitations should not be accepted [] absent clear disavowal, disclaimer or estoppel."[109] Although Dynetix cites parts of the specification describing computer model as being comprised of more "primitive models… such as… a D flip-flop,"[110] those excerpts are far from clear disavowals of claim scope because they merely cite examples, not requirements. Just because some examples of finite state machines are built from multiple components does not mean that all finite state machines must be so. Nothing in the claim language or specification limits a finite state machine to a more complex compilation of simpler components.

IT IS SO ORDERED.

Dated: September 11, 2013

[illegible]
PAUL S. GREWAL
United States Magistrate Judge

[108] '473 patent, col. 3, ll. 37-39.

[109] *Nuance Commc'ns, Inc. v. Abbyy Software House, Inc.*, Case No. C 08-02912 JSW, 2012 WL 1188903, at *4 (N.D. Cal. Apr. 9, 2012) (internal quotations omitted).

[110] '473 patent, col. 2, ll. 7-11.