



Planet Depos®
We Make It *Happen™*

Transcript of Massoud Pedram, Ph.D.

Date: January 5, 2024

Case: Realtek Semiconductor Corp. -v- ATI Technologies ULC (PTAB)

Planet Depos

Phone: 888.433.3767

Email: transcripts@planetdepos.com

www.planetdepos.com

Conducted on January 5, 2024

1 UNITED STATES PATENT AND TRADEMARK OFFICE 2 ----- 3 4 BEFORE THE PATENT TRIAL AND APPEAL BOARD 5 ----- 6 7 REALTEK SEMICONDUCTOR CORP., 8 Petitioner, 9 v. 10 ATI TECHNOLOGIES ULC, 11 Patent Owner. 12 ----- 13 14 Case No. IPR 2023-00687 15 U.S. Patent No. 8,854,381 16 17 18 DEPOSITION OF MASSOUD PEDRAM, PH.D. 19 VOLUME I 20 REMOTELY IN LOS ANGELES, CALIFORNIA 21 FRIDAY, JANUARY 5, 2024 22 23 24 REPORTED BY: NATALIE PARVIZI-AZAD, CSR, RPR, RSR 25 JOB NO.: CSR NO. 14125 519721	1 A P P E A R A N C E S 2 3 FOR THE PETITIONER, REALTEK SEMICONDUCTOR CORP.: 4 ORRICK, HERRINGTON & SUTCLIFFE, LLP 5 BY: JEFFREY JOHNSON, ESQ. 6 609 MAIN STREET 7 40TH FLOOR 8 HOUSTON, TEXAS 77002 9 (713) 658-6400 10 3J6PTABDOCKET@ORRICK.COM; 11 REALTEK-AMD_OHS@ORRICK.COM 12 13 FOR THE PATENT OWNER, ATI TECHNOLOGIES ULC: 14 MINTZ, LEVIN, COHN, FERRIS, GLOVSKY AND 15 POPEO, P.C. 16 BY: REZA DOKHANCHY, ESQ. 17 3580 CARMEL MOUNTAIN ROAD 18 SUITE 300 19 SAN DIEGO, CALIFORNIA 92130 20 (858) 314-1596 21 RDOKHANCHY@MINTZ.COM 22 23 24 25 ALSO PRESENT: 26 JERRY BUTLER, VIDEOGRAPHER; 27 KIERAN MORRISON, DEPOSITION TECHNICIAN; 28 STEPHANIE STUTZMANN, PARALEGAL WITH MINTZ LEVIN 29 30 31 32 33 34 35
1 UNITED STATES PATENT AND TRADEMARK OFFICE 2 ----- 3 4 BEFORE THE PATENT TRIAL AND APPEAL BOARD 5 ----- 6 7 REALTEK SEMICONDUCTOR CORP., 8 Petitioner, 9 v. 10 ATI TECHNOLOGIES ULC, 11 Patent Owner. 12 ----- 13 14 Case No. IPR 2023-00687 15 U.S. Patent No. 8,854,381 16 17 18 DEPOSITION OF MASSOUD PEDRAM, PH.D, VOLUME I 19 TAKEN ON BEHALF OF THE PATENT OWNER 20 REMOTELY VIA ZOOM VIDEOCONFERENCING, IN 21 LOS ANGELES, CALIFORNIA, BEGINNING AT 22 9:12 A.M. AND ENDING AT 4:18 P.M., ON 23 FRIDAY, JANUARY 5, 2024, BEFORE 24 NATALIE PARVIZI-AZAD, CERTIFIED SHORTHAND 25 REPORTER NUMBER 14125.	1 I N D E X 2 3 WITNESS PAGE 4 MASSOUD PEDRAM, PH.D 5 EXAMINATION BY MR. DOKHANCHY 6 6 7 E X H I B I T S 8 EXHIBIT NO. DESCRIPTION PAGE 9 EXHIBIT 1 DECLARATION OF DR. MASSOUD 7 10 PEDRAM IN SUPPORT OF 11 PETITION FOR INTER PARTES 12 REVIEW OF U.S. PATENT NO. 13 8,854,381 14 EXHIBIT 2 EXPERT REPORT OF MASSOUD 15 15 PEDRAM, PH.D. REGARDING 16 INVALIDITY OF U.S. PATENT 17 NOS. 7,742,053 AND 8,854,381 18 EXHIBIT 3 US PATENT NO.: 8,151,095, 41 19 WILT REFERENCE 20 EXHIBIT 4 US PATENT NO.: 8,854,381, 80 21 MANTOR REFERENCE 22 EXHIBIT 5 US PATENT NO.: 8,318,774, 88 23 JIAO REFERENCE 24 EXHIBIT 6 PCT. WORLD INTELLECTUAL 123 25 PROPERTY ORGANIZATION, INTERNATIONAL APPLICATION NUMBER: WO 00/62182, STUTTARD REFERENCE EXHIBIT 7 BY THE UNITED STATES ITC, 166 INITIAL DETERMINATION ON VIOLATION OF SECTION 337 AND RECOMMENDED DETERMINATION ON REMEDY AND BOND, INV. NO. 337-TA-1318 (**CONFIDENTIAL)

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 2 of 182

Conducted on January 5, 2024

5 1 REMOTELY IN LOS ANGELES, CALIFORNIA 2 FRIDAY, JANUARY 5, 2024, 9:12 A.M. 3 4 THE VIDEOGRAPHER: Here begins media 5 number 1 in the videotaped deposition of Dr. Massoud 6 Pedram in the matter of "Realtek Semiconductor Corp. 7 versus ATI Technologies" in the United States Patent 8 and Trademark Office. Case number IPR2023-00687. 9 Today's date is January 5th, 2023, and the 10 time on the video monitor is 9:12 a.m. Pacific 11 Standard Time. The remote videographer today is 12 Jerry Butler representing Planet Depos. 13 All parties of this video deposition are 14 attending remotely. Would counsel please voice 15 identify themselves and state whom they represent, 16 beginning with the taking attorneys. 17 MR. DOKHANCHY: Good morning. Reza 18 Dokhanchy from Mintz Levin on behalf of the patent 19 owner, ATI. 20 MR. JOHNSON: Jeffrey Johnson from Orrick, 21 Herrington & Sutcliffe on behalf of Realtek and the 22 witness. 23 THE VIDEOGRAPHER: Mr. Johnson, your audio 24 is kind of muffled. Could you restate who you're 25 with?	7 1 bit of a headache, but something that's manageable, 2 so we can continue. 3 Q. Okay. You understand that you're here to 4 testify in the matter of the inter partes review 5 IPR2023-00687 regarding US patent number 8,854,381; 6 is that right? 7 A. Yes, that is correct. 8 Q. Okay. Let's go ahead and introduce 9 Document 2, please. 10 DEPOSITION TECHNICIAN: Okay. Just one 11 moment. 12 THE CERTIFIED STENOGRAPHER: Counsel, when 13 you say "Document 2," is that being marked as 1? 14 MR. DOKHANCHY: Yes, thank you. As 15 Exhibit 1. 16 (Exhibit 1 marked.) 17 BY MR. DOKHANCHY: 18 Q. Okay. Dr. Pedram, Exhibit 1 is the 19 declaration of yours in this IPR proceeding; is that 20 correct? 21 A. That is correct. Yes, sir. 22 Q. And feel free to either download the copy 23 that is on the Dropbox or I believe you have a local 24 copy that you've said is unmarked. You can refer to 25 that as well.
6 1 MR. JOHNSON: I'm with Orrick and on 2 behalf of Realtek and behalf of the witness. 3 THE VIDEOGRAPHER: Thank you, Counsel. 4 The court reporter today is Natalie Parvizi, 5 representing Planet Depos. 6 The witness will now be sworn. 7 THE CERTIFIED STENOGRAPHER: Please raise 8 your right hand to be sworn. 9 10 MASSOUD PEDRAM, Ph.D., 11 having declared under penalty of perjury to tell 12 the truth, was examined and testified as follows: 13 14 EXAMINATION 15 BY MR. DOKHANCHY: 16 Q. Good morning, Dr. Pedram. 17 A. Good morning. 18 Q. I understand you recently became a little 19 ill, so we really appreciate you making this 20 deposition work remotely. 21 A. Thank you. 22 Q. Given your recent illness or any other 23 reason, is there any reason why you cannot give 24 truthful and accurate testimony today? 25 A. At this point, I'm okay. I have a little	8 1 A. I have it open at this time and it's 2 unmarked and in front of me. Please go ahead. 3 Q. If you can turn to paragraph 41. This is 4 in your claim construction section and item B is the 5 definition for task. You state here that you are 6 applying the definition of task as a unit of 7 processing work or a portion of application -- 8 excuse me -- of an application to be executed and it 9 need not include a priority; is that right? 10 A. Yes, sir. 11 Q. Is that a correct definition of the word 12 "task"? 13 A. For the purpose of this hearing, this is 14 the construction I have applied. 15 Q. I understand that. Is that a correct 16 definition of the word "task," in your opinion? 17 MR. JOHNSON: Objection. Form. 18 A. As far as the patents in -- in litigation 19 are concerned, in this particular patent '381 and 20 the prior art, yes, this is the correct 21 construction. 22 Q. Is it correct that a task is the smallest 23 unit of processing work that could be done for an 24 application? 25 A. That's not what I've said. That's not the

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 3 of 182

Conducted on January 5, 2024

9 1 construction applied. It's just a unit of 2 processing work. 3 Q. Could any other definition of the word 4 "task" be correct? 5 MR. JOHNSON: Objection. Form. 6 A. Potentially, yes, depending on what the 7 context -- what the reference is that's being 8 discussed. But in the context that we are looking 9 at the term "task," I believe this is a correct 10 construction. 11 Q. In the context of the '381 patent, could 12 another definition besides the one you're applying 13 here be correct for the term "task"? 14 A. I have not considered that. 15 Q. Okay. So as you sit here today, you have 16 no opinion on whether, in the context of the '381 17 patent, another definition besides the one you have 18 applied could be correct. 19 Is that fair? 20 A. Yes, that's fair. 21 Q. What is your understanding of the word 22 "thread" in the context of the '381 patent? 23 A. A thread is a stream of instructions. A 24 sequence of instructions that get executed. 25 Q. Sorry. I just want to -- it looks like	11 1 house could result in demolishing existing 2 structures, in laying the foundation, raising the 3 walls, in doing stucco, and putting the windows in 4 and so on, so forth. So the task could result in 5 multiple subtasks. Multiple subtasks result in 6 multiple threads and so on, so forth. 7 Q. Okay. So a task can be composed of 8 multiple threads; right? 9 A. Or a single thread. I mean, there's no 10 requirement that a task results in multiple threads. 11 Task could be a single thread. A task could be 12 multiple threads. And depending on the context, 13 also, a -- a thread itself might be a simple one or 14 a very complicated one. So there's no one-to-one 15 correspondence or, necessarily, one-to-many. Could 16 be many-to-one as well. 17 Q. So when you say it "could be one-to-many" 18 or "many-to-one," are you saying there could be 19 multiple tasks within one thread? 20 A. In the context of the '381 and the prior 21 art, this is not the construction I'm applying. So 22 in that particular context, a task could be a high 23 level task that comes in, generates some lower level 24 tasks. These lower level tasks could generate the 25 corresponding threads that get executed in hardware.
10 1 the realtime may have gotten part of your answer. 2 Maybe I'll just read it back and tell me if this is 3 what you were attempting to say. 4 So is it correct that "a thread is a 5 stream of instructions or a sequence of instructions 6 that get executed"? 7 Is that correct? 8 A. Yes, sir. It is. 9 Q. What is the relationship between a task 10 and a thread? 11 A. A task could result in one or more threads 12 to be spawned and subsequently executed in an 13 execution unit. 14 Q. What do you mean by "result in one or more 15 threads"? 16 A. If you think of a task as a -- some kind 17 of work to be performed by a computational engine, 18 that work may have different components associated 19 with it. It may comprise of multiple subtasks or 20 lower-level tasks, each of which might result in, 21 again, a one or more thread to be spawned so that 22 when all these thread are executed, in totality of 23 that execution, the task has been executed. 24 An example would be if you're building a 25 house and that's the task. The task of building a	12 1 So it is a one-to-many, potentially. 2 Although, it can be also one-to-one mapping. In 3 general, the other way around is also possible, but 4 that's not the -- in the '381 patent or in the prior 5 art references that I've analyzed. 6 Q. Okay. So your opinion is that a task 7 could be composed of one thread; correct? 8 A. Yes, sir. 9 Q. And conversely, a thread can be a task; 10 correct? 11 A. Or can -- yes, but it can also be part of 12 a task. 13 Q. And it is with the opinion that a thread 14 can be a task that you rendered your opinions in 15 this proceeding; correct? 16 MR. JOHNSON: Objection. Form. 17 A. My opinion in this proceeding is based on 18 totality of the understanding of form by looking at 19 the '381, its prosecution history, all the prior art 20 references, knowledge of a person skilled in the art 21 at the time of the invention, as well as all the 22 other information that has become available to me. 23 For example, as part of the ALJ proceedings as well 24 as the initial determination by the PTAB and the 25 claim constructions that they have put in place.

Conducted on January 5, 2024

13 1 So I would not agree with the statement 2 that my opinion expressed in my report is based on 3 the notion that a thread can be a task alone. There 4 are many other factors that go into that opinion. 5 Q. Right. I wasn't implying with my question 6 that it was the only thing you considered in 7 rendering your opinions. So let me restate the 8 question. 9 A. Even in that case, the key issue here in 10 rendering my opinion was not on the basis that a 11 thread can be a task, but the fact that there's a 12 relationship between tasks and threads. And the 13 relationship starts from the notion that a task is a 14 unit of workload or part of an application that 15 comes in to be executed. And then, in the process 16 of that task being executed, it may or may not, but 17 it can certainly be decomposed into a number of 18 lower-level tasks and spawn a number of threads that 19 then are dispatched to various execution units and 20 get executed. 21 And when all these subtasks and responding 22 threads are executed, then that task is also 23 executed. That is the notion that has been 24 important, and that's the one that I've applied 25 consistent with the PTAB's claim construction.	15 1 MR. JOHNSON: Objection. Form. 2 A. I believe I've already answered it. 3 Q. You haven't. So what is the answer: Yes, 4 you can answer "yes" or "no," or no, you cannot 5 answer "yes" or "no"? 6 A. I cannot answer "yes" or "no." 7 Q. Let's introduce, please, as Exhibit 2, 8 Document 18. 9 DEPOSITION TECHNICIAN: Pulling it up. 10 One moment. 11 (Exhibit 2 marked.) 12 BY MR. DOKHANCHY: 13 Q. Okay. Dr. Pedram, Exhibit 2 is your 14 report regarding invalidity of the '053 and '381 15 patent from the ITC proceeding 337-TA-1318; is that 16 correct? 17 A. Yes, sir. 18 Q. Okay. And I want to jump down to page 30. 19 And starting on page 29, there's your section 20 regarding claim constructions; correct? 21 A. Yes. 22 Q. And on page 30, you address the word 23 "task." And you explain that the ALJ has construed 24 task as "a unit of processing work or a portion of 25 an application to be executed" and it not need --
14 1 Q. Your opinion is that a thread can be a 2 task; correct? 3 MR. JOHNSON: Objection. Form. 4 A. Again, in the process I described to you, 5 there's logical connection between tasks and 6 threads. A task could result in a single thread if 7 it's simple enough. On the other hand, a task could 8 result in a number of threads, small or large number 9 of threads, that then have to be executed for the 10 task to be completed. 11 In one extreme case, a task could become, 12 or yet could only spawn, a single thread. But this 13 statement is different than saying a thread is a 14 task, in some cases. A task could generate only in 15 the end one thread or multiple threads. These are 16 different statements. 17 Q. Are you able to answer "yes" or "no" to 18 the question: Your opinion is that a thread can be 19 a task? 20 MR. JOHNSON: Objection. Form. 21 A. I believe I've already answered the 22 question. I can repeat it, but I cannot add 23 anything more to that. 24 Q. Okay. So you cannot answer "yes" or "no" 25 to my question; correct?	16 1 excuse me, "need not include a priority." 2 Is that right? 3 A. Yes. 4 Q. And that's the same definition that you 5 applied in this IPR proceeding; correct? 6 A. Yes. 7 Q. And under that you comment, "Threads are 8 not tasks, nor do I understand the ALJ's 9 construction to allow such an interpretation." 10 Right? 11 Is that correct, Dr. Pedram? 12 A. Yes. Yes, sir, it is. 13 Q. And when you said "threads are not tasks," 14 what did you mean? 15 A. I go on to explain it below. So if you 16 scroll down. Unfortunately, I don't have a copy of 17 this on my desktop, so I have to rely on this 18 document. So if you go down, I have a discussion 19 about why I say what I say. I can -- I'll be glad 20 to -- 21 Q. Okay. We can go down there in a second. 22 But just looking at this statement, do you recall 23 what you meant when you said "threads are not 24 tasks"? 25 A. This -- this had been some time ago. Can

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 5 of 182

Conducted on January 5, 2024

17 1 you please scroll down? 2 Q. Sure. You explain that statement by 3 saying, "Generally, tasks are considered to be a 4 higher-level program construct than a thread, 5 meaning that a task may be decomposed into or give 6 rise into multiple threads" -- "to multiple 7 threads." 8 Is that correct? 9 A. Yes. That's what I also said earlier 10 today, yes. 11 Q. And you still believe that to be correct; 12 right? 13 A. Yes, sir. 14 Q. So with that understanding, what did you 15 mean when you said "threads are not tasks"? 16 A. Exactly what I've said here. "Tasks are a 17 higher-level program constructs," and they could 18 result in one or more threads to be executed, but 19 they are not the same as a thread itself. 20 And I go on to give the example of the 21 displaying a polygon on a user device and showing 22 that doing such a rendering of a polygon on a user 23 display device will result in a plurality of 24 different threads. Vertex processing threads or 25 pixel processing threads, or a combination of the	19 1 objection? 2 MR. JOHNSON: Yeah. Objection. Form. 3 You can answer the question. 4 A. What I mean is that there are hundreds of 5 pages of report written. Many, many examples I 6 gave, many different things, and I cannot recall 7 that at no place in any of my reports or any of the 8 depositions that was taken from me or the court 9 proceedings or the -- any -- any other communication 10 that took place, I did not provide an example in 11 which a task is a single thread. 12 Maybe the question was not asked or the 13 issue did not come up, but again, I cannot confirm 14 one way or the other. I might have said it. I 15 might not have because I was not asked. But if I 16 had been asked, I would have definitely said that a 17 task could also result in a single thread if it's 18 simple enough. 19 Q. Dr. Pedram, my question was about your 20 report, so please try to pay attention the question. 21 In your report in the ITC, you gave an 22 example of decomposing a thread -- let me start 23 over. 24 In your report in the ITC when 25 interpreting what the word "task" means, you said "a
18 1 two. 2 Q. So now you are saying that tasks can be 3 decomposed into one or more threads. But do you see 4 here that on page 31 of your report, you didn't say 5 "one or more"? You said "multiple"; correct? 6 MR. JOHNSON: Objection. Form. 7 A. That's what I've meant. That's what I've 8 always meant. To me, it is obvious that if a task 9 could result into multiple threads, in the same 10 spirit, in the same way, if a task is simple enough, 11 it can result in a single thread. 12 There's no requirement and, in fact, would 13 have been sort of incorrect to assume that a single 14 task could necessarily result in multiple threads. 15 Maybe single thread is enough because the task 16 itself is a simple task. 17 Q. In your report in the ITC, you only 18 mention decomposing a thread -- excuse me -- a task 19 into multiple threads. You don't discuss 20 decomposing a task into a single thread. 21 Is that fair? 22 A. I cannot confirm this. 23 MR. JOHNSON: Objection. 24 THE WITNESS: Sorry. Should I continue? 25 THE CERTIFIED STENOGRAPHER: Was there an	20 1 task may be decomposed into multiple threads" but 2 you never said "it may be composed of a single 3 thread"; correct? 4 MR. JOHNSON: Objection. Form. 5 A. In the table you are showing me, and it's 6 in front of me, it is a correct statement that I've 7 said "a task may be decomposed into or give rise to 8 multiple threads." Note that I use the term "may 9 be." Not that it has to be of necessity that it 10 will have to be decomposed into multiple threads. 11 It may. That's what I've said here. And if you're 12 asking me to confirm that what I see in front of me 13 states that, yes, I do confirm this. 14 If you are asking me in that whole report, 15 is there any other place in which I have mentioned 16 that a task could be a single thread? Maybe I have 17 said it. Maybe I have not. I just don't remember 18 at this point. 19 Q. How about in the claim construction 20 section of your report? 21 A. Yes, in the claim construction section of 22 my report, again, the portion that you are showing 23 me -- because I really don't remember everything 24 else that is in that section at this point in 25 time -- but in the section you are showing me, I do

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 6 of 182

Conducted on January 5, 2024

21 1 confirm that the only statement I've made regarding 2 a task being -- and its relation to thread which is 3 shown and it's in front of me, this portion states 4 that a task may be decomposed into or give rise to 5 multiple threads. 6 Q. All right. If you need to download the 7 exhibit, feel free do that. But my question to you 8 is: In the claim construction section in the ITC 9 report on invalidity, you do not assert that a 10 thread -- excuse me -- a task cannot be decomposed 11 into a single thread; correct? 12 A. I need to download and take a look. I 13 don't have a copy of this with me now. 14 So it is a correct statement that in the 15 claim construction section of my ITC report on 16 invalidity, I have stated that a task may be 17 decomposed into multiple threads. In that portion, 18 I did not explicitly state the other possibility 19 that a task may also result only in a single thread. 20 Q. Okay. 21 A. And the reason is, I was trying to show an 22 example of where clearly one could see that a task 23 is different than a thread. And that would be a 24 case in which a task is complex and, therefore, it 25 would result in multiple threads. But even in the	23 1 you're talking about with an example? 2 A. I believe I already did that in the 3 context both a high-level programming language 4 versus the machine code. But if I'm taking a same 5 task, such as -- which could be a very simple task, 6 such as, maybe, just render this particular vertex, 7 depending on what the underlying hardware is, this 8 may translate into a different thread in only one 9 hardware platform versus the other hardware 10 platform. It's very machine-specific at that point 11 in time. 12 Q. So if the task is render a particular 13 vertex, what would the thread be that it decomposes 14 into? Or what would an example be? 15 MR. JOHNSON: Objection. Form. 16 A. I need to have a lot more information to 17 answer. This depends on the microarchitecture of 18 the underlying hardware, the particular sequence of 19 control signals are, and the low-level language 20 constructs are that get decoded and understood by 21 the hardware to execute that particular task. 22 So on an NVIDIA GPU versus an AMD GPU, for 23 example, the threads that correspond to even a 24 single task would have different representations. 25 They look different from each other.
22 1 case that a task would result in a single thread, 2 the task and the thread are not the same constructs. 3 They are different constructs. 4 Q. Explain that, please. Why are they 5 different constructs? 6 A. A thread is something that gets executed 7 in the end on piece of hardware. And it might be 8 necessary to take a high level task which may be a 9 simple task and rewrite it, restructure it for 10 efficiency reasons or for feasibility of execution, 11 so that it can then run on hardware. 12 It is in the same way that if you have a 13 program written in a high-level programming 14 language. An instruction that you see in that 15 high-level programming language would have to be 16 converted. Sometimes into a single machine code, 17 microinstruction, or multiple microinstructions that 18 then get mapped into hardware. 19 The task itself is just a unit of workload 20 or work to be performed or a portion of an 21 application. It may have to be rewritten. 22 Transformed in such a way that it can then 23 efficiently map -- get mapped into hardware that 24 implements it. 25 Q. Can you explain the transformation that	24 1 Q. Without knowing those details, would you 2 be able to map the thread to the task that it was 3 decomposed to -- from, rather? 4 MR. JOHNSON: Objection. Form. 5 A. I need a lot more information to be able 6 to answer this question. Obviously, there are ways 7 to do reverse engineering and go back from low-level 8 threads to the high-level tasks that created those 9 threads. Sometimes it's doable. Sometimes it's not 10 doable. And there's a lot of information needed 11 about the microarchitecture, about the hardware, the 12 machine, the compiling that's being used, the 13 programming involvement, different things. 14 So generally, it's a difficult task. It 15 may be possible in some cases. Sometimes not. I 16 don't know. 17 Q. Speaking generally, without knowing the 18 technical details that you described, would it be 19 possible to map the single thread to the single task 20 that it was decomposed from? 21 MR. JOHNSON: Objection. Form. 22 A. I really cannot answer that question. I 23 have to consider this more carefully to see whether 24 the -- this determination can be made. If it can be 25 made, under what conditions, what additional

Conducted on January 5, 2024

25 1 information is needed. I would just say it's a not 2 a trivial task to do so -- undertaking to do so, but 3 it can be possible. Hard to know and state at this 4 time. 5 Q. What additional information would you need 6 to be able to perform that mapping? 7 A. For example, I need to know what kind of 8 bookkeeping is happening in the hardware itself. Is 9 the hardware itself keeping track of these 10 correspondences or not? And if it does keep track 11 of these correspondences, how unambiguous -- or I 12 should say disambiguous this bookkeeping is, how 13 frequently it's updated, at what level of detail the 14 books are kept, and so on. If you imagine a task 15 running on -- multiple tasks running in a complex 16 parallel processing system, such as a GPU, there 17 could be tens of thousands of threads running at the 18 same time. And if you say, "Look, at this 19 particular thread running on this particular 20 processing element, seeing this process block, tell 21 me who the parent high-level task is that spawned 22 this particular thread," it's not a trivial job and 23 requires hardware support for itself. And depending 24 on what kind of hardware support is there, you could 25 answer the question, potentially. But again, you	27 1 determination or correspondence between a parent 2 task and a child thread that might have gone through 3 multiple levels of translation from tasks to 4 subtasks to sub-subtasks to then threads and so on. 5 Q. And in your answer just now, you mentioned 6 high-level tasks. Is your testimony the same as to 7 what you have called in your report "low-level 8 tasks"? 9 A. Yes, I think -- I'll tell you I said, 10 "yes," meaning that I acknowledge I hear the 11 question, nothing else. 12 I would say that I generally don't believe 13 that we can assign a fixed number of task 14 granularities, say, high and low, and be done with 15 it. I mean, there could be high-level tasks, 16 intermediate-level tasks, subintermediate-level 17 tasks, low-level tasks, very low-level tasks, 18 whatever. And it depends on the architecture. It 19 depends really what happens to a task when it flows 20 down in the hardware to get executed. Sometimes, 21 again, only one level of translation from a task to 22 thread happens. Sometimes a high-level task becomes 23 multiple low-level task -- tasks. And then, these 24 multiple low-level tasks become -- each one of them 25 becomes multiple threads. Sometimes, there's more
26 1 need to know what the hardware is doing. 2 Q. And when you say "hardware," what specific 3 components of hardware are you referring to? 4 A. In the GPU, there are lots of different 5 resources, controllers, memory scheduling units, 6 synchronization elements, performance monitoring 7 units, and so on. So usually, to track something 8 that was a high-level task usually coming through an 9 API to all of its children threads. Or in using the 10 term "decomposed threads" requires information about 11 exactly how the tasks are coming in, how they get 12 routed, how they get assigned, what processing 13 element they went to, what slots within the 14 processing element they were assigned to. Many of 15 these processing elements are simultaneously 16 multistranded processing elements. So it's -- for 17 example, or what have you. And so as a result, you 18 need to have a lot of information. 19 And again, it's something that you're 20 asking me to opine on on the fly. I'm doing my best 21 to be responsive to you, but I really need to 22 consider this a lot more carefully to even be able 23 to answer the question that you're asking, which is: 24 What information about the hardware, architecture, 25 microarchitecture is needed to be able to make that	28 1 levels in between. Three levels, four levels of 2 translation. So that's hard for me to only use the 3 taxonomy that there are high-level tasks and 4 low-level tasks and nothing else. 5 Q. I understand. I was just using the words 6 that you used in your report. But we can address 7 the concept that you just raised. 8 So my question is this: For all of the 9 variety of levels of tasks that one could conceive 10 of and that you just discussed, is your opinion the 11 same regarding mapping of the low-level thread up to 12 the various levels of tasks? 13 MR. JOHNSON: Objection. Form. 14 A. I'm not sure if I understand your question 15 properly, what you mean by what you asked. Let me 16 answer it this way. If what you're saying is that: 17 Is the level of difficulty associated with 18 identifying the -- a lower-level task parent of a 19 thread the same as identifying the higher-level 20 parent of the same thread? I mean, again, that's 21 one of those questions hard for me to answer. 22 Generally, the higher up you go, the more 23 work you need to do to identify the parent; right? 24 It's like if I ask you who your father is, you 25 immediately answer. Your grandfather, yes. But

Conducted on January 5, 2024

29 1 keep going up five generations ago, what was the 2 name of your great, great, great grandfather and 3 what was he doing and where he lived. Maybe you 4 know, maybe you don't. It gets more difficult as 5 you go up. So the more levels, in terms of 6 translation, exists between the task coming in 7 through the API and what gets executed as thread in 8 hardware, usually the more challenging it is to 9 establish that correspondence. 10 But I'm not saying it's not doable. As 11 I'm sure you know, you could also potentially trace 12 your ancestry to generations and generations ago if 13 you wanted to. 14 Q. Can we go back to Exhibit 2, please. 15 A. Yes, sir. 16 Q. And on page 31 -- okay. You give an 17 example of the task of displaying a polygon. 18 Do you see that? I can try to highlight 19 it, but it's right here. I don't know if that shows 20 up for you. 21 A. Oh, yes. Okay. I see it, yes, sir. 22 Q. What are the threads that the task of 23 displaying a polygon would be decomposed into? 24 A. I don't know if I can tell you all the 25 corresponding subtasks and threads that get	31 1 are involved. Texture shading would be a pixel 2 processing thing, whereas coordinate transformations 3 and many aspects of lighting are vertex processing, 4 and so on and so forth. 5 MR. DOKHANCHY: Let's take a ten-minute 6 break if that works for you, Dr. Pedram. 7 THE WITNESS: If you don't -- I like -- 8 I'm okay to continue. If you need a break, please, 9 let's take it. But I like to do as much as we can 10 early in the day before I get too tired. You know I 11 have COVID-19, and my experience last time was that 12 toward the evening, I get very tired and so on. So 13 I'd rather go straight -- more in the morning so 14 that I can retire earlier -- earlier in the day. 15 But if you need the time, by all means, please, sir. 16 MR. DOKHANCHY: Okay. We'll be as quick 17 as we can, but let's pause for ten now and come back 18 at 10:17. 19 THE WITNESS: Thank you, sir. 20 THE VIDEOGRAPHER: We're going off the 21 record. The time is 10:07 a.m. 22 (Recess.) 23 THE VIDEOGRAPHER: We're going back on the 24 record. The time is 10:19 a.m. on the video 25 monitor.
30 1 generated, but I can provide some examples. I mean, 2 one thing you have to do is to take the -- change 3 the coordinate systems to the -- from the world 4 coordinate system to the display coordinate system 5 that you're interested in. Then you have to look 6 and see when you display this object, this polygon, 7 what are the characteristics of the polygon itself? 8 What kind of surface does it have? Is it 9 transparent or not? If it's not transparent, 10 what other portions of the elements underneath that 11 polygon that already existed now will be hidden 12 because the polygon is being drawn on top of them? 13 If it has a shiny surface, how should I show the 14 shine on the surface depending on the one -- on the 15 location of the light sources for that display for 16 that scene. What's the angle of the viewing of the 17 polygon? 18 I mean, so that's what graphics processing 19 is. When you try to render something that includes, 20 for example, rendering this polygon, lots of things 21 having to do with the coordinates, with the 22 lighting, with the impact on surrounding objects, 23 with the texture -- texture of the polygon itself. 24 And that's why you do both vertex-type processing as 25 well as pixel-type processing. Because all of these	32 1 BY MR. DOKHANCHY: 2 Q. All right. Could we please put back up 3 Exhibit 1. Okay. 4 Dr. Pedram, on page 32 of your declaration 5 in this IPR proceeding begins your "Ground 1" 6 analysis; is that right? Regarding the Wilt 7 reference; is that correct? 8 A. Yes, sir. 9 Q. And down at paragraph 76, you discuss 10 the -- you discuss the "shader core configured to 11 receive the plurality of tasks" requirement in the 12 '381 patent; correct? 13 A. Yes, sir. 14 Q. And your opinion is that -- withdrawn. 15 Your first opinion is that each parallel 16 processing unit 202 in Wilt is the claimed shader 17 core; correct? 18 A. Yes, sir. 19 Q. You do not opine that any subcomponent or 20 group of subcomponents of the PPU in Wilt satisfies 21 the shader core requirement of the claims; correct? 22 MR. JOHNSON: Objection. Form. 23 A. I identified the shader core as PPU 212 in 24 Wilt after having identified the engines in the Wilt 25 reference. And, obviously, if a component that has

Conducted on January 5, 2024

33 1 been identified as an engine is within that PPU, I 2 did not intend to mean that that same component is 3 part of the shader core as well. What I mean 4 specifically is I identified the front-end unit 212, 5 the work distribution unit 210, and the core 6 interface, I believe it was 303, as the engines in 7 this case. And therefore, what remains in the PPU 8 other than these identified engines would constitute 9 the shader core. 10 Q. Okay. So your opinion today is that 11 everything that remains in the PPU other than the 12 engines that you've identified is the shader core; 13 correct? 14 MR. JOHNSON: Objection to form. 15 A. That would be correct, yes. That's fine, 16 except that, again, if you look at the -- there is 17 an element in the PPU called the host interface. 18 The host interface provides the -- the interface 19 between the memory subsystem and these PPUs. And 20 that is also not part of the computation of cores, 21 because the function that's performed by the shader 22 core is to really execute the tasks that they 23 receive. And the host interface is also not 24 anything that contributes to that function. So I 25 would say let's exclude the engines and the host	35 1 a collection. It could be a group of -- and maybe 2 there are some other alternative terminology that 3 one could use to convey the same construct here, 4 which is an array. 5 Q. Do the processing elements need to each be 6 the same? 7 MR. JOHNSON: Objection. Form. 8 A. I have not considered this question, 9 because it did not come up in any of the proceedings 10 I've come across throughout this case, in any of the 11 arguments presented from your experts or in your, I 12 guess, IPR filings and so on. But generally, the 13 sentence as is written, "array of processing 14 elements," is without any limitation on what the -- 15 these processing elements are and whether they have 16 to be homogeneously architected or they can have 17 different architectures, and therefore they look 18 different from each other. 19 So I would say the limitation itself does 20 not define one way or the other. The array of 21 processing elements could be homogeneous processing 22 elements all having the same microarchitecture or 23 they could be different. 24 Q. What is a processing element? 25 MR. JOHNSON: Objection. Form.
34 1 interface that's a gateway that connects to the 2 front-end unit 212. And then, the rest is the 3 shader core, yes, sir. 4 Q. In paragraph 40 of your declaration, you 5 state that you are adopting the construction of 6 shader core as being, quote, "an array of processing 7 elements." 8 Correct? 9 A. Yes, sir. 10 Q. What is your understanding of the word 11 "array"? 12 A. A collection. 13 I'm sorry, I said "collection" not 14 connection. 15 Q. So you would equivalently -- withdrawn. 16 So the term "shader core," to you, would 17 equivalently mean a collection of processing 18 elements; is that correct? 19 A. For example, you -- I think the term 20 "array" has a very clear meaning to anyone. 21 Certainly to a POSITA skilled in the art, an array 22 of elements, an array of processing elements has a 23 very well-defined meaning. So I don't need to 24 redefine it. But you asked me to do it, and so I 25 said a synonym for "array" in this context would be	36 1 A. It is an element that does processing. 2 Q. What do you consider processing? 3 MR. JOHNSON: Same objection. 4 A. In the context of what is being discussed, 5 which is '381 and the prior art references, the -- 6 in -- processing would be general-purpose 7 processing. It could mean graphics processing. And 8 when it's graphics processing, it could be vertex 9 processing, geometry processing, pixel processing. 10 That's it. 11 Q. I think we're running into the problem of 12 using a word in its own definition, so let me -- let 13 me try again. 14 What is processing? 15 MR. JOHNSON: Objection. Form. 16 A. I'm sorry, this is a very open-ended 17 question. I mean, what is processing in what 18 context? Please point me to the context that you're 19 referring to, and then I can provide a response. 20 In the context of '381 and the asserted 21 prior art references, I did explain to you that 22 processing could be general-purpose processing or 23 graphics processing, as examples. And I gave you 24 even subclassification of graphics processing, 25 including vertex, geometry, and pixel. And I

Conducted on January 5, 2024

37 1 thought that would be adequate. But if it's not, 2 please let me know what the missing point is from 3 your perspective, and I'll be glad to provide 4 additional details. 5 Q. Thank you. What does it mean to process 6 something? 7 MR. JOHNSON: Objection. Form. 8 A. I'm not sure what that something is. I 9 mean, what is -- what does it mean to process sand 10 into glass is a different question than what does it 11 mean to process a computational task, for example. 12 So what do you mean? First, please explain. 13 MR. DOKHANCHY: Karen, can we get 14 Exhibit 1 back up. And let's not take that down 15 until we move to the next exhibit. Let's just keep 16 all the exhibits up, please. 17 BY MR. DOKHANCHY: 18 Q. So in the context of the definition of 19 shader core that you're applying in this case, you 20 have said a processing element is something that 21 processes. My question is: What does the word 22 "processing" mean in this context? 23 A. I believe I've answered that. So again, 24 if you can be more specific. What do you think is 25 amiss in my response that you need further	39 1 Q. If something in a graphics system receives 2 data and simply passes it along, is that a 3 processing element? 4 MR. JOHNSON: Objection. Form. 5 A. Let me be specific. So if you are talking 6 about, for example, a buffer that receives on one 7 end -- its input end -- some form of data, and the 8 next cycle over or when some control signal comes 9 that says, "Now, release it," it will release that 10 input data that's been stored in its buffer storage 11 location to the output from wherever it has to go. 12 That buffer is not doing processing. 13 If on the other hand the data comes into 14 the so-called intermediary structure and undergoes 15 some kind of transformation, some kind of decoding, 16 encoding, some kind of annotation, additional data 17 is being loaded to help it so that when it's 18 released, it's not only a copy that came in but some 19 additional information that was tagged along, then, 20 yes, there is processing taking place there. 21 Q. Going back to the word "array" in the 22 construction of shader core, which is an array of 23 processing elements, does the -- or do the elements 24 of the array need to be connected in any way to one 25 another?
38 1 clarification? I'll be glad to do so. But I've 2 said processing could mean general-purpose 3 processing -- 4 Q. Well, I'll stop you there because I think 5 you're just going to say the same thing. What I 6 asked you is: What does it mean to process 7 something? And you said an example of processing is 8 vertex processing. In that answer, you didn't 9 answer what it means to process. You just gave an 10 example of something that could be processed. So 11 the disconnect here is you haven't answered: What 12 does it mean to process anything? 13 So in the context of the definition of 14 shader core, what does it mean to process something? 15 A. So -- so for example, I -- let's look at a 16 graphics task; right? In a graphics task, a vertex 17 shader may perform processing tasks such as 18 transformations, scaling, and lighting. A pixel 19 process -- a pixel processing task would involve 20 shading textures and so on. 21 There are other tasks, such as 22 tessellation, and so on and so forth. So these are 23 all examples of the kinds of processing that take 24 place in a graphic processing system. I hope that 25 answers your question.	40 1 MR. JOHNSON: Objection. Form. 2 A. There is no limitation as such imposed. 3 The array could be 1D or 2D. There could be 4 connections between the array elements such as 5 nearest neighbor connections, or they could be 6 connected via busses, or they may not. There's 7 nothing in the description, "an array of processing 8 elements," that requires a certain type of 9 connectivity between the array elements. 10 Q. Do the processing elements in your 11 definition of shader core need to be arranged in any 12 particular way to constitute an array? 13 A. Again, there is no limitation implied or 14 inferred about the arrangement of the processing 15 element in that array. The processing elements 16 could be in a one-dimensional array, in a 17 two-dimensional array, in a systolic array 18 configuration. It could have connections among 19 themselves or not. If they have connections, it 20 could be through different mechanisms, 21 point-to-point or bus-based or shared memory system 22 or what have you. 23 There is no such limitation present in the 24 language of, "an array of processing elements," that 25 one could infer or that is necessarily implied from

Conducted on January 5, 2024

41 1 the term itself. 2 Q. Could we please introduce as Exhibit 3, 3 Document 7. 4 (Exhibit 3 marked.) 5 BY MR. DOKHANCHY: 6 Q. All right. Dr. Pedram, Exhibit 3 is the 7 Wilt reference you rely on in your declaration; is 8 that correct? 9 A. Yes, sir. 10 Q. And if we go down to Figure 2, you see a 11 series of what are labeled PPUs: PPU 0, PPU 1 12 through PPU U-1. 13 Do you see that? 14 A. Yes, sir. 15 Q. Are these PPUs connected to each other in 16 any way? 17 A. I mean, you can see that all of these PPUs 18 are connected through the memory bridge 105 to some 19 system memory. So in fact, they can communicate 20 with each other, if they choose to or if the 21 application requires it, through a shared memory 22 system interface. But there is no direct 23 point-to-point connection between these parallel 24 processing units that I could see in this Figure 2. 25 Q. Does Wilt describe anywhere the PPUs	43 1 Figure 2, there is this memory bridge 105. And if 2 you look at Figure 3 of the Wilt reference, this 3 takes us to the global memory that has the PP 4 memory 204 and the system memory 104. And so 5 obviously, there is the possibility for the PPUs to 6 communicate information amongst themselves if 7 needed, if the application requires it. 8 And if you have, for example, coordinated 9 thread groups running at the same time on different 10 PPUs and you need to have data flowing between them, 11 the way they would do it, and a person skilled in 12 the art would understand the way it would be done in 13 Wilt system, is to do such transactions through the 14 shared system memory. For example, 104 or the PP 15 memory 204. 16 That's common knowledge and common 17 practice that when you try to communicate between 18 processes, between the thread groups and so on, you 19 could do it either through direct means of 20 communication, interprocess communication, or you 21 could do it through a shared memory interface. 22 These are the two standard ways of doing it well 23 known to the state of the art since 1980s, at least. 24 So a skilled person in the art would 25 understand that capability exists, that Wilt system
42 1 communicating in any way with one another? 2 A. I cannot recall. 3 Q. And you're not aware of any such opinion 4 in your report -- or your declaration, rather, are 5 you? 6 A. My report only tries to show how the Wilt 7 reference would render obvious by disclosing 8 different elements of various claims to a person 9 skilled in the art. And therefore, when doing that, 10 I quoted excerpts from Wilt and made opinions 11 that -- based on those citations and my 12 understanding of the '381 reference and prosecution 13 history to substantiate my claim of invalidity. And 14 I did not find the need to show any kind of 15 communication between these parallel processing 16 units. My report should not be taken to mean that 17 such communication is absent or that the Wilt system 18 does not enable such communication. It's just I 19 didn't find it necessary to look for it in the Wilt 20 reference. 21 Q. Does Wilt describe any of the 22 subcomponents of one PPU communicating with any 23 other subcomponents of another PPU? 24 A. Again, as you know because you have -- you 25 pointed me to the -- the figures, for example,	44 1 has been configured to allow for such communication 2 if needed by the application. 3 Q. Does Wilt state anywhere that any 4 subcomponent of a PPU communicates with any other 5 subcomponent of another PPU? 6 A. I do not recall off the top of my head. 7 It might, but I just don't recall. 8 Q. And you brought up memory 204. Each PPU 9 has access to its own memory 204, but not access to 10 memory 204 of another PPU; correct? 11 A. That's not how it's shown in Figure 3; 12 right? Figure 3, PPU memory 204 is shown as part of 13 the global memory 320, and it's not shown within the 14 PPU itself. 15 But let me look for this first. 16 Q. Well, before you go looking elsewhere, 17 let's stay on the figure that we were on, because we 18 were talking about Figure 2. In Figure 2, which 19 depicts multiple PPUs, a given PPU has access to a 20 given memory associated with it, but it does not 21 have access to a memory 204 associated with another 22 PPU; correct? 23 MR. JOHNSON: Objection. Form. 24 A. In Figure 2, what is shown is that PPU 0 25 has bidirectional access through memory

Conducted on January 5, 2024

45 1 interface 214 to PP memory 0, 204 -- parentheses, 2 (0), close parentheses -- and similarly, PPU 1 has 3 bidirectional access to PPU memory 1 and so on. 4 So that is correct. But as you can see, 5 these PPUs also talk through the memory bridge, 6 potentially to each other. 7 Q. I -- I'll try one more time because you 8 didn't answer my question. In Figure 2, a given PPU 9 has access to its own PPU memory 204, but it does 10 not have access to PPU memory 204 of any other PPU; 11 correct? 12 MR. JOHNSON: Objection. Form. 13 A. I can't say for sure. I mean, again, I -- 14 I really have to look at the full specification of 15 Wilt to see if there is a possibility for a PPU 0 to 16 also have access in addition to its own local PP 17 memory to some other PPU's local PP memory. 18 It might be possible through that same 19 system interface, but -- so hard to exclude that 20 possibility based on the single figure that you're 21 showing me. I have to look at the full specs. And 22 this is not something I've looked for or looked at, 23 because it was not necessary. And in none of the 24 proceedings and depositions I've seen from AMD or 25 its experts or the declarations, the issue has come	47 1 reference and of course Figure 3 itself, that 2 clearly puts these PP memories as part of the global 3 share memory space. 4 I read it for the record, "Processing 5 engines 302 also have access via memory interface 6 214 to off-chip global memory 320, which can 7 include, e.g. PP memory 204 and/or system memory 8 104, with system memory 104 being accessible by 9 memory interface 214 via host interface 206 as 10 described above. It is to be understood that any 11 memory external to PPU 202 may be used as global 12 memory 320." 13 To me, that is saying that PPU memory 1 14 can be considered as part of the global memory space 15 that's available to PPU 0. 16 Q. Figure 3 shows 1 PPU; right? 17 A. Yes, sir, I understand the question. Let 18 me check please. 19 Yes, sir, I see that. I'm sorry. I -- I 20 see in Figure 3, one PPU is shown but then global 21 memory 320 at the bottom of that drawing include 22 both the system memory 104 and the PP memory 204. 23 Q. And in addition to the one PPU in 24 Figure 3, Figure 3 shows one PP memory 204; correct? 25 A. I'm not sure if that's what it means.
46 1 up or been presented to me. So this is a new topic 2 that I've not looked at, and I need to consider it 3 more carefully to answer it. 4 Q. Okay. Well, feel free look at Wilt. Look 5 at the description at Figure 2 in the specification, 6 and tell me does a given PPU have access to the 7 local PPU memory 204 of another PPU in Wilt? 8 A. I'm looking at this. Thank you. 9 Q. Okay. 10 A. I would say, based on Figure 3 and some 11 other descriptions here, that a PPU 0 has access to 12 its own PP memory 0. And from its perspective, any 13 other memory that's external to it, including PP 14 memory 1 or PP memory U-1, can be pulled together as 15 part of a global memory that PPU 0 go potentially 16 will have access to. So the answer to your 17 question, therefore, is, yes, it is possible from 18 what I understand now, having looked at this a 19 little bit. And I reserve the right to modify it, 20 but at this point, based on -- based on what I did 21 quickly, it seems that it is possible for the PPU 0 22 to actually have access to PP memory 1, or PP memory 23 2, or PP memory U-1. 24 And I can point you to column 7, say, line 25 15, starting line 15. If you read it, of the Wilt	48 1 Because previously when, in Figure 2, they wanted to 2 show one local PP memory, they give it ID 0, 1 down 3 to U-1. This one just says "PP memory 204." This 4 could also mean the PP memory bank of -- group of PP 5 memory or something. It's not clear from this 6 description that the PP memory 204 is necessarily 7 just one of those PP memories. 8 Q. So you're not sure if the lack of an 9 additional numeral after PPU and after PP memory is 10 indicative of whether there are one or more PPUs or 11 one or more PP memories; is that right? 12 A. That's not the basis -- the only basis for 13 what I'm saying. I read to you a full excerpt of 14 column 7 of Wilt. That, to me, clearly indicates 15 that any memory that's not local to PPU 202(0) can 16 be put together as part of the global memory that's 17 accessible to that PPU 0. That excerpt was clearly 18 stating that. 19 And then, in Figure 3, I see PP memory 204 20 being placed under the name -- the general heading 21 "Global Memory 320." And based on these two facts, 22 I would say I cannot conclude from this Figure 3 23 that the PP memory 204 necessarily is just one of 24 the PP memories which is local, or some of the other 25 PP memories that are not local to the PPU 0, for

Conducted on January 5, 2024

49 1 example, and they have been pulled together here as 2 part of the global memory. 3 Again, you're asking me to opine on 4 something that your side has never presented 5 anything about. And you're asking me to do the 6 analysis on the fly. And I'm doing -- I'm trying to 7 do my best, and I found already an excerpt that 8 seems to confirm that a PPU 0 would have access to 9 local memory PPU 1, local memory of PPU 2, and so 10 on. And so, that's what I've done. 11 Q. Global memory could have parts that are 12 accessible only by one PPU and other parts 13 accessible only by other PPUs. 14 Is that true? 15 MR. JOHNSON: Objection. 16 A. Not normally. Normally, that would not be 17 the case. The term "global memory" typically 18 means -- typically means that it's memory that's 19 globally available to all the workers, the worker 20 units, processing blocks -- processing units in the 21 system. That's normally what it means. 22 Q. And it means that all parts of the global 23 memory are accessible to all workers in the system. 24 Is that your opinion? 25 MR. JOHNSON: Objection. Form.	51 1 A. I believe the excerpt that I read to you 2 implies that. "It is to be understood" -- on 3 paragraph 7 of Wilt, starting at line 19, "It is to 4 be understood that any memory external to PPU 202 5 may be used as global memory"; right? So to me, it 6 says it in some ways. 7 Q. Let's look at Figure 2. Figure 2 has more 8 than one memory 204 PPU -- PP memory 204(0), 204(1), 9 et cetera; correct? 10 A. Yes, sir. That's right. 11 Q. Those are depicted in Figure 2 as distinct 12 memories; correct? 13 A. Yes, sir. 14 Q. And in Figure 2, the only memory 204 that 15 PPU 0 has access to is PP memory 0204(0); true? 16 A. Not true. I mean, because imagine PPU 1 17 wanting to have access to PPU memory U-1. It will 18 go through the memory bridge, come back through PPU 19 U-1, passing it to getting to PPU memory U-1. 20 Details, I'm not sure. This is a block 21 diagram. This is much more complicated, much more 22 detailed; right? So if you're saying, "Is there any 23 path?" Yes. There is an indirect path from PPU 1 24 through the memory bridge, going through the PPU 25 U-1, are you bypassing it, going to PPU memory U-1.
50 1 A. Again, I cannot and I do not want to make 2 general statements about all possible architectures 3 in the world. So how do I know what the design of 4 the system was, and what the intention was, and what 5 kind of applications were targeted? 6 If your question is about Wilt, which I 7 assume it is, in the context of the Wilt reference, 8 although, I reserve the right to look at this again 9 more carefully offline and modify what I say now 10 because it's a new question that I've not considered 11 and I've not seen any position taken by AMD or its 12 experts. 13 But from what I've seen now and what I can 14 see in Figure 3 and the excerpt I read to you, seems 15 to me that the global memory would also comprise of 16 the local -- the collection of the local PP memories 17 of the different PPUs, and that there's a 18 possibility that through the memory interface, and 19 the host interface, and so on, that one PPU 0, for 20 example, would have access to local memory of some 21 other PPU. 22 Q. Okay. Does Wilt ever affirmatively 23 describe that PPU 0 has access to PPU memory 1 or 24 any other PPU memory besides PPU memory 0? 25 MR. JOHNSON: Objection. Form.	52 1 That path, even in this figure, exists. 2 Q. I understand what you're saying -- 3 A. I'm sorry. If you're saying, "Is there a 4 direct connection between PPU 1 and PPU U-1?" No, 5 there's no direct connection. But honestly, there's 6 an indirect path through the memory bridge and so on 7 to this PPU memory U-1. 8 Q. I understand you're saying there's a path. 9 My question, Dr. Pedram, is: Does Wilt ever say 10 that PPU 0 accesses the contents of PP memory 1? 11 A. I can't recall. Again, I have to go back 12 and read the full specification, and to see not only 13 if it's expressly stated or not, whether it's also 14 implied that such a connection, such an access is 15 possible. I really have to go back and look at it. 16 I'm not in a position now to say one way 17 or the other. There is either express or implied 18 disclosure that such a connection exists. The one 19 excerpt that I quoted you in column 7 of Wilt seems 20 to indicate that such a possibility is in play, 21 exists. 22 Q. The column 7 portion that you pointed me 23 to references a single PP memory 204. It does not 24 reference multiple PPU memories, such as in 25 Figure 2; correct?

Conducted on January 5, 2024

53 1 A. No, not correct. When Wilt wanted to 2 refer to only one such memory, they use this 3 parenthetical numerals associated with the PPU or 4 associated with the PP memory. For example, as you 5 saw in Figure 2, there's PPU 0, PPU 1. There is PP 6 memory 0, which is named 204(0), and so on, so 7 forth. 8 So when it wanted to identify a particular 9 instance of the PPU or its memory, it would use 10 numerals associated with them. Here it's using that 11 in that excerpt generically to refer to a PPU and to 12 refer a PP memory. And whether the PPU is accessing 13 its only local memory or somebody else's local 14 memory, it's not there. 15 Like the -- the -- that designation is not 16 there. The fact that it says anything external to 17 PPU 0 can be used to -- PPU can be used as global 18 memory for that PPU would imply to me, and to a 19 person skilled in the art, that some other PPU's 20 local memory could serve as part of the global 21 memory that's accessible by a particular PPU, say, 22 PPU 0. That's my reading of it, sir. I understand 23 your reading might be different, but that is mine. 24 Q. So your reading of Figure 3 in the column 25 7 Wilt that you pointed me to, is that it describes	55 1 memory. 2 You cannot be both local and global at the 3 same time. So the fact that this PP memory 204 is 4 shown as part of the global memory 320, in view of 5 that disclosure that I've read to you a couple of 6 times already, tells me that this PP memory 204 may 7 include PP memories of other PPUs also. 8 So that's my thinking at this time, and I 9 reserve the right to look at it again. 10 Q. So can you give me a number? How many PP 11 memory 204 are depicted in Figure 3? 12 A. There's one PP memory block 204 shown as 13 part of the global memory 320 in Figure 3. There's 14 a block there called PP memory 204 and there's only 15 one such block. The question is: What does it 16 mean? 17 Q. PP memory 204 is local memory; correct? 18 A. It's called local PP memory, yes. 19 Q. What does "local" mean in this context? 20 A. The corresponding PP unit would have 21 priority fast access to it. That's one meaning. It 22 doesn't mean that it will have exclusive access to 23 it. This is -- I mean, in fact, this is well-known 24 in the state of the art in parallel processing 25 systems, that different parallel possessing blocks
54 1 more than one PPU and more than one memory; correct? 2 A. What I'm saying here, my understanding of 3 Figure 3 in view of the excerpt in column 7, is that 4 a particular PPU could potentially have access to 5 the local PP memory of other PPUs in the system as 6 part of its global memory space that's accessible to 7 it, depending on the application whether it's needed 8 or not. 9 Q. How many PPUs -- 10 A. That's all I'm saying, sir. Nothing more. 11 Q. How many PPUs are depicted in Figure 3? 12 A. It is showing PPU 202 without any numerals 13 parenthesis, so it is a generic PPU. One of the 14 plurality of the PPUs. 15 Q. So the answer is one? 16 A. Appears that way, yes. 17 Q. How many PP memory 204 are shown in 18 Figure 3? 19 A. In that figure, there's a block -- a 20 single block called "PP memory 204" that's included 21 as part of the global memory 320. One such block is 22 shown. The question is: What does that block mean? 23 Because if it was the local memory of the PPU 202 24 shown above, why is it included as part of the 25 global memory? It's not global memory. It's local	56 1 could have their own memory but that -- which means 2 they have fast access to it and they preferably 3 write their intermediate results there, keep the 4 constants there, so on and so forth. 5 But it does not mean that they do not have 6 access to the local memories of other parallel 7 possessing blocks in the system. It's just slower 8 access. So if -- if I'm running a particular task 9 on a given PPU that requires a lot more memory than 10 I have in my local memory space, I could potentially 11 use the local memory of some other PPUs if the 12 application asks me to do it. 13 This is common. It's not uncommon 14 phenomena that a local memory of a parallel 15 processing block is used by some other parallel 16 processing block if the mechanism for it exists and 17 if the need for it is there. 18 Q. You're not aware of Wilt actually 19 describing that happening; right? 20 A. If you ignore Figure 3 that clearly puts 21 PP memory as part of the global memory and if you 22 ignore the disclosure that I read to you in Figure 7 23 [sic] and maybe there are some other disclosures 24 that I can identify offline, then, yeah. But 25 Figure 3 and the excerpt on column 7, at least those

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

Conducted on January 5, 2024

57 1 two, to a person skilled in the art, disclose that 2 the possibility exist in Wilt system for one PPU to 3 access the local memory of some other PPU. Because 4 as was stated, it is to be understood that any 5 memory external to PPU 202 may be used as the global 6 memory, that's accessible to -- 7 Q. I just want to stop you because you're 8 starting to repeat yourself. I'm not talking about 9 possibility, Dr. Pedram. 10 I'm saying Wilt does not describe one PPU 11 accessing the local memory of another PPU? 12 A. I disagree. 13 Q. Okay. And you disagree because of the 14 portion you pointed me to in column 7 and Figure 3; 15 correct? 16 MR. JOHNSON: Objection. Form. 17 A. Perhaps more, but I have to look at the 18 Wilt reference carefully. I need more time to 19 analyze it. And the more time could be in a matter 20 of hours, not minutes, because I need to read, 21 again, the full Wilt specification, understand, and 22 recollect all the interactions that exist between 23 PPUs and/or all memory system architecture and how 24 it's been used to deal with coordinated thread array 25 accesses, to be able to provide a response -- a more	59 1 A. I'm sorry. You are on paragraph 88? 2 Q. Yes. Paragraph 88 of your declaration 3 contains your opinions about why Wilt discloses 4 claim 19 of the '381 patent; right? 5 A. Yes, sir. 6 Q. I'm going to jump to paragraph 89. 7 Paragraph 89 contains your opinions about why Wilt 8 discloses claim 20 of the '381 patent; correct? 9 A. Yes, sir. That's correct. 10 Q. Okay. And your rely on this context 11 swapping disclosure in Wilt. What is "context 12 swapping"? 13 A. So context swapping generally means a -- a 14 running thread is swapped out to bring in a new 15 thread to be executed. It can also mean that you 16 have multiple threads running at the same time, 17 different context. So for example, there are two 18 different PPU context as part of a single CPU 19 context and that you are swapping between these 20 different PPU context. 21 So that's the meaning. And if you look at 22 page 20 of the '381 patent, this is showing -- or 23 claiming the situation where a -- the shader core 24 configured to receive a plurality of tasks based on 25 the priority. And the reason I brought up the
58 1 clear response. I'll be glad to do it if I'm tasked 2 to do it. I'll get back to you. 3 Q. So now your testimony is not only does 4 Wilt disclose the possibility that one PPU accesses 5 another PPU's local memory, Wilt actually discloses 6 that a PPU -- 7 (Reporter admonition.) 8 MR. DOKHANCHY: I'll restart question 9 over. 10 Now your testimony is not only that only 11 does Wilt disclose the possibility that one PPU 12 accesses another PPU's local memory, Wilt actually 13 discloses that one PPU does access the contents of 14 another PPU's local memory. 15 Is that your testimony? 16 A. So my testimony is that Wilt system can be 17 configured to process for this access. And that's 18 based on the totality of what I recall in view of 19 figures 2 and 3, at least the excerpt on column 7 of 20 Wilt. 21 Q. All right. Let's move on. Let's go back 22 to your declaration, which I've got up on the 23 screen. Exhibit 1. And at paragraph 88, you 24 address why you say Wilt discloses the -- discloses 25 claim 19 of the '381 patent; right?	60 1 context switching here is that one way you want to 2 switch context is because the task of a higher 3 priority comes in and you do the context swap, so 4 that the higher priority tasks runs ahead of the 5 lower priority or normal priority tasks. 6 So, to me, a system that's configured to 7 enable context swapping will also be capable of 8 configure to execute tasks based on different 9 priorities. 10 Q. What other reasons are there for context 11 swapping in a graphics processing environment 12 besides accommodating different priorities? 13 MR. JOHNSON: Objection. Form. 14 A. Priority is one of the main one. This 15 whole idea of preemption and so on when high 16 priority task comes in, you preempt a low priority, 17 normal priority task execution to bring this new 18 high priority task to the front and run it. So 19 that's one of the main use scenarios. 20 But there could be some other use -- 21 situations in which you do context swap. For 22 example, if a thread is running and gets to a point 23 in its execution that it needs certain data item 24 that's not available, then instead of waiting for 25 the data item to become available and therefore keep

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 16 of 182

Conducted on January 5, 2024

61 1 the processing unit in standby state doing nothing, 2 you context swap it. Bring another task in to run 3 to maximize utilization of the processing element 4 itself. So that's another common scenario. 5 That's scenario happens more often in CPUs 6 rather than GPUs, because GPUs by definition are 7 data intensive computing hardware. Which means that 8 all the data that you need, all the architecture has 9 been designed to be provided to you in a streaming 10 fashion so that we don't have to wait on anything. 11 So normally, you don't get that second scenario in a 12 GPU graphics processing task causing the context 13 swap. But, possible. 14 I mean, you could also use GPUs to run 15 general-purpose computing task. That's very well 16 known; right? This whole notion of GPGPUs; right? 17 General-purpose GPUs, machine learning applications 18 included and so on. So sometimes it is possible 19 that you run into that situation as well. 20 Q. What other reasons are there for context 21 swapping in a GPU? 22 A. Again, this thing about a CPU context 23 having multiple PPU context and then, to support 24 multiple PPU context with -- associated with the 25 single CPU context is another example that can be	63 1 you, Dr. Pedram. 2 THE WITNESS: Yes. 3 THE VIDEOGRAPHER: We are going off the 4 record. The time is 11:32 a.m. 5 (Recess.) 6 THE VIDEOGRAPHER: We are back on the 7 record. The time is 11:44 a.m. 8 BY MR. DOKHANCHY: 9 Q. All right. If we could get Exhibit 3 up, 10 please. 11 Dr. Pedram, your theory about the engines 12 in Wilt are that they are the work distribution unit 13 210, the front-end unit 212, and the core interface 14 unit 303; right? 15 A. Correct, sir. 16 Q. And -- 17 A. I'm sorry. Just in clarification, this is 18 one of our proposed identification, both the 19 engines; right? 20 Q. Right. 21 A. Because we have two different 22 alternatives. This is the primary one, correct, 23 yes. 24 Q. And the way -- I want to talk a little bit 25 about the way the alleged tasks flow through the
62 1 mentioned, and so on. 2 MR. JOHNSON: I had an objection. 3 A. Let me finish this. The point here is 4 that although there's a number of ways to do context 5 swapping. But this is a small, finite number of 6 ways. And one of the important well-known context 7 swapping scenarios is when you have the higher 8 priority task preempting an already running normal- 9 or low-priority task. 10 It is not like you have an infinite 11 possibility or a very large number of possibilities 12 and then -- this is just one of the few cases in 13 which you do context swaps generally. And certainly 14 context swaps due to priority ordering of the tasks 15 to be executed or running is one of the prominent 16 ones -- scenarios. 17 Q. So one of the most prominent scenarios. 18 What are the other prominent scenarios in a GPU that 19 would give rise to context swapping? 20 A. I mentioned two already to you. I'm not 21 sure if you want me to repeat that again. But I 22 mentioned already two, and that's all I can think of 23 at this time, so... 24 MR. DOKHANCHY: Let's take a short break. 25 We can just do five minutes if that's better for	64 1 PPUs. So the task will come in through host 2 interface 206 within the PPU. 3 Is that your opinion? 4 A. Yes, sir. 5 Q. And then, it will be passed to front-end 6 unit 212 here at line 61 of column 3; is that right? 7 A. Yes. 8 Q. And then, from front-end unit 212 at the 9 top of column 4, we see that the commands which you 10 allege are tasks are passed to work distribution 11 unit 210; is that correct? 12 A. Yes, sir. 13 MR. JOHNSON: Objection. 14 Q. And then, from work distribution 210, the 15 tasks are passed to the core 208; is that right? 16 A. Through the core interface 303. 17 Q. Okay. So that's what I want to talk 18 about. So column 4, lines 7 through 10 say, "cores 19 208 receive possessing tasks to be executed via a 20 work distribution unit 210." 21 Do you see that? 22 A. I'm sorry. We are on column 3 of the Wilt 23 reference? 24 Q. Column 4, line 7 through 10. 25 A. Yes. Yes, I see that. So what's the

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

Conducted on January 5, 2024

65 1 question? I'm sorry. 2 Q. The question is: That the cores 208 3 receive tasks from work distribution unit 210 as 4 stated here; correct? 5 A. I will answer yes. Please go ahead. 6 Q. Okay. And we can go down to column 8. We 7 see some description of core interface 303, for 8 example, starting at column 8, line 14. 9 Do you see that? 10 A. Yes, sir. 11 Q. And does anywhere here in -- let's take 12 column 8 -- describe that core interface 303 passes 13 or sends or transmits commands to core 208? 14 A. Yes, there is. 15 Q. Okay. Can you point me to that, please? 16 A. So from the very first sentence in column 17 8, starting line 14, "Operation of core 208 is 18 advantageously controlled via a core interface 303." 19 So even this by itself would imply that 20 commands are being sent to core 208 via this core 21 interface, because we know the function of the core 22 is to execute instructions in a thread that's 23 being -- that's running. 24 And so, if the operation of the core which 25 is to execute instructions is controlled via this	67 1 unit, to this core interface. We know data is 2 coming in also a lot. 3 And so, the data and the commands that 4 tell the core what to do is making its way to the 5 core, so that core can do its job. There is no 6 other possibility here;; right? So it's clear in 7 the first sentence of this excerpt. But also it is 8 inherent in the way the system works that the -- the 9 commands telling the core what -- how -- what kind 10 of processing to perform on the data is coming 11 through that same pathway, which is the core 12 interface 303, and makes its way to the executing 13 core, i.e., core 208. It's inherent. There is no 14 other possibility. 15 Q. So looking at Figure 2. And then, we'll 16 get to Figure 3. As we just discussed, the commands 17 come into the PPU by a host interface 206; they then 18 travel to front end 212; they then travel to work 19 distribution unit 210; they then travel to core 20 interface 303 in Figure 3; and then, they travel to 21 core 208. 22 Is that fair? 23 A. Yes, sir. 24 Q. Are these items -- host interface 206, 25 front end 212, work distribution 210, and core
66 1 core interface, obviously the core interface is 2 facilitating the transfer of instructions from what 3 is above it, which is the work distribution unit 210 4 to the core. So that by itself is enough in my -- 5 is enough of a disclosure. 6 But let's keep going. "In some 7 embodiments, core interface 303 receives data to be 8 processed (e.g., primitive data, vertex data, and/or 9 pixel data) as well as state parameters and commands 10 defining how the data is to be processed, e.g., what 11 programs to be executed from work distribution 12 unit 210." 13 So the -- at least the last part of this 14 sentence is saying that the core interface 303 is 15 receiving from work distribution unit and sending it 16 down, the commands, defining how data is to be 17 processed. 18 Q. Where is the "sending it down" part? I 19 see the reception. I don't see the sending it down. 20 Can you point me to that? 21 A. Yes. The operation of the core. We know 22 what the core does. The core is executing 23 instructions, these commands. The commands, we know 24 we can trace it down all the way from the front-end 25 unit, from the interface unit, to work distribution	68 1 interface 303 -- in parallel with one another? 2 A. No, sir, they are not. 3 Q. And as you alluded to earlier, you have 4 another interpretation of the engine -- excuse me, 5 the plurality of engines. And that is that 6 items 212, 210, and 303 in combination with one 7 another are a single engine; is that correct? 8 A. Yes, sir. 9 Q. Okay. Why would a POSITA consider these 10 three things to be one unit? 11 A. These are -- so I'm presenting here in my 12 declaration two alternatives that go hand to hand. 13 One alternative is, I say, the shader core. The 14 engines are the front-end unit 212, the work 15 distribution unit 210, the core interface 303 in one 16 of the PPUs. And then, the corresponding shader 17 core is the computational elements remaining in the 18 PPU, which is the collection of the -- the cores and 19 the memory interface and so, on. 20 So that's one alternative. And it makes 21 sense and performs -- the engines perform all the 22 required functions, which is to receive tasks from 23 the scheduling module and at the same time to load 24 the state data. And the cores do their own job to 25 receive the tasks from the -- one of the engines and

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 18 of 182

Conducted on January 5, 2024

69 1 then, execute them. Maybe two of them at the same 2 time. So that's -- that's one -- one sort of way 3 that Wilt reads on '381 claim, and therefore renders 4 it obvious in view of a person of skill in the art's 5 knowledge. 6 On the other hand, there's a second way 7 one could look at Wilt and define the collection of 8 these three units, that I'm not going to mention 9 again, as one engine. And then, in that case, you 10 have a plurality of engines because you have a 11 plurality of this collection of modules. And then, 12 the shader core is the plurality of the 13 computational parts of the plurality of PPUs in the 14 system. And that's, again, a valid read. And that, 15 again, renders obvious the asserted claim of the 16 '381 in view of a knowledge of a person skilled in 17 the art. 18 So why would a person skilled in the art 19 look at these three as one unit? This is one 20 construction. It's a box drawing thing saying that, 21 "Well, I can look at these three, and they actually 22 perform in combination in one of my proposed reads 23 of the Wilt reference in -- these three units 24 perform all the functions that are required by one 25 of the engines." But then there is another engine	71 1 So that's just one read. It's a valid 2 read, in my view. It's very well supported by the 3 Wilt reference and as it would have been understood 4 by a person skilled in the art. 5 Q. So let's set aside the '381. I'm not 6 asking about the '381. So my question is just about 7 a POSITA and Wilt. 8 I understand why the three items you 9 identified as engines might be considered three 10 engines. My question to you is: Why would a POSITA 11 looking at Wilt consider those three things together 12 one engine? 13 A. There is no mention of the term engine in 14 Wilt that I recall. Wilt describes its system. And 15 what I'm trying to do as an expert here is to show 16 how a person of skill in the art looking at Wilt 17 would have come up with a -- an understanding of the 18 way that Wilt's system works that would render 19 obvious whatever it is that '381 has claimed as its 20 invention. It's invalidating Wilt -- '381 in -- in 21 view of Wilt, because I can identify, by this box 22 drawing exercise that I've applied, a combination of 23 the front-end unit, the work distribution unit, and 24 the core interface 303 within one of the PPUs 25 asserting as one of the engines, claimed engines, of
70 1 that does the same thing. Another engine that does 2 the same thing for a different PPU. So I'm not sure 3 anything more than this is needed. 4 Q. Well, I -- 5 A. I -- 6 Q. I think you sort of touched on my 7 question, but, you know, let's start with the 8 discussion in Wilt. Wilt describes and depicts the 9 three items that you, on one hand, call engines as 10 different things. But Wilt never calls these three 11 things -- 210, 212, and 303 -- together any 12 particular name or certainly not an engine; right? 13 A. It gives the names. And each one of them 14 does its own specific function and facilitates the 15 transfer of the incoming tasks to the execution core 16 or cores in the system, at the same time loading the 17 state data. 18 So as I stated previously, in the second 19 construction presenting for how Wilt would 20 invalidate, in view of a person of skill in the art 21 at the time of the invention, the claim of -- or the 22 asserted claims of the '381, is to consider the 23 combination of these three units as one of the 24 engines with many other engines existing in parallel 25 across different PPUs.	72 1 the '381. 2 Q. So -- so again, I -- I'm not asking about 3 the '381. I'm asking about -- a POSITA looking at 4 the disclosure of Wilt sees the three items: 210, 5 212, and 303. 6 What rationale or what basis would such a 7 person have for considering those three things to be 8 one item? 9 A. It may not consider them to be an item, a 10 single item. It would consider them to perform the 11 functions of, one, receiving tasks to be performed, 12 and then, loading the state data associated with 13 those tasks, and then, passing the tasks down to the 14 cores. It doesn't know -- I mean, a person skilled 15 in the art would not even try to establish a 16 correspondence between these three units, the 17 combination of these three units, and the engine. 18 It would just know. 19 If you -- if you presented this to a 20 person skilled in the art and say, "I want to know, 21 does -- do these engines receive tasks?" 22 "Yes." 23 "Do they load state data?" 24 "Yes." 25 "Do they pass it down to the core?"

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 19 of 182

Conducted on January 5, 2024

73 1 "Yes." 2 That's all. They would not call it an 3 engine or anything. I mean, they just -- on the 4 times you asked them what is being claimed as the 5 function of the '381 engine and what is the 6 configuration of the '381 engines with respect to 7 other claimed components in '381, and if you ask the 8 right question of a person skilled in the art, you 9 would get the answer that, yes, yes, yes, and yes. 10 So in view of that, '381 is invalid. 11 I mean, a person skilled in the art 12 doesn't know about '381;; right? I mean, the '381 13 is not there yet. This is prior art to '381. 14 Q. You mentioned loading state data. Your 15 opinion is that core interface 303 loads state data; 16 correct? 17 A. That is correct, yes, sir. 18 Q. Front-end unit 212 does not load state 19 data; correct? 20 A. No, it does not. 21 Q. Work distribution unit 210 does not load 22 state data; correct? 23 A. (No audible response.) 24 Q. Did you say yes? 25 A. No, it does not. Work distribution unit	75 1 the texture data that you need to really apply to 2 the object that's being rendered on the user's 3 display device. 4 Q. Okay. So "handle" in this context means 5 what, in your view? 6 A. Some kind of pointer. 7 Q. And what's your view on the phrase "the 8 like"? What is included in "the like"? 9 A. When you try to do graphics processing, 10 there is usually a lot of transformations that you 11 need to apply to change coordinates to do maybe 12 rendering based on the point of view, to look at how 13 the light source or multiple light sources are 14 affecting the lighting conditions, and so, on. And 15 many of these transformations and operations require 16 a bunch of constants in the form of matrices or 17 vectors that get multiplied with the data that's 18 being provided by the application about the image or 19 the screen itself that needs to be rendered. So 20 those constants typically are a part of this "the 21 like"; right? 22 Q. Is there any sort of data associated with 23 a task that you would not consider to be state data 24 associated with that task? 25 MR. JOHNSON: Objection. Form.
74 1 does not. Sometimes I respond, but somehow my 2 responses don't make their way to you. So that's a 3 bit disconcerting, but... 4 Q. Yeah, I -- I didn't hear your answer, for 5 what that's worth. But so if I -- if I ask, it's -- 6 it's not to be duplicative. I just didn't hear. 7 A. No, I'm -- I'm apologizing. Saying that 8 if there is a bad connection on my side, I'm sorry. 9 Because in my view, I respond, but then I -- after a 10 long pause from you usually comes, "Did you 11 respond?" And in my view, I already did, but -- 12 so... 13 Q. Okay. 14 A. And I apologize if the connection is not 15 good on my side. 16 Q. That's all right. Thank you. 17 All right. Let's jump back to your 18 declaration. Paragraph 74, you talk about state 19 data. And you say, "State data refers to texture 20 handles, shader constants, transform matrices, or 21 the like associated with tasks"? 22 MR. JOHNSON: Objection. 23 Q. What are texture handles? 24 A. Sir, this is information about the 25 different textures. The pointers where you can find	76 1 A. So again, look at what we are trying to 2 do. So there is an image that I have a description 3 of in terms of a high-level programming language, a 4 graphics language, that says, "Well, there is this 5 car driving on this street and that's where the 6 traffic light is and that's where the sun is and 7 that's where the person in the image is that's 8 supposedly looking at the car coming at it." Try to 9 render this on some three-dimension -- some user's 10 screen that might be of different resolutions, 11 different sizes, has different color palettes, and 12 so, on and so, forth. And so, try to do the best 13 to do this rendering if it's a still image or maybe 14 if it's a video showing from one frame to one next 15 frame what happens. 16 And the data that's not part of this 17 rendering context is the data that describes the 18 original scene itself, like the -- the car, the fact 19 that there is a car, and the car has a particular 20 shape. How is it captured depending on what -- how 21 you -- if it's vector graphics or it's pixel 22 graphics or whatever, how -- I mean, there are 23 different representations of the same scene in 24 different formats. So that's the actual data that's 25 there. It's not -- so that's not part of the

Conducted on January 5, 2024

77 1 rendering context as such. 2 It's just the data that's there. The 3 rendering context would be these other things that I 4 need to be able to map this actual scene to the 5 three -- to the user's screen;; right? Which 6 includes, for example, transformation matrices that 7 allow me to do the coordinate transformations or to 8 perform appropriate lighting or shading in -- and 9 texture, handles and so, on;; right? 10 So -- so there are certainly data that's 11 not state parameters. 12 Q. Okay. Is state data about the state of a 13 task? 14 MR. JOHNSON: Objection. Form. 15 A. It could mean that. There is nothing that 16 excludes it. It could also mean or include that, 17 but it doesn't have to. 18 Q. So there is data that is not about the 19 state of a task that you would consider state data; 20 is that correct? 21 MR. JOHNSON: Same objection. 22 A. I'm not sure what you mean by "state of a 23 task." 24 Q. Have you ever come across the concept of 25 the state of a task in your experience?	79 1 constants, transform matrices, or the like) that is 2 locally stored in data storage units of the GPU. 3 The locally stored state data is called a context." 4 Sometimes, we will call it "rendering 5 context." So that's what I would normally go by;; 6 right? That's what it is. That's how '381 defined 7 it. 8 Q. Does -- withdrawn. 9 Is a primitive state data? 10 A. What do you mean by "primitive"? A 11 graphics primitive, such as a triangle or a polygon? 12 Is that what you mean? 13 Q. In the context of the '381 patent and in 14 the context of the Wilt reference and the other 15 references you've used in your declaration, what is 16 a -- 17 A. So let me -- 18 Q. -- is a primitive? 19 A. I understand the question, sir, yes. So 20 let me define a primitive to be a graphics 21 primitive, such as a line segment or a triangle or 22 maybe an -- a polygon. Then I would say that those 23 types of graphics primitives are not considered to 24 be state data, no. 25 Q. Okay. Are those types of graphics
78 1 MR. JOHNSON: Objection. Form. 2 A. I don't want to speculate, I'm sorry. You 3 are asking the question. Don't ask me to define 4 your question for you. What is the question you're 5 asking me when you say, "Do you -- is there data in 6 the state -- is there something in the state data 7 which is not about the state of task?" I'm saying 8 this is ambiguous question. I cannot answer it. I 9 don't know what you mean by "state of task," and 10 you're asking me to define state of task for you. 11 Your question, please be clear what you 12 mean. 13 Q. Sure. So the claim phrase is "state data 14 associated with each of the plurality of tasks"; 15 right? 16 A. Yes. 17 Q. And so,, state data is data regarding the 18 state. That seems correct; right? 19 A. No, I mean, I'm not sure where -- what 20 you're talking about. Because state data is data 21 about state. That's not a good way. So I look at 22 column 1 of '381 patent; right? Starting line 53, 23 maybe 54, "Each thread (e.g., shader program)" they 24 call it, "is typically associated with a set of 25 state data (such as texture handles, shader	80 1 primitives tasks? 2 A. No. A graphics primitive is an object, 3 line segment, triangle, and so, on. And you can 4 perform tasks on the primitives using the associated 5 state data rendering context information. But they, 6 by themselves and of themselves, are not to be 7 confused with tasks. 8 (Exhibit 4 marked.) 9 BY MR. DOKHANCHY: 10 Q. Let's introduce as Exhibit 4, Document 5. 11 And sorry, before we go into that line of 12 questioning, you also would not consider state data 13 in and of itself to be a task; correct? 14 A. With the same token, no, it's not a task. 15 It's part of the information that's needed typically 16 to perform a graphics processing task. 17 Q. Okay. Looking at the '381 patent, go down 18 to Figure 4. 19 A. Sir, is this a new exhibit you -- this is 20 the '381? 21 Q. This is the '381 patent, yes. So in 22 Figure 4. You see a number of virtual engines? 23 A. Yes, sir. 24 Q. And those are examples of the engines 25 recited in claim 15 of the '381 patent; correct?

Conducted on January 5, 2024

81 1 A. I believe so, yes. 2 Q. And you see that in Figure 4, each engine 3 is connected directly to a distinct command buffer 4 for each engine;; right? 5 A. I see that, yes, sir. 6 Q. And going back to Wilt, Figure 2 and 7 Figure 3, does Wilt describe here or elsewhere that 8 the engines you identified -- 210, 212, and 303, are 9 each connected to a distinct command buffer? 10 A. As you know, we have two different 11 alternative constructions for the set of engines in 12 Wilt. I assume you're talking about the first 13 construction, in which the engines are 212, 210, and 14 303; correct? 15 Q. Yes. 16 A. So for that alternative, no, there is no 17 disclosure in Figure 2 or 3 that each of the three 18 units that I mentioned -- 212, 210, and 303 -- are 19 independently connected to a buffer, a command 20 buffer. It's not there. Only the first one, which 21 is the front-end unit 212, is connected to a host 22 interface 206, because it receives information from 23 there. We know that. And that host interface 206 24 most likely has buffers inside of it. 25 Q. Okay. And so, you say "most likely." Do	83 1 A. It's not called a command buffer, of 2 course, in the Wilt reference. But I would say 3 based on the explanation I already provided, the 4 host interface 206 will have some kind of buffer, a 5 single-entry buffer, dual-entry buffer, or more 6 generally a queue as part of its general design. 7 And because each PPU has its own host interface 206, 8 as is clearly seen in Figure 2 or 3, then one would 9 say in that case there is a host interface 10 associated with each PPU. And hence, there is a 11 buffer associated with the -- each of the engines 12 under that second alternative. 13 Q. Would there be any other alleged command 14 buffer other than the one you are saying may be in 15 host interface 206 that would exist in that second 16 read that we've been discussing? 17 A. Again, the details of how the work 18 distribution unit is implemented in hardware or what 19 the core interface 303 implementation is are 20 missing. But a person skilled in the art looking at 21 a synchronous design would understand that buffers 22 are prevalent in all kinds of components to make 23 sure that you don't get write-after-write hazards, 24 you don't run into race conditions, you hold values 25 long enough for the consumers of the values to make
82 1 you know whether Wilt describes host interface 206 2 as containing a command buffer? 3 A. I don't recall that. But again, a person 4 skilled in the art, if you tell them there is a host 5 interface, a gateway, that brings data, state 6 parameters, rendering context information, whatever, 7 to a front-end unit, and then, -- obviously brings 8 it, it must have a buffer to temporarily store it so 9 that the front-end unit could look at it and make 10 use of it. Otherwise, we run into all kinds of 11 hazards. Bad situation in the system that would 12 cause it to fail. So host interfaces typically 13 would have buffers, queues inside of them to hold 14 the values for some period of time before new values 15 come in. 16 That's very common. I can't think of any 17 host interface that would not have that kind of 18 buffer inside of it, but again... 19 Q. And what about your second interpretation 20 of the engines, which as I understand it -- correct 21 me if I'm wrong -- is that front-end unit 212, 210, 22 core interface 303 of a single PPU are one engine. 23 And those equivalent components in the other PPUs 24 are other engines. Do those engines have a distinct 25 command buffer for each engine?	84 1 use of them before you overwrite them. And because 2 of this, I expect and I think any person skilled in 3 the art looking at this would expect, for example, 4 there are buffers inside the front-end unit itself 5 and there is buffer inside the Wilt distribution 6 unit and there is buffer inside the core. The 7 buffer may just be a single register that holds the 8 value, but still it's there because this is 9 synchronous design. 10 If you try to push data from the host 11 interface to the core without any buffer inserted on 12 the -- in the front-end unit, or work distribution 13 unit, or the core, you will be definitely running a 14 very slow, very haphazard operating system that 15 would crash, almost guaranteed, all the time because 16 bad things will happen to such a system. That's one 17 answer. 18 The other answer is at least we know for 19 sure the work distribution unit would have buffers 20 inside of it, because work distribution unit holds 21 some of the tasks if none of the cores is ready to 22 run. So it must have a buffer to hold the task that 23 cannot be dispatched to the cores because all of the 24 cores are busy. Otherwise, you get the situation in 25 which tasks are coming in, and they come to the work

Conducted on January 5, 2024

85 1 distribution unit, no core can run them, and they 2 drop off and disappear. And that's the worst thing 3 that happens;; right? A task came in and you had to 4 drop it because you have no way to hold it. 5 That's not a good design. No one would do 6 it. This is an erroneous design, and no one will 7 design a system like that. 8 Q. What do you mean by Wilt is a synchronous 9 design? 10 A. It means that there is a clock that -- 11 which takes -- dictate how data progresses according 12 to a well-defined control flow in the overall 13 pipeline. I mean, this is high-performance computer 14 graphics system, and these systems are -- from 15 NVIDIA, from AMD, from anyone who does it almost -- 16 are synchronous designs. It's well known. 17 I mean, I don't know of any NVIDIA system 18 that was a graphics processing system working 19 asynchronously without a clock. You have one or 20 potentially more than one clock, but at least you 21 have one clock. All systems I've seen from AMD or 22 NVIDIA that do graphics processing are clock 23 systems. And back then -- definitely now, 24 definitely -- I don't know of anything that is 25 not --	87 1 work distribution unit 210 and cores 208; is that 2 right? 3 A. It's not shown in Figure 2, but it is 4 shown, as you know, in Figure 3;; right? According 5 to page 303, sitting between the work distribution 6 unit 210 and core 208. 7 Q. So this PPU 0 will have a core interface 8 303 within it; right? 9 Is that correct? 10 A. Yes, sir, it is. Yes, sir. 11 Q. And in Figure 2, the details of PPU 1 and 12 other PPUs are not shown. But they are assumed to 13 be structured in a similar way to PPU 0; is that 14 right? 15 A. Yes, sir, that is correct. 16 Q. And so,, those other PPUs will have their 17 own cores 208 that are distinct from the cores 18 within PPU 0; is that right? 19 A. Yes, sir, that is correct. 20 Q. And the other PPUs will have core 21 interfaces 303 that are distinct from the core 22 interface 303 and PPU 0; is that right? 23 A. Yes, sir, that is correct. 24 Q. And the core interface 303 within PPU 0 25 only sends to the cores 208 within PPU 0; is that
86 1 Q. In -- 2 A. -- clock. 3 Q. In Wilt, is the task transmission 4 necessarily synchronous? 5 A. Again, that's a very different question -- 6 MR. JOHNSON: Objection. Form. 7 A. -- than what I meant by synchronous 8 design. You could have a synchronous design using 9 clocks -- clock element, clock signal that serves as 10 the synchronization mechanism in the system. That's 11 a low-level mechanism that's needed for correct 12 operation of the circuit and high-performance 13 design. And that is not incompatible with whether 14 the dispatch of the tasks is done in a synchronous 15 fashion or asynchronous fashion. So do not confuse 16 the asynchronous dispatch statement or requirement 17 or feature of a system with the fact that the system 18 is using clocks as a low-level synchronization 19 mechanism. They are independent. They have nothing 20 to do with each other. They are two different 21 concepts. 22 Q. And so, part of your rationale for why -- 23 well, let me take that back. Withdrawn. 24 In Figure 2, I think you said although 25 it's not shown, core interface 303 will sit between	88 1 right? 2 A. Yes, sir, that is correct. 3 Q. And when I say "send," do you understand I 4 mean send the commands that you are saying are 5 tasks; right? 6 A. Yes, sir. That's what I meant. So thank 7 you for the clarification. 8 Q. Yes? 9 A. But of course not only that;; right? We 10 know there's also state data and so, on, but those 11 included, yes. Those tasks included. 12 MR. DOKHANCHY: So I'm doing to transition 13 to another reference. I don't know, Dr. Pedram, do 14 you want to take a lunch break now? We can take a 15 short one, or do you want to keep going? 16 THE WITNESS: I can keep going. I rather 17 keep going. Again, I'm trying to. But depends on 18 you guys, and videographer, and the others. I'm 19 fine either way. 20 MR. DOKHANCHY: Why don't we keep going 21 for a little bit and -- maybe another 20 minutes or 22 so. 23 All right. Let's introduce Document 8 as 24 Exhibit 5, please. 25 (Exhibit 5 marked.)

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

Conducted on January 5, 2024

89 1 MR. JOHNSON: I don't mind if we keep 2 going but I do need -- I do need a five minute 3 break. 4 MR. DOKHANCHY: Why don't we go off the 5 record? We'll discuss the duration of the breaks. 6 THE VIDEOGRAPHER: We're going off the 7 record. The time on the video monitor is 12:37 p.m. 8 (Recess.) 9 THE VIDEOGRAPHER: We are back on the 10 record. The time on the video monitor is 1:10 p.m. 11 BY MR. DOKHANCHY: 12 Q. Okay. If we could please get up 13 Exhibit 5. 14 Dr. Pedram, Exhibit 5 is the Jiao 15 reference that you rely on in your declaration; is 16 that right? 17 A. Yes, sir. 18 Q. I'll scroll us down just to -- in Jiao, 19 the items that you allege constitute the engines of 20 the '381 patent are the vertex shader 304, geometry 21 shader 306, and pixel shader 310; is that correct? 22 A. Yes, sir. 23 Q. And I may refer to those as "the shaders" 24 or "the graphic shaders," so is that understood 25 that's what I'm referring to?	91 1 A. Yes. Obviously, there should be some 2 other information because the vertex shader could 3 perform in a reconfigure manner a number of 4 different operations. Some examples are given here; 5 skinning lighting, and so, on. And so,, there 6 should be an instruction also received by the vertex 7 shader that tells it what to do. 8 Otherwise, does it do skinning or does it 9 do lighting? Later on when we describe other -- 10 other shaders, there are a lot of other operations 11 that it could perform. So each one of them should 12 know what it is that they are supposed to do. This 13 is not a -- a -- a fixed function. It is a 14 reconfigurable functionality that's provided by the 15 shader and therefore, there must be instructions, 16 i.e., tasks received by these shaders to tell it 17 what to do. 18 Q. When you say, "there must be tasks passed 19 to the shader to tell it what to do," is there any 20 part of Jiao that you think explicitly discloses 21 passing tasks to shaders? 22 A. Yes. The EU pool control and 23 vertex/stream cache, which is really the main 24 scheduler here. So shown in Figure 2 but also 25 placed as the top entry point in Figure 3, is the
90 1 A. (No audible response.) 2 Q. Sorry. Is that a yes? 3 A. Yes. 4 Q. Thank you. All right. 5 In column 6 of Jiao, at line 20, Jiao 6 states, "The primitives, once assembled, are passed 7 to the vertex shader 304. The vertex shader 304, as 8 is known, processes vertices by performing 9 operations such as transformations, skinning, and 10 lighting." 11 Do you see that? 12 A. Yes, sir, I do. 13 Q. As we discussed before, those primitives 14 are not tasks; correct? 15 A. Primitives are not tasks, correct, yes. 16 Q. And that is the -- withdrawn. 17 Those primitives are the vertex 18 information that Jiao describes as being passed to 19 the vertex shader in order to allow the vertex 20 shader to perform operations such as 21 transformations, skinning, and lighting; correct? 22 A. Yes, sir. 23 Q. Is there any other information described 24 as being passed to the vertex shader to allow the 25 vertex shader to perform vertex processes?	92 1 component that's providing these tasks through the 2 -- a set of components modules which we have 3 identified as vertex shader, geometry shader, and 4 pixel shader; i.e., the engines. 5 So therefore, it is there that Jiao 6 specifically disclosing that tasks are being passed 7 down to the shaders. 8 Q. So what I'm trying to get a sense for is 9 are you more relying on an understanding that 10 something must be done or an explicit description 11 that the EU pool control passes tasks to the 12 shaders? 13 A. There's certainly disclosure in Jiao that, 14 through this EU pool control unit, tasks are being 15 passed down to the vertex shader, geometry shader, 16 and pixel shader to perform their required 17 functionality, which is reconfigurable and one out 18 of many for each one of them. So that selection of 19 exactly what to do on a given set of primitives 20 coming in, is what the EU pool control is doing. 21 Of course, instruction's really coming 22 from the driver software that sits outside, through 23 the API, writing into registers in the GPU system. 24 So to me, that's very clear disclosure. It's not 25 just a matter of it must be. It is that way.

Conducted on January 5, 2024

93 1 Q. So your opinion is that the commands -- 2 I'll just say -- withdrawn. 3 Your opinion is that tasks come in from 4 the driver software to the EU pool control. And 5 from the EU pool control, tasks are sent to the 6 shaders; correct? 7 A. As you know, the EU pool control contains 8 circuitry that includes also a cache but also the 9 control unit. And this control unit is 10 orchestrating the transfer of tasks from the 11 software driver, through the API, to the graphics 12 pipeline that goes through the input assembler, the 13 vertex shader, the geometry shader, the rasterizer, 14 the pixel shader and, finally, generates these set 15 of threads that get executed on the shader core, the 16 computational core of Jiao. 17 Q. So I just want to distinguish two concepts 18 and ask you for some clarification. So we've looked 19 at -- in column 6, line 20 -- how primitives and 20 constants are sent to the vertex shader. 21 My question to you is: Can you point me 22 to any disclosure like that that says that commands, 23 instructions, tasks, or anything like that are sent 24 to the pixel shader or vertex shader? 25 MR. JOHNSON: Objection. Form.	95 1 The first sentence and rest of that description, any 2 person skilled in the art looking at that 3 description -- this is not only my opinion, but also 4 that's ALJ's opinion in their initial determination. 5 And that is also the PTAB's decision when they 6 instituted the -- these proceedings, that, in fact,, 7 these components the vertex shader the, graphics 8 shader -- I'm sorry -- the geometry shader and the 9 pixel shader receive some tasks and do some 10 processing on them to generate newer tasks that 11 finally get routed to the execution unit pool to 12 complete the overall graphics processing task in the 13 application. 14 Q. So we looked at, in column 7 -- excuse 15 me -- column 6, a very clear explanation in Jiao of 16 how primitives are sent to the vertex shader 304, 17 specifically. My question to you is: Can you point 18 me to a citation that says in that level of 19 specificity that instructions, commands, or tasks 20 are passed the vertex shader 304? 21 MR. JOHNSON: Objection. Form. 22 A. Yeah. Let me read this for the record. 23 Also for you. For example, I'm just -- I picked one 24 of the three shaders, geometry shader 306. On 25 column 6, starting line 29, I'm reading, "The
94 1 A. I understand the question. 2 MR. JOHNSON: Objection. Form. 3 A. I think I understand the question. So top 4 of page -- sorry -- column 6 of Jiao says, "Figure 3 5 is a block diagram that illustrates an embodiment of 6 the logical graphics pipeline 208." And as part of 7 that logical graphics pipeline, which means it is a 8 pipeline that's doing at least a portion of the 9 graphics processing, the tasks come down as shown in 10 Figure 3. They go through different processing 11 steps. Each one of them is reconfigurable and one 12 of many and, finally, make their way to the 13 computational core of the Jiao system, which is the 14 execution unit pool as shown also in Figure 4 of 15 Jiao. 16 So to me, that's very clear expression 17 disclosure that it is tasks that is flowing or is 18 the set of tasks that's flowing down with the aid or 19 facilitation of the EU pool control unit from the 20 API software driver, through the API, to that 21 logical graphics pipeline. 22 Q. Which column and line number are you 23 referring to? 24 A. I would say column 6 itself. When you 25 start with column 6, "Figure 3 is a block diagram."	96 1 geometry shader 306 receives, as inputs, vertices 2 for a full primitive, and is capable of outputting 3 multiple vertices that form a single topology, such 4 as a triangle strip, a line strip, point list, et 5 cetera. The geometry shader 306 also receives 6 constants from the constant buffer of the constant 7 buffer system, and may be further configured to 8 perform various algorithms, such as tessellation, 9 shadow volume generation, et cetera. The geometry 10 shader 306 then outputs information to a rasterizer 11 308" -- and it continues. 12 So obviously, a skilled person in the art 13 looking at this would realize that this geometry 14 shader receiving the inputs that it needs, receiving 15 the right bunch of constant that it needs. There 16 are lots of constants in the world. Which constants 17 do you access? The constants that the application 18 program, the task itself, is telling you are the 19 right constants to use. 20 It receives the data. It receives the 21 right set of contextual information it needs. The 22 constants that it needs. It is told what algorithm 23 to run tessellation, shadow volume generation, or 24 what have you, does the right thing, and goes out. 25 And that's very clear description that these are

Conducted on January 5, 2024

97 1 tasks being performed by that, and the tasks are 2 received from the next element higher up in that 3 graphics pipeline. 4 It's a pipeline. What flows on the 5 pipeline? Task is flowing on the pipeline. So 6 another way of saying it would be looking at -- we 7 know this logical graphics pipeline end in through 8 Figure 4, for example;; right? 9 So it ends in the access to the EU pool 10 414 that does the work. This is doing the work -- 11 the final work that needs to be done to complete the 12 graphics processing task. There are tasks there. 13 There are tasks. There must be tasks. Otherwise, 14 what are they doing? There must be instruction 15 streams. 16 The only way to make your way to the 17 execution unit is through these vertex shader, 18 geometry shader, pixel shader. There's no other 19 path. 20 If we know at the end the execution unit 21 pool is doing the tasks, there's no disagreement 22 there -- or threads -- 23 Q. Let me clarify -- 24 A. I'll finish quickly. Therefore the tasks 25 are coming through that logical graphical pipeline,	99 1 instruction is being provided from the software 2 driver with the help of the EU pool controlling it. 3 What other possibility exists? None. 4 Q. That's -- that's what I'm asking. I asked 5 you at the outset, are you saying must be done or 6 something is described as being done? So let me -- 7 with that context, let me re-ask my question. 8 Column 6 describes the geometry shader 9 receiving constants, receiving primitives, and 10 performing tasks. Where does it do the next level 11 of describing that the geometry shader not only 12 performs tasks, but receives tasks, just like it 13 receives constants and primitives? 14 A. It is there in Figure 3; right? It's 15 there in the whole description in paragraph 6. 16 Tasks are being performed. Where did they come 17 from? Where they are -- to me, it's clear. It's 18 expressly stated. This is a logical graphics 19 pipeline. Tasks are flowing from one end and 20 falling off of the other end of this logical 21 graphics pipeline. 22 There is no other possibility. And it's 23 not a matter of it must be. It is that way. 24 There's no other reading. No person skilled in the 25 art looking at this would come up with any other
98 1 and it's there. It's very express disclosure, in my 2 view. 3 Q. Okay. So let me take it piece by piece, 4 because that was very long answer, and that didn't 5 really address what I'm talking about so let me -- 6 let's just go slowly through this. 7 You are saying the shaders perform tasks; 8 right? We agree with that? 9 A. Yes. 10 Q. Okay. So not disputing that they perform 11 tasks, my question to you is: In what you just 12 quoted to me it is stated that the geometry shader 13 receives primitives and constants. It does not 14 state that the geometry shader receives tasks. 15 Where do you see the geometry shader receiving -- 16 not just performing -- receiving tasks? 17 A. It is really expressed that it is 18 performing tasks;; right? What do you mean it is 19 not performing tasks? It is performing tasks. 20 There are examples of tasks performed or given as 21 algorithm, such as tessellation by the geometry 22 shader. That is a task being performed and one of 23 many possible tasks being performed. 24 So there is an instruction that's telling 25 it, "Perform this task versus the other task." That	100 1 conclusion. The tasks are being received by the 2 different shaders. And path -- some processing is 3 done. New tasks are generated and sent down. 4 Q. I understand. So are you -- is it your 5 opinion that it is impossible that a configuration 6 of the shaders comes from an entity other than a 7 scheduling module? 8 MR. JOHNSON: Objection. 9 A. I do not understand what you mean by the 10 "configuration of the shaders." Can you elaborate 11 what that means to you? 12 Q. You pointed to text saying that the 13 shaders can be configured -- can be reconfigured. 14 My question to you is that the claims say that tasks 15 must be received by the engines from a scheduling 16 module. Given that, is it impossible that the 17 configuration that you're pointing to of the shaders 18 comes from an entity other than a scheduling module? 19 MR. JOHNSON: Objection. Form. 20 A. And for the record, the scheduling module 21 that -- that we -- that I'm identifying here is the 22 EU pool control unit that's causing or helping the 23 software driver program running on the CPU, sort of 24 sending the task through this graphical pipeline so 25 your question is...

Conducted on January 5, 2024

101 1 Q. Well, let's start with that. Where does 2 Jiao state that the EU pool control is what 3 configures the operation of the shaders? 4 A. Let me look at this. 5 THE VIDEOGRAPHER: And, Doctor, when you 6 give a response, if you would just be a little more 7 centered -- or if you could turn the camera towards 8 your face. You're leaning really far out. We're 9 not able to see you when you're leaning over to read 10 off of that other monitor. 11 A. Yeah. But then I will block the monitor. 12 I cannot read off of it. 13 MR. DOKHANCHY: Jerry, I think we're not 14 extremely concerned with that. Let's let Dr. Pedram 15 do what he needs to do. 16 THE VIDEOGRAPHER: Okay. 17 A. Let's refer to my own declaration. 18 Q. Okay. 19 MR. DOKHANCHY: That's fine with me. 20 MR. JOHNSON: Fine here. 21 THE VIDEOGRAPHER: We are going off the 22 record. The time is 1:36 p.m. 23 (Recess.) 24 THE VIDEOGRAPHER: We are back on the 25 record. The time on the video monitor is 1:38 p.m.	103 1 graphics processing tasks. So the controller for 2 the respective inflow and outflow of the tasks and 3 the required data for these tasks is this EU pool 4 control unit. That's one reference. 5 The other one... 6 It says -- on column 8, line 21, says 7 "Thus, the EU pool 414, at its peak, can process 8 multiple threads simultaneously or substantially 9 simultaneously." So it is processing threads 10 through the shaders and eventually through the 11 shader core. 12 Then another reference is... 13 Q. Let's address the ones that you -- 14 A. No. Let me finish this, because then I 15 have to start over again. So I'll just give you the 16 three or four things that I find now, and it's also 17 in my report. And then, you can address them one at 18 a time, please. 19 Q. Okay. 20 A. So another one is bottom of column 8, 21 starting line 65. "The execution unit flow of the 22 EU pool 414 generally comprises several levels, 23 including rendering context level, thread or task 24 level, and an instruction for execution level. At 25 any given time, there may be multiple (e.g., two are
102 1 MR. DOKHANCHY: I'm not sure if we have 2 counsel back. Jeffrey, are you there? Okay. 3 You're muted but looks like you're back. 4 MR. JOHNSON: Yeah, sorry. 5 MR. DOKHANCHY: No problem. 6 BY MR. DOKHANCHY: 7 Q. All right. Dr. Pedram, looks like you've 8 had a chance to review your report. 9 A. Multiple disclosures in Wilt that show 10 that this EU pool unit serves as the global 11 scheduler of tasks that assigns these tasks to the 12 graphics logical -- the logical graphics pipeline, 13 including the shaders. Then, also, to threads that 14 get executed on the computational core, the shader 15 core itself. And for example, if I were to read 16 into record -- 17 Q. Column and line number, please. 18 A. Let me get there. Because I'm not taking 19 notes, I'm just... 20 So for example -- there are multiple of 21 these -- but if you look at column 7, line 26 and 7, 22 it says, "In general, the EU pool control unit 206 23 controls the respective inflow and outflow to the 24 computational core 204." Computational core is the 25 one that's performing all the graphics tasks --	104 1 described as a non-limiting example in the present 2 disclosure) rendering contexts allowing [sic] in 3 each execution unit 420, with the contexts 4 identified by the use of one bit flag or other 5 mechanisms. The rendering context check (or 6 rendering context information) is passed from the EU 7 pool control unit 206 before tasks belonging to this 8 context are commenced. The rendering context 9 information may include shader type, number of 10 input/output registers, instruction starting 11 address, output mapping table, vertex identifier, 12 and constants in a respective constant buffer. Each 13 execution unit 420 of the EU pool control 414 may 14 store a plurality of tasks or threads (e.g., in some 15 embodiments, 32 threads) at the same time. In one 16 embodiment, each thread fetches an instruction 17 according to a program counter." 18 Continues on, column 9, line 15. "The EU 19 pool control unit 206 functions as a global 20 scheduler for the tasks and assigns appropriate 21 threads in the execution units 420 using a 22 data-driven approach (e.g., vertex, pixel, geometry 23 packets in the input). For instance"-- 24 Q. Dr. Pedram, I'll stop you there because 25 it's taking too long to read all this into the

Conducted on January 5, 2024

105 1 record. Let's address what you've pointed to so 2 far. And then, if there's more, there's more. But 3 it's just becoming a little too time consuming. 4 So let's take that last sentence that you 5 read. "The EU pool control unit functions as a 6 global scheduler for the tasks and assigns 7 appropriate threads in the execution units 420." 8 Okay. 9 Execution unit 420 are what you call the 10 shader core; correct? 11 A. Yes, sir. 12 Q. Those are not the engines; correct? 13 A. They are not the engines, no. 14 Q. Is there anything that you read me or 15 anything else that says something like that with 16 respect to the engines? That is: The EU pool 17 control sends tasks or instructions to the shaders? 18 A. Yes. 19 Q. Okay. 20 A. It is. For example, we -- the lines above 21 it on the same column 9, when it talks about the 22 execution unit flow of the EU pool 414 data, 23 describe that there are several levels: Rendering 24 context level, thread or task level, instruction or 25 execution level. When the EU pool control that	107 1 pool control unit. 2 And when the Jiao reference says this is 3 the global scheduler, it doesn't mean it's a global 4 scheduler only at one level. It's a global 5 scheduler at multiple levels. And those multiple 6 levels do include the level in which this logical 7 graphical -- graphics pipeline is also operating. 8 That's how a person skilled in the art would 9 understand it, and that's how it is. 10 If you go back to the other ones I quoted 11 also, it does mention that the EU pool control unit 12 in addition to assembling the inputs and so on, is 13 actually telling the appropriate shader what to do, 14 or otherwise -- 15 Q. Where -- 16 A. -- what does the shader do? 17 Q. Where does it say that, Dr. Pedram? 18 A. I gave you three references. Please 19 mention the three again to me. I don't take notes, 20 so... 21 Q. Sure. The first one was 7, 26. 7, 26 22 says the EU pool control controls the inflow and 23 outflow to the core 204. So let's go look at -- 24 let's look at that. 25 A. Okay.
106 1 comes in to provide the threads into thread slots of 2 the different elements in the execution unit pool, 3 it is dealing with instruction or execution level. 4 But then there is the thread or task level that 5 happens in the -- for example, graphics, the logical 6 graphics pipeline. And EU pool control is the -- is 7 the orchestrator of that as well. 8 It did explain that the rendering context 9 information includes shader type, number of 10 input/output registers, instruction starting 11 address. Why are my -- why is the EU pool control 12 unit providing the instruction starting address? 13 Because there are instructions there to be executed. 14 Q. And what unit is it sending that to? It's 15 execution unit 420; right? Where does it talk about 16 the shaders? 17 A. So the -- the point here is that the EU 18 pool control unit has operated at three different 19 levels. At the lowest level, it's dealing with the 20 EU pool execution unit, execution unit pool. And so 21 there is a function that it performs there. But 22 there is also the other function that it performs as 23 it controls the flow of the same tasks down through 24 the logical graphics pipeline, to then eventually 25 make its way after appropriate processing to the EU	108 1 Q. Core 204 includes -- 2 A. It's the first (phonetic) processing. 3 It's the whole -- 4 Q. It's a number of things; right? This -- 5 A. Yes. 6 Q. Later, it clarifies that 206 inputs 7 instructions into the EU pool into items 420. 8 What I'm looking for, Dr. Pedram, is any 9 clear statement that 206 puts instructions or gives 10 instructions to the shaders. Not the core, not to 11 item 204 generally, but to the shaders. And you 12 have pointed me to nothing that says that EU pool 13 control issues commands to the shaders. 14 The closest you've come is item 204 and 15 the things that you call the shader core 420. But 16 there has been no discussion of these shaders 17 receiving instructions. 18 And so that's the specific thing I'm 19 looking for. And if all that you can point to is 20 what you've pointed me to, that's fine. We can move 21 on. But I want to know if you see any discussion in 22 Jiao of these shaders receiving instructions. Not 23 204 receiving instructions. Not 420 receiving 24 instructions. 304, 306, and 310 receiving 25 instructions.

Conducted on January 5, 2024

109 1 A. Yes. To me, there is -- and to a person 2 skilled in the art, there is disclosure in Jiao that 3 the EU pool control unit also provides instructions 4 to the various engines in the logical graphics 5 pipeline of 303. 6 This is seen by the linear structure of 7 that: Tasks come from the top; they make their way 8 through these different shaders; they get passed to 9 the execution limit to be processed therein; and 10 then, there goes the output. 11 So it's there in Figure 3. It's also 12 there in the three-level operation of the -- this EU 13 pool control that deals with both at the instruction 14 level, at the very lowest level. But at -- also at 15 the task level, which is this intermediate level, 16 working with the graphics shaders. 17 There is the understanding -- a clear 18 description, rather than understanding, that these 19 shaders can be programmed. They can be configured 20 to do different algorithms. And that is an 21 indication that an instruction is provided to them 22 by the EU pool control unit that says, "Do this 23 versus the other thing," as long as they don't know 24 what to do. So to me, that's very clear, sir. But 25 I have nothing further to add at this time.	111 1 of the EU pool 414, are the low-level tasks 2 generated by these shaders. The shaders themselves 3 receive higher-level tasks coming to them through 4 the EU pool control unit as part of the logical 5 graphics pipeline. There is no other way. There is 6 a software driver that describes a graphics 7 processing application or task to be performed. 8 That gets translated through multiple 9 levels of subsequent processing into other 10 lower-level tasks. And the fact that some 11 intermediate level of this processing generates 12 lower-level task doesn't mean that it did not have 13 as input the higher-level tasks that corresponded to 14 the graphics processing application running as -- in 15 the user space. There is no inconsistency here. 16 Anything other than this interpretation is 17 a nonsensical interpretation, because then it says, 18 "A task is running in the user space and never makes 19 it to the execution unit, but suddenly the results 20 come out right." 21 No. There is no magic here. This is a 22 deterministic system. It is an engineered system. 23 It's not a magical system, sir. 24 Q. So there is no way that the shaders could 25 be configured or programmed to do what they need to
110 1 Q. In column 9, it states at line 35 to 41, 2 "As various shader tasks are generated by the vertex 3 shader, geometry shader, and pixel shader 310, they 4 are delivered to respective execution units 420 to 5 be carried out. 6 As individual tasks are generated, the EU 7 pool control 206 handles the assigning of those 8 tasks to available threads within the various 9 execution units 420." 10 And so I want to go back up to Figure 4. 11 Those tasks that are being described, do you agree, 12 Dr. Pedram, are being generated by these items 304, 13 306, and 310, and being directed to the items 420 by 14 the EU pool control? That's what's being described; 15 correct? 16 A. Not entirely. What is being described is, 17 in fact, there are some low-level tasks generated by 18 the said shaders in response to processing of 19 higher-level tasks coming through the logical 20 graphical pipeline to these shaders. Again, there 21 are different levels of tasks. The tasks goes 22 through different processing, being transformed to 23 other tasks. 24 And so although it is true that what 25 drives the execution unit 420a, c, and so on as part	112 1 do other than receiving tasks from a scheduling 2 module; is that your position? 3 A. Wilt describes a general graphics 4 processing system capable of servicing almost any 5 graphics applications at the time that the system 6 was envisioned and constructed. It cannot limit 7 itself to only a small or -- or just a -- one way of 8 doing things. 9 The fact that there is clear, express 10 disclosure in Jiao that these shaders are 11 reconfigurable to perform any of a number of given 12 applications means that the application itself at 13 the highest level is telling them what configuration 14 to assume to provide the required functionality. 15 And that's what it means by them actually running 16 tasks. The application generates high-level task 17 that, as they way -- they make their way through the 18 logical graphical pipeline all the way to the 19 execution unit pool, they are providing information 20 to various components and modules in the systems as 21 to what to do. And what to do is done through an 22 instruction decoder, of sorts, that says, "This is 23 what I want you to do." And that's what the meaning 24 of "instruction stream" is. And that's what the 25 meaning of a task is in this context, sir.

Conducted on January 5, 2024

113 1 And this is not my only -- my -- only my 2 opinion. ALJ agreed to this. PTAB agreed to this. 3 Everyone sees that, in fact, tasks -- there are 4 high-level tasks, intermediate-level tasks, 5 low-level tasks, and that they just flow down. And 6 no one found any inconsistency in the statement that 7 the shaders generate lower-level tasks that get 8 their way to EUP. Because everyone understood -- 9 and a person skilled in the art will understand -- 10 that these tasks are generated in response to 11 higher-level tasks coming to them. 12 Q. Let's turn to your opinion on claim 19 as 13 to Jiao. And claim 19 requires what are called 14 "general-compute tasks." 15 Do you understand that? 16 A. Yes. 17 Q. What is your opinion on what a 18 general-compute task is? 19 A. Something other than a graphics processing 20 task, I would say. In the context of the -- Jiao 21 and the context of '381, a task that involves 22 standard fetch and decode, standard execution, 23 write-back to memory, load/store, ALU operations, 24 addition, multiplication, shift, so on, these are 25 all general-compute tasks.	115 1 13 as an example of general-compute tasks. 2 A. 6, 1 to 48. So the first reference I have 3 is 6, 1 to 48. 4 Q. I'll come back to that one. I want to do 5 this one first. So here, you point to this 6 paragraph in Jiao discussing instruction, decoding, 7 operand fetch, et cetera. 8 Do you see that? 9 A. Yes. 10 Q. What does -- withdrawn. 11 What components of Jiao are described as 12 performing these functions? 13 A. I mean, there is express disclosure here 14 that in the -- in column 11, starting line 4 -- that 15 this execution unit data path, EUDP 514, comprises 16 arithmetic logic units. These are units that 17 perform general computation. Everyone knows that; 18 right? 19 So any CPU that -- even the very first CPU 20 that was ever built, before there is even the notion 21 of graphics processing present, the very first 22 general purpose CPU that was built includes ALU, 23 arithmetic logic unit. 24 So a component that has ALU can do 25 general-purpose computing. And other logic
114 1 Q. What is -- setting aside the word 2 "general," what is a compute task? 3 A. In the context of the '381 and the prior 4 art references, it's never been defined. As such, 5 there is no need to define it. It's just as a plain 6 and ordinary meaning of the term, which would -- 7 which would be a task that involves computation. 8 Q. And what do you mean by "computation"? 9 A. For example, addition, subtraction, 10 shifting, exponentiation. Maybe some transcendental 11 functions, trigonometry functions, whatever; right? 12 I mean, it's general computational tasks. 13 Q. And when you say "shifting," do you mean 14 some sort of mathematical shifting? 15 A. Yeah, shift, as you know, shift right or 16 shift left to divide by two or multiply by two, 17 because of the binary number system representation 18 that's used in these systems. So normally, when we 19 do shift, it's because you're trying to do division, 20 multiplication by powers of two. But also 21 sometimes, you do shift because you are trying to do 22 bit extraction and so on. So it depends what 23 you're -- exactly what you're trying to do. 24 Q. And in the Jiao reference, you point to -- 25 well, we've got it right here. Let's take 11, 4 to	116 1 responding to functionality pertaining to 2 instruction decode. Of course, you have to decode 3 to know what it is. Operand fetch, if you know the 4 operands to be there, you go and fetch them from 5 cache or from memory. Branch computation, if it's 6 conditional in your program flow, you need to decide 7 if you want to go right or left on the branch. 8 Instruction execution, that's when you actually 9 execute the instruction in the ALU. Shift/logic, 10 because sometimes you want to shift, for reasons I 11 described. Load/store, finally, when you have to 12 load values, maybe constants, you have to store 13 results back to some memory. 14 "Among other processing functionality or 15 logic." So this description, for a person skilled 16 in the art, clearly unambiguously implies that EUDP 17 514 is a component capable of doing general-purpose 18 compute. 19 Q. Okay. And EUDP 514 is in -- looking at 20 Figure 1, that will be located in CPU 126; right? 21 A. Okay. 22 Q. Is that correct? 23 A. I don't remember off the top, but I assume 24 that's correct. Please go ahead. 25 I'm not sure, actually. It -- no, no,

Conducted on January 5, 2024

117 1 probably not. I mean -- no, no. I think -- now 2 that I look at this again, I think it's part of the 3 GPU 114. It's part of the graphics processing unit, 4 GPU 114. I take it back. It's part of the GPU 114. 5 Q. Okay. And where does Jiao say that? 6 MR. JOHNSON: Objection. Form. 7 A. So look at Figure 6A; right? Figure 6A 8 shows EUDP 514 as feeding into the constant buffer 9 controller 602 of the -- of Jiao system; right? And 10 there is -- I think there's no argument ever about 11 the CB controller being outside the GPU; right? And 12 this EUDP 514 is directly feeding into it -- also 13 directly receiving the constant information from the 14 constant buffer 510; right? And so, if -- if it was 15 in the CPU, as you will find it, of course, there 16 will be no such direct connections. You would have 17 to go through, I don't know, PCIE BIU 118, the 18 chipset 122, the bus that is there to the CPU 126, 19 and, God knows, some other components within that 20 CPU, to get there. So definitely, it is not in the 21 CPU 126, and 100 percent it's inside GPU 114. 22 Q. Okay. And you're -- I'm a little confused 23 on your rationale for that. It's that it describes 24 the constant buffer in Figure 5; is that your reason 25 why item -- I believe it was -- 512 is in the GPU?	119 1 that you are rely on for claim 19; right? 2 A. Yes. 3 Q. Okay. And so, it's not the shader 4 programs for these shaders that we talked about that 5 perform the functions in 11, lines 4 to 14; right? 6 A. No. But it -- so it -- the Jiao reference 7 does not describe the physical location of the EUDP 8 514, as I recall. But it is -- would have been 9 obvious to a person skilled in the art, that the 10 EUDP 514 because of its nature, which is a 11 computational resource, would be included in the 12 computational fabric underneath the Jiao system, 13 i.e., it would be part of the shader core doing this 14 whole thing, computational or processing work for 15 the graphics applications. 16 So it's a resource. Again, you have a 17 resource in the system that can do general-purpose 18 compute. And you can use that to do general-purpose 19 computation, which is a computational work; right? 20 And so, it is then: What is the language 21 of the claim you're talking about here? Let's look 22 at the exact language and see if there is any 23 additional requirement that EUDP has to satisfy. 24 And all it's saying is that the computing system of 25 claim 15 -- it doesn't say the shader core or -- it
118 1 A. Not only Figure 5. Because Figure -- in 2 the Figure 5, it shows EUDP is there; right? As 3 part of this constant buffer system. But if you 4 look at Figure 1, the constant -- or Figure 2; 5 right? It doesn't start with Figure 1. Figure 1, 6 clearly, the whole constant buffer system 200 is 7 part of GPU 114. In fact, that's the main 8 invention; right? That's the main thing about 9 Wilt's claims: This constant buffer system and how 10 they use it, and it is inside their GPU. 11 Q. Okay. How does the constant buffer system 12 relate to Figure 4? 13 MR. JOHNSON: Objection. 14 A. Good question, yes. The constant buffer 15 system is used to, sort of, provide the constants 16 that are needed, the rendering context information 17 that's needed for various graphics processing tasks 18 in a very efficient way so that you can maximize the 19 performance of the system of -- of Jiao. So it is 20 part of GPU and there -- it's a resource available 21 to the graphics processing tasks generally. It's a 22 resource that's heavily used, actually. 23 Q. And so the -- I think you already said 24 this, but let me confirm. The EUDP 514 is what 25 performs the operations in column 11, lines 4 to 13,	120 1 says -- just says the computing system of claim 15 2 where the first task comprises a graphics processing 3 task and the second task comprises a general-compute 4 task. These tasks come in. If it's general-compute 5 task, it will be dispatched to EUDP to perform it. 6 If it's graphics processing task, it just describes 7 to the rest of the shader core that's doing the 8 work. 9 Q. Does Jiao ever state that EUDP 514 can 10 comprise the computational elements 420? 11 A. No. 12 Q. Okay. You opined that the three shaders 13 that we've been discussing, the pixel shader, 14 geometry shader, and vertex shader load state data; 15 correct? 16 A. Yes. 17 Q. And then, you also have an additional 18 theory that the engine -- plurality of engines in 19 the claims comprise what Jiao calls a "thread slot." 20 Do you recall that? 21 I'm sorry. I'm not sure if we caught your 22 answer, Dr. Pedram. 23 A. Yeah, no, I haven't responded. I'm sorry. 24 I'm trying to read the transcript, but it doesn't 25 make sense, the sentence. So I think maybe they're

Conducted on January 5, 2024

121 1 going to fix it later on. But at this point when I 2 read the sentence, it doesn't make any sense to me. 3 So maybe you can repeat the question, 4 please. 5 Q. Sure. I'll just point you to your 6 declaration. So you have an alternative allegation 7 of what the claimed engines are here at 8 paragraph 98. And you say, "In the alternative, the 9 claimed engines are"-- the three shaders -- "plus, 10 'thread slots' within execution units 420." 11 Do you see that? 12 A. Yes, sir. 13 Q. What are these thread slots? 14 A. These are like reservation stations. 15 Locations in which a thread can be -- or a pointer 16 to the starting address of the thread is being 17 inserted and says, "Well, whenever you're ready to 18 run that thread, here is the starting address. Go 19 and run it." 20 So it's just a pointer. Some kind of 21 pointer to the thread to be executed on the core 22 within which that slot exists. And there are these 23 cores. They have their own thread slots. And if a 24 thread is injected into one of the slots of one of 25 the cores, it means you have to run it.	123 1 THE VIDEOGRAPHER: We are going off the 2 record. The time is 2:20 p.m. 3 (Recess.) 4 THE VIDEOGRAPHER: We are back on the 5 record. The time is 2:29 p.m. 6 MR. DOKHANCHY: All right. Could we 7 please introduce, as Exhibit 6, Document 9. 8 (Exhibit 6 marked.) 9 BY MR. DOKHANCHY: 10 Q. Dr. Pedram, I want to ask your opinions 11 about the Stuttard reference. Let me just -- let's 12 jump to the claims first. On the '381 patent, there 13 is a requirement that the shader core be configured 14 to execute a first task from the plurality of tasks 15 while executing a second task, quote, "from the 16 plurality of tasks"; correct? 17 A. Yes, sir. 18 Q. And Exhibit 6 that's up on the screen now, 19 that is the Stuttard reference; is that correct? 20 A. Yes, sir, it is. 21 Q. And one of the areas you point to in 22 Stuttard -- well, I guess before we get into that, 23 the -- in terms of what you point to as the shader 24 core, it's the set of processing blocks 106 in 25 Stuttard; is that right?
122 1 Q. The thread slots in Jiao do not load state 2 data; correct? 3 A. No, they don't. 4 Q. Going back to Jiao, we talked earlier in 5 the context of the '381 patent about engines and 6 whether they have distinct command buffers. And so 7 my question here is: The vertex shader, geometry 8 shader, and pixel shader of Jiao, are those 9 described as being connected to distinct command 10 buffers, one for each engine? 11 MR. JOHNSON: Objection. Form. 12 A. In Jiao's system, these shaders -- vertex 13 shader, geometry shader, pixel shader -- they are 14 not associated with distinct command buffers. Jiao 15 does not exclude that possibility. He doesn't say 16 that there are no such command buffers. So it is 17 possible, but I don't recall any disclosure of -- in 18 Jiao -- express disclosure that they exist. And -- 19 but Jiao is certainly -- and Jiao's teachings 20 certainly do not exclude the presence of such 21 buffers. 22 MR. DOKHANCHY: All right. Let's take a 23 ten-minute break, please. 24 THE WITNESS: All right. 25 MR. DOKHANCHY: Thank you.	124 1 A. Yes. It's the set of processing 2 blocks 106; correct. 3 Q. And we see in Figure 3 that set of 4 processing blocks is something distinct from thread 5 manager 102; correct? 6 A. Yes, sir, that is correct. 7 Q. And so, in Stuttard, one of the citations 8 you provide for this concept of executing two 9 threads in the shader core is on page 11, this 10 citation starting at line 16 that says, "At any -- 11 and starting on page -- sorry, line 19, "At any 12 particular moment in time, each thread may be 13 assigned to any one of a number of tasks." 14 Correct? 15 A. Yes. 16 Q. And this is in the section on -- if we go 17 up to the previous page, 10, on the thread manager; 18 is that right? 19 A. Yes. 20 Q. And so we'll flipping back and forth to 21 the figures a bit, but let's -- so let's jump down 22 to Figure 4. And you talk in your petition that -- 23 or excuse me, your declaration, about eight threads. 24 And some can be assigned to some tasks, and some can 25 be assigned to other tasks; is that right?

Conducted on January 5, 2024

125 1 A. Yes, sir. 2 Q. And in the citation on page 11 that you 3 point to that is describing in the thread manager, 4 the eight threads that are discussed are items B0 5 through B7 in the TM cache 1024; correct? 6 A. Yeah, I mean, that's where they get 7 stored. But, of course, they come from top. Yes. 8 (Reporter clarification.) 9 THE WITNESS: Yes. From top of the 10 figure, there is a description that threads from 11 EPU-controlled DMA, coming in on bus 1022, getting 12 stored into this cache unit 1024. 13 BY MR. DOKHANCHY: 14 Q. And so, it is the threads stored in B0 15 through B7 in the thread manager that can be 16 assigned to different tasks, as stated in page 11? 17 A. Yes, sir. That's correct. 18 Q. And the -- just taking a step back. We've 19 talked about the processing blocks. Those are the 20 components that you say execute tasks; is that 21 right? 22 A. Yes, sir. 23 Q. The thread processor 0 through 8, you do 24 not allege that those execute tasks; is that right? 25 MR. JOHNSON: Objection. Form.	127 1 There's -- I mean, as you know, also you 2 may recall there are other descriptions in Stuttard 3 that talk about multithreading and multitasking in 4 the system. 5 Q. I do want to get to those in a second. 6 But let's focus on this section. So as we just 7 talked about, this is a section describing the 8 thread manager 102. And this section that you 9 pointed me to in the middle of page 11 doesn't 10 explicitly mention the capabilities of the 11 processing blocks. 12 Would you agree with that statement? 13 A. I'm not sure if I understand the -- your 14 question. When you say "the capabilities of the 15 processing blocks," what do you mean by 16 "capabilities of the processing block," that 17 supposedly should have been described in this 18 paragraph and are not? 19 Q. Yeah. So this section is describing, as 20 the heading states, the thread manager. And it's 21 describing the threads and the assignment of threads 22 in the thread manager; right? 23 Is that correct? 24 A. Yes, that's correct. But these threads 25 feed into the processing blocks of Stuttard. So any
126 1 A. They do some processing in terms of the 2 flow control and so on, but they -- they do not 3 perform the graphics processing tasks or 4 general-purpose processing tasks that user provides 5 to them. 6 Q. And so, we go to column 11 -- sorry -- 7 page 11. This section on the thread manager does 8 not state the capability of processing blocks 106 to 9 simultaneously perform different tasks. 10 Do you agree? 11 A. No, I do not agree. 12 Q. Okay. Is there any place in this section, 13 including page 11, that describes what tasks the 14 processing blocks can perform, other than this last 15 paragraph that's up on the screen right now on 16 page 11? 17 A. Yes, there is. The paragraph above it, 18 that we can both see, describe the fact that you 19 could have multiple tasks active at the same time. 20 For example, the task would be assigned to thread 0, 21 which does the general system control. There's a 22 task assigned to thread 1, which does 2D activities. 23 And there are six remaining threads, 2 to 7, 24 assigned to 3D activities such as calculating 25 vertices, primitives, or rasterization.	128 1 description of the threads here and the ability of 2 the system to handle multiple of these threads at 3 the same time also applies to the processing blocks 4 that receive the threads. That's how I would read 5 it. That's how a person skilled in the art would 6 read it. 7 Q. And so, where does Stuttard describe the 8 relationship between threads in the thread manager 9 and threads in being actively run in the processing 10 blocks 106? 11 MR. JOHNSON: Objection. Form. 12 A. To me, this one paragraph, I guess it's 13 page 11, that you are in the middle paragraph that 14 describes general control versus 2D versus 3D 15 activities, is a clear description that multiple 16 tasks could be concurrently executing in the 17 Stuttard system. The system is configured to or 18 capable of performing multiple tasks at the same 19 time. Because you could have these threads all 20 coming in and being assigned to different processing 21 blocks of the Stuttard. 22 This is also in view of the multithreading 23 and multitasking statements of Stuttard and in view 24 of the architecture. The abstract does state, for 25 example, that the key invention or one of the

Conducted on January 5, 2024

129 1 inventions of Stuttard is that they organize the 2 processing elements into multiple processing blocks 3 so that these processing blocks could actually 4 support different tasks running. That would be a 5 person of skilled in the art's understanding of the 6 totality of the Stuttard reference. 7 Q. Well, let's jump to the abstract since you 8 mentioned it. And you said that it's stated that 9 the multiple processing blocks can process different 10 tasks. 11 So I've got the abstract up on the screen. 12 What it says is "The processing elements are 13 operable divided into a plurality of processing 14 blocks, the processing blocks being operable to 15 process respective groups of data items." 16 It doesn't say anything there about tasks 17 or instructions but, rather, data; wouldn't you 18 agree? 19 A. I would agree that what you read to me in 20 the abstract is accurate. What I said, then, if you 21 look at my response, is the person skilled in the 22 art looking at the totality of this Stuttard 23 reference in view of the abstract: The fact that 24 the processing elements were organized into 25 different processing blocks; the fact that there is	131 1 other than the multiple data of a SIMD? 2 A. The processing apparatus described in 3 Stuttard comprises of multiple processing blocks. 4 Each of these processing blocks is organized as its 5 own SIMD engine, independent SIMD computational 6 engine, which means each of the processing blocks 7 can run instructions of a single thread on 8 respective data items, but different processing 9 blocks could run different threads. 10 That is why Stuttard reference does the 11 partitioning. If, in fact, Stuttard system intended 12 to have one and only one big SIMD array of 13 processing elements, there would be no need for it 14 to organize it into different processing blocks with 15 eight different instruction -- sorry, they're called 16 "thread processors." Each of these thread 17 processors putting potentially different instruction 18 streams together for different processing blocks. 19 If, in fact, this was a single SIMD array, 20 as AMD seems to allege, a singular thread processor 21 in Figure 4, the thread manager would suffice. Why 22 should I have eight different thread processor, each 23 one of them responsible for one of the threads that 24 runs on one of the processing blocks, if all the 25 threads were identical? This is nonsensical. This
130 1 reference to multithreading, multitasking; the fact 2 that there's explicit disclosure that you could be 3 doing multiple different types of thread at the same 4 time by assigning them to different threads in the 5 execution pipeline; looking at that, would conclude 6 that Stuttard's reference is not only running one 7 thread on one task, but it is capable -- although, 8 we can do that also -- but it is capable of also 9 running multiple tasks and the corresponding threads 10 at the same time. 11 Q. I was just asking about the abstract and 12 you started talking about a lot of other things. So 13 let's stick to the particular question at hand. 14 My question is: The abstract says that 15 the data processing apparatus comprises a SIMD, 16 which means single instruction multiple data, array 17 10 of processing elements. Single instruction 18 multiple data. The processing elements are operable 19 divided into processing blocks. The processing 20 blocks being capable to process respective groups of 21 data items. 22 That data items reference is a reference 23 back to multiple data in the SIMD; right? 24 A. It can. 25 Q. Okay. What else would it be referring to	132 1 is redundant. This is wasteful. They would have 2 not do it. 3 The fact that there are eight independent 4 thread processors is a very strong indication that 5 these processing blocks are independent of each 6 other. Each one of them is SIMD; correct? Which 7 means each one of them will run multiple threads or 8 the same thread -- I'm sorry -- the same thread on 9 multiple data items. 10 Q. Well, let me ask you this. So a single 11 SIMD benefits from a design of single instruction 12 multiple data wherein the processing elements within 13 a SIMD, all operate on the same instruction but on 14 different data; correct? 15 A. Yes, sir. 16 Q. Why, then, is it not simply an extension 17 of that benefit it to have multiple SIMDs, wherein 18 each SIMD all run on the same instruction? 19 A. I already answered the question, so I'm 20 not -- I'll try again. Sorry to be repetitive in 21 some ways. 22 Q. I don't want you to be repetitive. Let me 23 re-ask the question a different way, then. 24 You have said that it would be redundant 25 and a waste of resources operating on the same

Conducted on January 5, 2024

133 1 instruction; right? 2 A. No, I didn't say that. What I said, sir, 3 was that if, in fact, AMD and its experts -- expert 4 were correct that Stuttard is only describing a 5 single SIMD array, they can perform only one thread 6 on different data items at the same time and no 7 more, then it would have been wasteful design. 8 Because Stuttard system has spent resources, 9 allocated computational resources to create eight 10 thread processors. 11 These eight thread processors -- each one 12 of them is in charge of one of the thread and one of 13 the processing blocks. And if the same stream is 14 running all the time on their own application 15 scenarios, there's no need to have eight thread 16 processors. Only one thread processor would do the 17 job. 18 Obviously, that's not what they did so 19 that's obviously what they did not intend to do. 20 They intended to do more than this. Although, it is 21 possible for the Stuttard system to run exactly and 22 only one thread on all of its processing blocks, it 23 is not the only thing it can do, as described, also 24 explicitly, on page 11, the paragraph that we talked 25 about, but also implied by someone looking at the	135 1 Option one, I organize this 128 processing 2 elements into one big SIMD array with one thread 3 processor. Or, I organize the 128 processing 4 elements into eight parallel blocks. Each one of 5 them having 16 processing elements and eight 6 different thread processors allowing eight different 7 instruction streams to run through these processing 8 blocks. 9 Do you see a benefit of one versus the 10 other system? The other resources is relatively in 11 terms of the PE count. The only difference is in 12 the organization from one array to eight subarrays, 13 you like to call it. And then, eight different 14 thread processor and then just one thread processor. 15 And the answer to that case is obviously, 16 there's an advantage to the second design of eight 17 subarrays. Each one of them with 16 processing 18 elements, because now that enables me to run eight 19 different threads at the same time. 20 In the first system, one big array, all 21 SIMD, only one thread could run. Maybe this thread 22 I'm running only has 12 items -- data items to 23 operate on. So then what? 24 Only one of these 12 of the processing 25 elements are used. The rest of them are sitting
134 1 Figure 4 and the design that they have come up with. 2 And if you look at what the thread 3 processor does, which actually does the control flow 4 of instructions and assembles together the 5 instructions that get executed in a processing 6 block, then they would come to the conclusion -- the 7 only logical construction that these thread 8 processors are there to orchestrate, potentially, 9 the execution of multiple tasks. Multiple threads 10 corresponding to multiple tasks. The configuration 11 is there. The capability is there. 12 Q. I think we're talking a little bit past 13 each other. I don't want to belabor the point, but 14 let me see if I can concisely rephrase my question. 15 Why wouldn't the same benefit that results 16 from a single SIMD having multiple processing 17 elements also extend and also be beneficial when you 18 have multiple SIMDs with even further processing 19 elements? 20 A. The question is not well posed. The 21 question could be: If I fix the total number of 22 processing elements that I have, say, into 128 23 total; right? I'm just coming up with a number. 24 128 total processing element. Now I have two 25 options.	136 1 idle. They do nothing. But they have to sit idle, 2 do nothing. That's not a good system. That's not 3 an efficient system. That's not what Stuttard came 4 up with. Am I clear in what I'm saying, sir? 5 Q. I think so. But I can envision some 6 benefits I think you're ignoring, and I'd like to 7 get your take on them. So the increased benefit of 8 dividing up the processing units into more than one 9 SIMD, I think, could follow from -- could mirror the 10 benefits you get from parallelism in other contexts. 11 For example, don't you think it's 12 beneficial if one of those SIMDs were to break or go 13 down, the other seven of them could keep operating? 14 Would that be a benefit? 15 A. You're talking about redundant design. 16 That's never been claimed or stated either in the 17 '381 design or in Stuttard. There's no such 18 reference. This is pure speculation that the 19 Stuttard came up with this design because it wanted 20 to make sure that if one of the processing blocks 21 breaks, all the other processing could -- still the 22 processing could continue. 23 But first of all, that's not the only 24 option. Even if I have a one big SIMD array, all my 25 128 processing elements are there. As long as I

Conducted on January 5, 2024

1 could keep track of the status of the processing 2 elements that failed on me, I could just bypass 3 them. This is how most people know how to do it. 4 In memory, that's how it's done. In processing, 5 that's how it's done. 6 I just know the status of the block -- the 7 processing element is broken or not. If it's 8 broken, I just bypass it. I set a flag. Suddenly, 9 my big processing array from 128 PEs drops to 116 10 because 12 of them broke. Fine. 11 I mean, I don't need to organize it into 12 eight parallel processing blocks, have eight 13 different thread processors with their own flow 14 control and complex designs to manage all of these 15 things. Because I can tell you none of the AMD 16 experts has ever come up with a response to what I'm 17 saying, because there's no response. 18 And what I'm saying is if, in fact, 19 Stuttard meant that although the system is 20 partitioned into eight blocks, all of these blocks 21 are only running one thread, there's no reason in 22 the world that they would have eight different 23 thread processors present. Even in the case of the 24 false (phonetic) that you described. Okay. This 25 processing block broke. Set it aside.	137 1 maintain processing of simultaneously active program 2 threads, so that each of the processing blocks 106 3 can be active for as much time as possible." 4 They are explicitly saying there could be 5 multiple simultaneously active program threads. 6 It's there. It's in writing. Black and white. 7 Q. Sir -- 8 A. Come back and says, "Well, there's an 9 example." I give you a critical example. In that 10 particular example, it happens to be the case that 11 the same instruction stream is applied to all of the 12 processing blocks. 13 Sure, this is an example. As there are 14 other examples, and other the examples would be just 15 a paragraph as well in which there are three 16 different threads, at least, running -- I'm sorry -- 17 eight threads. At least three different tasks 18 running at the same time. 19 Q. Sir, respectfully, you're 20 mischaracterizing the document. It does not say "In 21 some other new particular example." It says "In 22 this particular example." "This" means exactly what 23 you read. "The thread manager maintains processing 24 of simultaneously active program threads so that 25 each processing block can be active for as much time 139
138 1 Why should I have eight different thread 2 processors doing potentially different things but 3 still one thread processor would have been enough. 4 And they have had ample opportunity to respond. No 5 one ever responded. Because there's no response, as 6 far as I can see. 7 Q. I'd like to go back to page 11. And the 8 last paragraph here says, "In the example shown in 9 Figure 4, the thread manager includes one thread 10 processor for each thread." And then, at the end of 11 this paragraph, it states, "In this particular 12 example, the same instruction stream is applied 13 [sic] to all of the processing blocks in the 14 system." 15 My first question is: What is the 16 antecedent basis for the phrase "this particular 17 example"? 18 A. First of all, because we are reading in 19 the record, I point out that you -- you -- you 20 skipped a critical sentence here. I read it for the 21 record. 22 Q. Okay. 23 A. In that same paragraph, it says, "The 24 thread processors 1026 control the issuance of core 25 instructions from the thread manager so as to	140 1 as possible." 2 In this. Not some other example. In this 3 example, the same instructions stream is applied to 4 all threads; correct? 5 A. That's what I read. I mean, it says -- in 6 the same way that started that same paragraph, 7 what's -- by saying, "In the example shown in 8 Figure 4," rather than saying, "In an example shown 9 in Figure 4." What is "the example" mean here, as 10 if it's been discussed before? 11 So an example shown in Figure 4, it didn't 12 say that; right? It said, "In the example shown in 13 Figure 4." 14 Here, it's also saying, "In this 15 particular example," which may or may not be the 16 same as the example shown in Figure 4, "the same 17 instruction stream is applied." 18 I don't necessarily connect the two. I 19 understand your take on it, but I do not necessarily 20 connect the two. And regardless, to me, this is not 21 really an important point for distinction. The fact 22 remains that there's clear disclosure in the 23 architecture of the Stuttard system. 24 In the paragraph above what we're 25 discussing now, in the middle of the paragraph we're

Conducted on January 5, 2024

141 1 discussing about, that there could be simultaneously 2 active program threads. 3 Now in one example, this particular 4 example, there's just one active thread. Fine. I'm 5 not saying that's not possible. 6 Q. What is the antecedent? 7 MR. JOHNSON: Hold on. Dr. Pedram, can I 8 just ask you to slow down a little bit. Because 9 you're going so fast that I think the court reporter 10 is probably... 11 A. I'll try. Thank you. Thank you for 12 pointing this out. 13 Q. Dr. Pedram, what is the antecedent basis 14 for the phrase "this particular example" in line 33 15 to 35 of page 11 of Stuttard? 16 A. I think it's trying to say -- to me, 17 there's an example shown in Figure 4. Okay. Fine. 18 And it's not clear so let me put it another way. 19 Let's look at Figure 4 first then I'll make my point 20 again. 21 Q. I just want to -- 22 A. No, you asked the question. The question 23 is open. Let me finish. I'll be quick. 24 Figure 4 is not an example scenario of, 25 for example, a task coming in or multiple tasks	143 1 the BMW is -- the car is blue. Okay. Fine. 2 So one is making a statement that this is 3 a BMW. The other one is making a statement that it 4 is blue. Yes, they are two different -- yes, they 5 are related, because one continues from the other. 6 But they are not the same thing. They are 7 describing different things. One is the describing 8 a car. One is describing the color of the car. 9 That's like this. 10 Q. Okay. That was very confusing and I 11 really don't understand what you're talking about. 12 But if you just want to answer my question, it is: 13 What is the antecedent basis for the phrase "this 14 particular example"? 15 A. I'm sorry. Ask your question again. What 16 do you mean by "antecedent basis." I don't 17 understand the meaning of antecedent basis as you 18 apply. 19 Q. Do you know the concept of when you say 20 "this," it refers to something previously discussed? 21 A. Yes. 22 MR. JOHNSON: Objection. Form. 23 Q. Okay. So -- and that is the concept of 24 antecedent basis. Something that appears earlier in 25 the text about which the word "this" is referring
142 1 coming in and so on. It's just a block diagram; 2 right? So it's showing -- the example there that's 3 mentioned is an example describing this 4 architecture; right? The architecture of the 5 Stuttard system as it relates to thread manager. 6 And it shows there are up to eight threads existing 7 and so, on so forth; right? 8 And then the second time, it says, "In 9 this particular example." It changes the nature of 10 what's being discussed; right? 11 So the first one is architecture. The 12 second one is giving you, in this particular case, 13 now let's consider the fact that there's only one 14 active thread. It's referring to the same figure. 15 I'm not saying it's not referring to the same 16 figure, but it is not saying that, therefore, the 17 first example which related to the architecture -- 18 an architecture block diagram of the thread manager 19 is of the same nature -- apple and orange nature -- 20 as the second particular example that really talks 21 about one thread being active. 22 So you understand what I'm saying? So it 23 is like you're talking about -- I don't know a -- 24 a -- I don't know -- BMW being an example of a car, 25 and then saying in the -- in this particular case	144 1 to. 2 Does that make sense? 3 A. Okay. 4 Q. Okay. So with that understanding, what is 5 the antecedent basis example in that paragraph that 6 the phrase "this particular example" at the bottom 7 of page 11 of Stuttard is referring to? 8 MR. JOHNSON: Objection. Form. 9 A. As I said before, I read this paragraph to 10 describe -- to provide additional information by way 11 of an example of the architecture that's being 12 discussed in Figure 4. So that's what I read it. 13 When it says "in this example," Figure 4, it's 14 referring to the architecture of the thread manager 15 as shown. 16 When it talks about it, "in this 17 particular example," means that let me make a 18 further assumption, and that further assumption is 19 now only when a thread is active. It's not saying 20 the first case that was the architecture is limited 21 to the special case assumption I made later on. 22 That is what you are trying to imply by 23 the question about antecedent basis; right? That 24 because I started something general as an example 25 and later on, I narrowed down that example by adding

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 37 of 182

Conducted on January 5, 2024

1 further limitations to it, that limitation existed 2 from the very beginning in the original system. 3 That, I disagree. I understand that's your read. I 4 understand why you're also saying it, but I disagree 5 with that read. 6 Q. Do you see anything else described in this 7 paragraph as an example, other than the example 8 shown in Figure 4? 9 MR. JOHNSON: Objection. Form. 10 A. Again, the question has been asked and 11 answered. I think it's somewhat going to be 12 repetitive. I can go back and repeat everything 13 I've said, but really what I've said is what I mean. 14 And there's nothing that you could ask in any other 15 form that would make me change my statement. 16 So if you want me go through the exercise 17 and -- I'll do it again. But I thought I was very 18 clear what I think this sentence meant -- or this 19 paragraph meant, even in view of your antecedent 20 basis so on and so forth. But if you want me to 21 continue, sir, I'll continue. 22 Q. No, that's fine. If you have nothing to 23 add to your previous answer, we can move on. 24 A. Thank you very much. I have nothing 25 further to add.	145 1 system control application. 2 Q. Can you give me any examples of general 3 system control tasks that are not computational 4 tasks? 5 A. Yes, of course. A general system control 6 task would be a state machine. A state machine 7 says, "Under this input response, provide that 8 output response." And that output response may be a 9 computational response but could also be just a 10 physical response or an actuation response. 11 So systems that have sensors and 12 actuators, these are not computational systems; 13 right? You have a physical sensor, you have a 14 physical actuation thing, and you have a general 15 control algorithm that tells you, responds to what 16 sensor input, what actuation should be applied; 17 right? So you're just sending a control signal: 18 "Do this, do that." A state machine does it, for 19 example. Or a bunch of interacting state machines 20 would do it. Concurrent state machines will do it. 21 So these are well-known examples to the 22 general computing community that general -- that 23 softwares and hardware systems are either data 24 processing systems, there are general control or 25 maybe custom control systems, and -- or a
146 1 Q. Okay. We'll go back to your declaration. 2 Here in paragraph 135 of your declaration, you opine 3 on why Stuttard discloses the limitations of claim 4 19 of the '381 patent; is that right? 5 A. Yes, sir. 6 Q. And you point to the concept of general 7 system control. 8 Do you recall that? 9 A. Yes. 10 Q. So we'll come back to the word "general," 11 but what sort of activity is "system control"? 12 A. Process flow control is an example of a 13 system control. You have some kind of system that 14 can perform any number of different functions in 15 response to any number of different situations. So 16 reactive system, for example. And then, there's a 17 controller or the control algorithm that gets 18 executed on some embedded processor that tells such 19 a reactive system how to react to environmental 20 conditions or to new events and so on. 21 And so, typical example would be you have 22 an autonomous self-driving car and something's 23 happened, and you have to decide if I want to brake 24 or maneuver this, that, to the left or the right, or 25 what have you. And so, that's an example of a	147 1 combination of the two. It's not necessarily only 2 one or the other. 3 And I think Stuttard system was designed 4 so that it can provide both types of functionality. 5 It can run state machines to do general system 6 control without any execution of component to it or 7 with any minor little execution processing. Or it 8 could be data-dominated, such as -- data-intensive, 9 such as signal processing or -- or graphics 10 processing. 11 Q. And what about IO operations, memory 12 accesses, and things like that? Are those within 13 the bucket of computational tasks? Or are those 14 control tasks that are not computational? 15 A. No, I mean, the other -- neither. I mean, 16 when you say "IO," usually to even do data-intensive 17 computations, graphics processing tasks, you have to 18 do IO access. Typically going to DRAM, dynamic RAM, 19 external memory, bringing data in, is considered an 20 IO operation. You go and bring, through the IO 21 channels that you have, maybe there is a DDR 22 interface that fits the DDR4, what have you. You go 23 through that interface and bring data in so that you 24 can act on it. 25 So for example, the primitives you talked 148

Conducted on January 5, 2024

149 1 about, or the constants, or what have you may be 2 stored in some external memory. But when the 3 application is running, you bring them on in -- 4 on-chip in the system so that you can operate much 5 faster with much higher efficiency. So you use IO 6 mechanism, IO access to bring it. 7 So there are key tasks that need to be 8 performed. But I wouldn't put them either -- under 9 either data processing or reactive control system 10 classification, because they are -- they are all 11 tasks that potentially support both other types of 12 processes. 13 Q. Okay. So similar to the state data 14 concept we discussed earlier, you wouldn't consider 15 IO and memory activities themselves as computational 16 tasks; is that right? 17 A. I mean, this is a very open-ended 18 question. I can tell you I -- I do a lot of 19 research. We have published hundreds of papers on 20 many different things, and there's a very active 21 area of research on in-memory compute, or 22 near-memory compute, or doing some operations also 23 during IO. So it is possible to combine and mix 24 these things in very fancy, exotic, interesting 25 ways.	151 1 IO actions such as "memory retrieve," 2 "memory transfer," "data transfer," those in and of 3 themselves, you wouldn't consider computational 4 tasks; is that correct? 5 A. They could happen during -- as part of a 6 computational task. But by themselves, if you just 7 say, "I went to memory -- external memory through 8 the DDR interface to bring in some data item and 9 write it in the shared memory or in the local memory 10 or some processing unit in my system," the fact that 11 you went there and you brought it in, that's not a 12 computational task, no. It's not. 13 But if you are doing a computation, as 14 part of that computation that's doing matrix 15 multiplication, or doing invention of some matrix, 16 or what have you, as part of that, I will and most 17 likely and will most of the time go to memory to 18 bring whatever I need to operate on. 19 Q. Understood. That clarifies what I was -- 20 A. Thank you. 21 Q. -- asking. And as to claim 20 for 22 Stuttard -- we're at paragraph 136 of your 23 declaration -- you referred to threads with 24 different priority levels. And you say, "A task 25 would comprise one or more threads, and a
150 1 But in the context of what we are talking 2 about, the '381, the alleged prior art, I -- I would 3 agree. And I would state that, when we talk about 4 IO operations, it's not meant to include data 5 processing in it or as part of it. 6 Q. Nor computational tasks; is that fair? 7 A. Again, sir, really ill formed and 8 therefore hard to respond. I would -- no affront 9 intended, I'm really -- with all due respect, sir. 10 I don't mean anything negative. I'm just saying the 11 reason I hesitate to respond is because when you say 12 "computational task," a computational task does 13 involve IO operations. 14 So I cannot say if it's computational 15 task, it's not IO. This distinction is artificial. 16 Not there. Nonsensical in many ways. 17 So when you ask the question for me, "Does 18 IO operation include a computational task?" It's 19 not well formed. 20 Q. Okay. I'm happy to rephrase it. I -- 21 that's why I tried to preface it the way I did 22 before. But let me -- let me just ask the question 23 differently. 24 A. Please do. Thank you. 25 Q. The actions -- withdrawn.	152 1 high-priority task would give rise to high-priority 2 threads." 3 Does Stuttard disclose that the threads 4 within a given task all have to have the same 5 priority level? 6 A. I don't recall anything like that. And it 7 might have. I don't recall it. But the fact 8 remains that if a task is high priority, a 9 high-priority task, the threads that it spawns to be 10 executed, those threads also would be high-priority 11 tasks. Whether they are all exactly the same 12 priority or different priorities, I don't think 13 Stuttard has discussed that. It would be more of an 14 implementation detail. 15 And I can give you examples in which all 16 threads must have the same priority and maybe cases 17 in which they don't. Who knows. 18 Q. Well, let's -- sure. That would be 19 helpful. Let's talk about -- if you can give me an 20 example task, and then two subtasks that have 21 different priorities. 22 A. No, I can't do it off the top of my head. 23 I -- what I meant is that I have to think about it. 24 Think about it hard to see under what conditions it 25 makes sense and what application is and so on.

Conducted on January 5, 2024

153 1 The normal default assumption and the 2 straightforward implementation would be if a parent 3 task generates two child threads, their priorities 4 would be the same as the parent. That's what I 5 normally would assume is the case. 6 But I'm saying I can image that -- this 7 not being the case. But exactly what is the 8 application and what it means, I don't know at this 9 point, and I'm not ready to opine on it. 10 Q. Okay. If the -- if a task -- if two tasks 11 had constituent threads of a variety of priorities, 12 does the Stuttard reference allow the system to 13 change from a task -- excuse me, a thread in one 14 task to a thread in another task? 15 MR. JOHNSON: Objection. Form. 16 A. Well, I mean, I really have to sit down 17 and think. And these are not easy questions that 18 you're asking, and asking me on the fly when there's 19 no prior discussion from any of your experts 20 regarding any such issues. Maybe it's because of 21 some plan you have, making changes to some of the 22 claims or what have you, but I just don't understand 23 exactly -- 24 Q. Sure. I can -- 25 A. -- what you're asking me. And I --	155 1 possibilities as is -- okay. And so, I will take it 2 as is stated, which is it could switch from one 3 thread involving running to another thread if you do 4 all the right things in terms of state saving and 5 storage of the state when you start again, so, yeah. 6 Q. All right. And in paragraph 137, the last 7 sentence -- well, let me read the whole paragraph. 8 So 137 says, "Based on the disclosure of Stuttard, 9 it is my opinion that a POSITA would understand that 10 a task would comprise one or more threads and that a 11 high-priority task would give rise to high-priority 12 threads. Conversely, once all high-priority threads 13 associated with a task are completed based on their 14 assigned priority levels, the task has received a 15 high priority processing by the computing system." 16 And it's just the phrasing that I'm a 17 little confused by. What do you mean by "received a 18 high priority processing"? 19 A. What I mean here is that the task had -- 20 was a high-priority task. When a high-priority task 21 comes in, it doesn't mean that if a system is not 22 well designed, it will receive high-priority 23 threads. Imagine a dumb -- a dumb system that's -- 24 that's ignorant to any priority levels assigned to 25 any task.
154 1 Q. -- simplify. 2 A. -- cannot answer on the fly like this, 3 anyway. 4 Q. Okay. That's fine. I can simplify it. 5 And let's just focus in on your declaration. So you 6 talked about, in paragraph 136, the concept of 7 switching threads. For example, switching from one 8 thread to another, you mentioned in the middle of 9 the paragraph 136; right? 10 A. Yes. 11 Q. Okay. Does Stuttard limit the thread 12 switching to only be within one task? Or can 13 Stuttard switch from threads in the one task to 14 threads in another task? 15 A. Again, I really can't answer the -- off 16 the top of my head, this question. Because all I 17 have quoted here is the fact that you could switch 18 from one thread to another thread. And these 19 threads could correspond. So as it is, there's no 20 statement that says, "These threads should 21 correspond to one task. Only then you can switch," 22 or that, "These threads should necessarily 23 correspond to two different tasks. Otherwise, we 24 cannot switch." 25 Such a statement that allows both	156 1 So the task comes in. It keeps shouting, 2 "I'm high priority. I'm high priority." It's just 3 deaf. You don't hear what it's saying, or you don't 4 care; right? And so, you don't provide 5 high-priority service. It goes out the door like 6 any other low-priority task. 7 What I'm saying here is that Stuttard's 8 system is actually different. Stuttard's system is 9 a smart system that can hear a high-priority task 10 coming in saying, "I'm high priority. Give me high 11 priority," and says, "Fine. I'll give you high 12 priority. I'll give you the resources you need, so 13 that you execute faster and you exit the end -- the 14 back end of the system faster." 15 And when it does so, it has provided 16 high-priority processing to the incoming 17 high-priority task. In other words, the 18 high-priority task has now received high-priority 19 processing. 20 Is this clear, sir? 21 Q. I have to be honest, not really. But I'm 22 going to try to think about it a little bit before I 23 ask you to clarify. I think I understand what 24 you're saying. But are you saying that by virtue of 25 the constituent threads being granted a high

Conducted on January 5, 2024

157 1 priority and executed pursuant to that high 2 priority, the parent task has therefore been 3 executed according to a high priority? 4 A. Almost. Almost. The only addition to 5 what you said is that when a constituent thread has 6 been granted a high priority, in Stuttard's system, 7 it means it's going to run earlier than the other 8 task, the other threads, or may preempt some of the 9 running threads so that it can finish its execution 10 faster. It's not just granting of the high-priority 11 tag to a thread, but the fact that the system is 12 configured to respond to such assignments to 13 threads, honor it, and do its best to complete those 14 tasks at the expense of other running tasks. 15 Potentially, lower-priority tasks. I -- 16 Q. I think that kind of gets at the question 17 before, which is that: Can the system in Stuttard 18 switch between threads in one task to threads in 19 another task on the basis of the priority of the 20 threads themselves? 21 And I think what you just said is yes. 22 A. No, what I said -- I don't know. I have 23 to think about this more carefully. I have to go 24 back and look at Stuttard's reference and see 25 exactly how I would read behind the lines, in the	159 1 microcode instructions to the PEs in the processing 2 array." 3 Then you further say, "The PE instruction 4 sequencer 1044 and/or the load/store controller 1045 5 within the array controller 104, load the state data 6 associated with each task, e.g., data associated 7 with each task that is present in the PE register 8 file 1061b and/or the PE memory unit 1061." 9 Did I read that; right? 10 A. Yes. 11 Q. So your opinion here is that the 12 components of the thread manager that we discussed 13 earlier were -- sorry, withdrawn. 14 In that paragraph, 126, you're discussing 15 components within the array controller 104 that 16 loads state data; correct? 17 A. Yes. 18 Q. And then, the next paragraph, 127, you 19 opine that, "The channel controller 108 loads state 20 data"; correct? 21 A. Yes, that's correct. Yes. 22 Q. Okay. You have not opined that the 23 binning unit 1069 loads state data; correct? 24 A. Well, I mean, if you read the description 25 of the channel controller paragraph, 108, how it
158 1 white space between the lines to see under such a 2 scenario, how Stuttard's system would respond. 3 Maybe there is a -- there's an answer there, but 4 maybe it isn't explicitly described. And so... 5 Q. Okay. All right. Well, let's move on. 6 Are you okay to keep going, or do you want to take a 7 break? 8 A. No, I -- I'd rather keep going while I can 9 so that we finish maybe earlier hopefully. 10 Q. Yeah. All right. Let's jump back to 11 your -- well, it's part of your declaration. 12 Paragraph 125, you opine on the engines in Stuttard 13 and you say, "The binning unit 1069, the channel 14 controller 108, the array controller 104, and 15 components of thread manager 102 (including cache 16 memory unit 1024, thread processors 1026, status 17 block 1030) constitute the claim's plurality of 18 engines"; right? 19 A. Yes. 20 Q. Okay. And with respect to the loading of 21 state data -- 22 A. Yes. 23 Q. -- you state, "The PE instruction 24 sequencer 1044 uses instruction information to index 25 a PE microcode store 105 for transferring PE	160 1 loads state registers 1103 associated with each 2 task, you can see it -- there are references to 3 binning unit and the binnitization. 4 So binning unit is involved in this 5 process also; right? So this -- when you -- I can 6 read into the record if you want. But it -- but 7 it's there; right? In the -- in the codes on 8 page 43, line 22 to 44, page 9 [sic]. And in 9 Figure 7 that I have on page 67, paragraph 127 of my 10 declaration, there is a discussion of how the array 11 controller loads -- loads state registers and 12 binning units there; right? 13 Q. So my question is just simply: You have 14 not opined that the binning unit loads state data? 15 That's not an opinion in your report; correct? 16 A. Not that I recall. Though I've said the 17 channel controller loads state registers associated 18 with each task. 19 Q. Okay. And similarly, the thread 20 manager 102 does not load state data; correct? 21 A. I wouldn't say that. Because the status 22 block 103 within this thread manager contains this 23 status information for the threads. So definitely 24 thread manager is involved in this loading of the 25 state data -- or data.

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 41 of 182

Conducted on January 5, 2024

1 Q. We'll go to Stuttard in a second. But 2 let's first establish regarding your report. So 3 your report does not contain any opinion that any 4 component of the thread manager loads state data; 5 correct? 6 A. I disagree. I mean, maybe I have a 7 particular meaning about what "loading" means, and 8 that's why your question is being asked. So maybe 9 you have your own construction of the loading that 10 there's no agreement on what it means; right? 11 But -- 12 Q. No -- 13 A. -- if not, then clearly even first line of 14 paragraph 126 that I have in my report, "Stuttard 15 teaches that the status block 1030 within the thread 16 manager 102 receives status information" -- state 17 data -- "1036 from each of the threads." 18 So one of the key components of the thread 19 manager receives this information; right? 20 Q. Yeah, so I -- I'm not trying to get into 21 the actual -- well, we can in a second -- about the 22 substance. 23 What I'm saying is there's a claim 24 requirement for the engine or the plurality of 25 engines to load state data. So I'm trying to	161 1 loading the state data in that -- 2 Q. So I understand that's your opinion now. 3 I'm talking about what you wrote. You wrote that 4 the array controller loads state data, and you wrote 5 that the channel controller loads state data. You 6 never wrote the thread manager loads state data. 7 And if you think you did, please show me 8 where you said the thread manager loads state data. 9 A. I have to read through this. Maybe this 10 is a good time for a break when I can read through 11 the excerpts there and get back to you. 12 MR. DOKHANCHY: Okay. Let's take a 13 ten-minute break. 14 THE WITNESS: Thank you. 15 THE VIDEOGRAPHER: We're going off the 16 record. The time is 3:43 p.m. 17 (Recess.) 18 THE VIDEOGRAPHER: We're back on the 19 record. The time is 3:55 p.m. 20 BY MR. DOKHANCHY: 21 Q. Okay. Dr. Pedram, before we went off the 22 record, the question that you were looking at was: 23 In your declaration for this IPR proceeding, did you 24 ever opine that the thread manager or components of 25 thread manager loaded state data?
162 1 identify your allegations in your report that say, 2 "Here is my proof that this item loads state data." 3 And I see two items identified: One is -- 4 we went over -- you say, "The items within array 5 controller 104 load the state data corresponding to 6 the claim language." And then, another place in 7 127, you say, "The channel controller loads state 8 registers." 9 There's no allegation in your report that 10 anything within the thread manager 102 loads state 11 data; is that right? 12 A. No, I disagree with you. So we have a 13 discussion of the term "load" on page 21 under my 14 claim construction section. Paragraph 44, it says, 15 "The ITC has construed this term" -- i.e., the 16 load -- "to not include extracting and storing, but 17 otherwise did not construe this term." 18 It's presumed to have its plain and 19 ordinary meaning. I will adopt this limiting -- 20 more limiting construction for the purposes of this 21 declaration. So as long -- so that's what -- the 22 meaning I've assumed. And under this limiting -- 23 more limiting construction, that it's not to include 24 extracting and storing, to me, the thread manager 25 and the component within the thread manager is	163 1 A. No, I did not mean to say that. Also 2 previously having reviewed the record and my 3 declaration more carefully, the elements or 4 components of the thread manager are involved in 5 that process, but the actual loading of the state 6 data is done by the array controller or the channel 7 controller. 8 Q. Okay. And similarly, the binning 9 unit 1069 does not load state data; correct? 10 A. In the same spirit, yes. That is correct. 11 Q. Okay. Go to Figure -- Stuttard -- let's 12 start with Figure 4. The components -- withdrawn. 13 The engines that you've identified are 14 cache memory unit 1024. It's in the thread manager, 15 which feeds into status block 1030 -- or withdrawn. 16 Let me -- let me say that a little more simply. 17 So your opinion regarding the engines is 18 that the engines in Stuttard are cache memory 19 unit 1024, status block 1030, the thread 20 processors 1026. Then continuing into Figure -- or 21 back into Figure 3, the engines further include 22 array controller 104, channel controller 108, and 23 binning unit 1069; is that correct? 24 A. Yes, sir. 25 Yes, correct.

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 42 of 182

Conducted on January 5, 2024

1 Q. Okay. And are all of those engines in 2 parallel with one another? 3 A. No, they are not. 4 Q. And does Stuttard describe each of those 5 engines as being connected to a distinct command 6 buffer? 7 A. No, I don't believe it's -- it has done 8 so. 9 Q. And in terms of the flow of tasks -- go to 10 your declaration. So in Exhibit 1, your 11 declaration, at paragraph 129, you say, "The 12 processing blocks" -- which are the alleged shader 13 core -- "receive the plurality of tasks from at 14 least the thread manager 102 via the array 15 controller 104"; correct? 16 A. Yes, sir. 17 Q. You don't allege that the processing 18 blocks receive any of the plurality of tasks from 19 the channel controller 108; correct? 20 A. Correct, yes. 21 Q. And you still agree with that; that the 22 processing blocks 106 in Stuttard do not receive 23 tasks from the channel controller 108; correct? 24 A. Yes, that's my understanding. That's what 25 I recall.	165 1 for each of the pluralities of tasks that are 2 received. 3 Do you see that? 4 A. Yes, sir. I do. 5 Q. Do you agree with Realtek's position on 6 that issue? 7 A. That's not my opinion. That was not in my 8 invalidity report submitted in response to the same 9 proceeding. I understand this is part of the 10 strategy that respondents adopted with respect to 11 their noninfringement position. But what I have 12 opined, and what I'm able to opine today, and what 13 is reflected in any of my reports that was ever 14 submitted in relation to ITC for this particular 15 proceedings, I've always stated that it's not 16 required that the individual engines must be 17 configured of -- I'm sorry. You're changing the 18 document on me -- but that they must be configured 19 in parallel or anything like that. 20 Q. Okay. I don't think I moved the document 21 while you were speaking. But let me clarify, 22 because it wasn't about any sort of parallel and the 23 way you stated it. But my question is just simply: 24 Do you agree with the respondents' position in the 25 ITC that each individual engine might be configured
166 1 Q. And similarly, the processing blocks 106 2 do not receive tasks from binning unit 1069; 3 correct? 4 A. That is correct. That's also what I 5 recall, yes. 6 Q. Can we introduce, as Exhibit 7, 7 Document 14. 8 (Exhibit 7 marked.) 9 BY MR. DOKHANCHY: 10 Q. Okay. This is the public version of the 11 initial determination in the ITC matter at 12 Exhibit 7? 13 Do you see that, Dr. Pedram? 14 A. Yes, I do. 15 Q. And I'm going to scroll down to 104. And 16 this is Exhibit 2003 in this IPR proceeding, also 17 for the record. 18 And you understand that, in this 19 proceeding, AMD was the complainant and Realtek was 20 one of the respondents? 21 A. Yes, sir. 22 Q. And you see at the bottom of page 104, the 23 judge is saying that the respondents, which includes 24 Realtek, assert that each individual engine in the 25 '381 patent must be configured to load state data	167 1 to load state data? 2 A. No. Again, for the same reason. To me, 3 there are two requirements on the plurality of 4 engines. It is, one, to receive task from the 5 scheduling module. And the other one is that they 6 should load state data. And it's always been my 7 position that any one of those engines need to 8 perform or at least facilitate the performance of 9 one or the other of these two required elements but 10 not both at same time. So I do not agree with this 11 statement, no, sir. 12 MR. DOKHANCHY: Okay. All right. Pending 13 any redirect and follow up on that, that's it for 14 me. 15 A. Thank you. 16 MR. JOHNSON: I didn't know you were going 17 to be done so quick. Can I have a few minutes just 18 to look through the transcript to see if there's 19 anything I need to do? I don't think there is, but 20 just give me five minutes. 21 MR. DOKHANCHY: Sure. Five minutes. All 22 right. We'll reconvene in five. 23 THE VIDEOGRAPHER: We're going off the 24 record. The time is 4:06 p.m. 25 (Recess.)

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 43 of 182

Conducted on January 5, 2024

169 1 THE VIDEOGRAPHER: We're going back on the 2 record. The time is 4:18 p.m. 3 MR. JOHNSON: I have no further questions 4 at this time. 5 MR. DOKHANCHY: Okay. I think we can wrap 6 it up. Thank you, everybody. 7 THE VIDEOGRAPHER: Okay. Counsel, please 8 stand by. This concludes the deposition of 9 Dr. Massoud Pedram. We are going off the record. 10 The time is 4:18 p.m. 11 (Deposition concluded at 4:18 P.M.) 12 -oOo- 13 14 15 16 17 18 19 20 21 22 23 24 25	171 1 STATE OF CALIFORNIA)) SS. 2 COUNTY OF LOS ANGELES) 3 4 5 I, MASSOUD PEDRAM, PH.D, HEREBY CERTIFY 6 UNDER PENALTY OF PERJURY UNDER THE LAWS OF THE 7 STATE OF CALIFORNIA THAT THE FOREGOING IS TRUE AND 8 CORRECT. 9 EXECUTED THIS ____ DAY OF _____, 10 2024, AT _____, CALIFORNIA. 11 12 13 14 15 _____ 16 MASSOUD PEDRAM, PH.D 17 18 19 20 21 22 23 24 25
170 1 ERRATA SHEET FOR THE DEPOSITION OF: 2 3 CASE NAME: REALTEK SEMICONDUCTOR CORP. V ATI TECHNOLOGIES ULC 4 5 CASE NUMBER: IPR2023-00687 6 U.S. PATENT NO. 8,854,381 7 8 DEPO DATE: FRIDAY, JANUARY 5, 2024 9 10 DEPONENT: MASSOUD PEDRAM, PH.D 11 CORRECTIONS 12 PG. LN. NOW READS SHOULD READ REASONS 13 _____ 14 _____ 15 _____ 16 _____ 17 _____ 18 _____ 19 _____ 20 _____ 21 _____ 22 _____ 23 _____ 24 25 SIGNATURE _____ DATE _____	172 1 CERTIFICATE 2 I, Natalie Parvizi, Certified Reporter and 3 Notary Public within and for the State of California 4 do hereby certify: 5 6 That Massoud Pedram, Ph.D., the 7 witness whose deposition is hereinbefore set forth, 8 was duly sworn by me before the commencement of such 9 deposition and that such deposition was taken before 10 me and is a true record of the testimony given by such 11 witness. 12 I further certify that the adverse party, 13 was represented by counsel at the deposition. 14 I further certify that the deposition of 15 Massoud Pedram, Ph.D., occurred virtually, on 16 Friday, January 5, 2024, commencing at 9:12 a.m. PST 17 to 4:18 p.m. PST. 18 I further certify the inspection, reading 19 and signing of said deposition were not waived by 20 counsel for the respective parties and by the witness. 21 22 I further certify that I am not related to 23 any of the parties to this action by blood or 24 marriage, I am not employed by or an attorney to any 25 of the parties to this action, and that I am in no way

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 44 of 182

Conducted on January 5, 2024

173

1 interested, financially or otherwise, in the outcome
2 of this matter.

3 IN WITNESS WHEREOF, I have hereunto set my
4 hand this 10th day of January 2024.

5
6 
7

8
9 _____
10 NOTARY PUBLIC IN AND FOR THE
11 STATE OF CALIFORNIA
12
13
14
15
16
17
18
19
20
21
22
23
24
25

A			
ability	above	across	additional
128:1	47:10, 54:24,	35:10, 70:25,	24:25, 25:5,
able	66:3, 105:20,	77:24	37:4, 39:16,
14:17, 24:2,	126:17, 140:24	act	39:19, 48:9,
24:5, 25:6,	absent	148:24	119:23, 120:17,
26:22, 26:25,	42:17	action	144:10
57:25, 77:4,	abstract	172:23, 172:25	address
101:9, 167:12	128:24, 129:7,	actions	15:22, 28:6,
about	129:11, 129:20,	150:25, 151:1	58:24, 98:5,
16:19, 19:19,	129:23, 130:11,	active	103:13, 103:17,
20:19, 23:1,	130:14	126:19, 139:1,	104:11, 105:1,
24:11, 26:10,	access	139:3, 139:5,	106:11, 106:12,
26:24, 39:6,	44:9, 44:19,	139:24, 139:25,	121:16, 121:18
40:14, 44:18,	44:21, 44:25,	141:2, 141:4,	adequate
49:5, 50:2,	45:3, 45:9,	142:14, 142:21,	37:1
50:6, 57:8,	45:10, 45:16,	144:19, 149:20	admonition
59:3, 59:7,	46:6, 46:11,	actively	58:7
61:22, 63:11,	46:16, 46:22,	128:9	adopt
63:25, 64:18,	47:5, 49:8,	activities	162:19
71:6, 72:2,	50:20, 50:23,	126:22, 126:24,	adopted
72:3, 73:12,	51:15, 51:17,	128:15, 149:15	167:10
74:18, 74:24,	52:14, 54:4,	activity	adopting
75:18, 77:12,	55:21, 55:22,	146:11	34:5
77:18, 78:7,	56:2, 56:6,	actual	advantage
78:20, 78:21,	56:8, 57:3,	76:24, 77:4,	135:16
81:12, 82:19,	58:13, 58:17,	161:21, 164:5	advantageously
98:5, 105:21,	96:17, 97:9,	actually	65:18
106:15, 117:10,	148:18, 149:6	46:22, 56:18,	adverse
118:8, 119:4,	accesses	58:5, 58:12,	172:12
119:21, 122:5,	52:10, 57:25,	69:21, 107:13,	affecting
123:11, 124:23,	58:4, 58:12,	112:15, 116:8,	75:14
125:19, 127:3,	148:12	116:25, 118:22,	affirmatively
127:7, 129:16,	accessible	129:3, 134:3,	50:22
130:11, 130:12,	47:8, 48:17,	156:8	affront
133:25, 136:15,	49:12, 49:13,	actuation	150:8
141:1, 142:21,	49:23, 53:21,	147:10, 147:14,	after
142:23, 143:11,	54:6, 57:6	147:16	32:24, 48:9,
143:25, 144:16,	accessing	actuators	74:9, 106:25
144:23, 148:11,	53:12, 57:11	147:12	again
149:1, 150:2,	accommodating	add	10:21, 14:4,
150:3, 152:19,	60:12	14:22, 109:25,	19:13, 20:22,
152:23, 152:24,	according	145:23, 145:25	25:25, 26:19,
154:6, 156:22,	85:11, 87:4,	adding	27:21, 28:20,
157:23, 161:7,	104:17, 157:3	144:25	33:16, 36:13,
161:21, 163:3,	accurate	addition	37:23, 40:13,
167:22	6:24, 129:20	45:16, 47:23,	42:24, 45:13,
	acknowledge	107:12, 113:24,	49:3, 50:1,
	27:10	114:9, 157:4	50:8, 52:11,

55:9, 57:21, 61:22, 62:21, 69:9, 69:14, 69:15, 72:2, 76:1, 82:3, 82:18, 83:17, 86:5, 88:17, 103:15, 107:19, 110:20, 117:2, 119:16, 132:20, 141:20, 143:15, 145:10, 145:17, 150:7, 154:15, 155:5, 168:2 ago 16:25, 29:1, 29:12 agree 13:1, 98:8, 110:11, 126:10, 126:11, 127:12, 129:18, 129:19, 150:3, 165:21, 167:5, 167:24, 168:10 agreed 113:2 agreement 161:10 ahead 7:8, 8:2, 60:4, 65:5, 116:24 aid 94:18 algorithm 96:22, 98:21, 146:17, 147:15 algorithms 96:8, 109:20 alj 12:23, 15:23, 113:2 alj's 16:8, 95:4 all 5:13, 10:22, 12:19, 12:21, 13:21, 21:6,	26:9, 28:8, 29:24, 30:25, 31:15, 32:2, 35:22, 37:16, 38:23, 41:6, 41:17, 49:19, 49:22, 49:23, 50:2, 54:10, 57:22, 57:23, 58:21, 61:8, 62:22, 63:9, 66:24, 68:21, 69:24, 73:2, 74:16, 74:17, 82:10, 83:22, 84:15, 84:23, 85:21, 88:23, 90:4, 102:7, 102:25, 104:25, 108:19, 112:18, 113:25, 119:24, 122:22, 122:24, 123:6, 128:19, 131:24, 132:13, 132:18, 133:14, 133:22, 135:20, 136:21, 136:23, 136:24, 137:14, 137:20, 138:13, 138:18, 139:11, 140:4, 149:10, 150:9, 152:4, 152:11, 152:15, 154:16, 155:4, 155:6, 155:12, 158:5, 158:10, 165:1, 168:12, 168:21 allegation 121:6, 162:9 allegations 162:1 allege 64:10, 89:19, 125:24, 131:20, 165:17 alleged 63:25, 83:13,	150:2, 165:12 allocated 133:9 allow 16:9, 44:1, 77:7, 90:19, 90:24, 153:12 allowing 104:2, 135:6 allows 154:25 alluded 68:3 almost 84:15, 85:15, 112:4, 157:4 alone 13:3 along 39:2, 39:19 already 14:21, 15:2, 23:2, 30:11, 49:7, 55:6, 62:8, 62:20, 62:22, 74:11, 83:3, 118:23, 132:19 also 3:24, 11:13, 12:2, 12:3, 12:11, 13:22, 17:9, 19:17, 21:19, 29:11, 33:20, 33:23, 45:5, 45:16, 47:5, 48:4, 50:15, 52:13, 55:7, 59:15, 60:7, 61:14, 67:2, 67:7, 77:16, 80:12, 88:10, 91:6, 91:24, 93:8, 94:14, 95:3, 95:5, 95:23, 96:5, 102:13, 103:16, 106:22,	107:7, 107:11, 109:3, 109:11, 109:14, 114:20, 117:12, 120:17, 127:1, 128:3, 128:22, 130:8, 133:23, 133:25, 134:17, 140:14, 145:4, 147:9, 149:22, 152:10, 160:5, 164:1, 166:4, 166:16 alternative 35:2, 68:13, 68:20, 81:11, 81:16, 83:12, 121:6, 121:8 alternatives 63:22, 68:12 although 12:2, 50:8, 62:4, 86:24, 110:24, 130:7, 133:20, 137:19 alu 113:23, 115:22, 115:24, 116:9 always 18:8, 167:15, 168:6 ambiguous 78:8 and 23:22, 45:24, 50:11, 85:15, 85:21, 131:20, 133:3, 137:15, 166:19 amiss 37:25 among 40:18, 116:14 amongst 43:6 ample 138:4 analysis 32:6, 49:6
---	--	---	---

analyze 57:19 analyzed 12:5 ancestry 29:12 angeles 1:20, 2:21, 5:1, 171:3 angle 30:16 annotation 39:16 another 9:12, 9:17, 39:25, 42:1, 42:23, 44:5, 44:10, 44:21, 46:7, 57:11, 58:5, 58:12, 58:14, 61:2, 61:4, 61:25, 68:1, 68:4, 68:7, 69:25, 70:1, 88:13, 88:21, 97:6, 103:12, 103:20, 141:18, 153:14, 154:8, 154:14, 154:18, 155:3, 157:19, 162:6, 165:2 answer 10:1, 14:17, 14:24, 15:3, 15:4, 15:5, 15:6, 19:3, 23:17, 24:6, 24:22, 25:25, 26:23, 27:5, 28:16, 28:21, 28:25, 38:8, 38:9, 45:8, 46:3, 46:16, 54:15, 65:5, 73:9, 74:4, 78:8, 84:17, 84:18, 98:4,	120:22, 135:15, 143:12, 145:23, 154:2, 154:15, 158:3 answered 14:21, 15:2, 37:23, 38:11, 132:19, 145:11 answers 38:25 antecedent 138:16, 141:6, 141:13, 143:13, 143:16, 143:17, 143:24, 144:5, 144:23, 145:19 any 6:22, 6:23, 9:3, 19:7, 19:9, 20:15, 32:19, 35:9, 35:10, 35:14, 39:24, 40:11, 41:16, 42:1, 42:3, 42:14, 42:21, 42:22, 44:3, 44:4, 45:10, 46:12, 47:10, 48:15, 50:11, 50:24, 51:4, 51:22, 54:12, 57:4, 70:11, 75:22, 82:16, 83:13, 84:2, 84:11, 85:17, 90:23, 91:19, 93:22, 95:1, 99:25, 103:25, 108:8, 108:21, 112:4, 112:11, 113:6, 115:19, 119:22, 121:2, 122:17, 124:10, 124:11, 124:13, 126:12, 127:25, 145:14, 146:14, 146:15, 147:2, 148:6, 148:7,	153:19, 153:20, 155:24, 155:25, 156:6, 161:3, 165:18, 167:13, 167:22, 168:7, 168:13, 172:23, 172:24 anyone 34:20, 85:15 anything 14:23, 33:24, 38:12, 49:5, 53:16, 61:10, 70:3, 73:3, 85:24, 93:23, 105:14, 105:15, 111:16, 129:16, 145:6, 150:10, 152:6, 162:10, 167:19, 168:19 anyway 154:3 anywhere 41:25, 44:3, 65:11 api 26:9, 29:7, 92:23, 93:11, 94:20 apologize 74:14 apologizing 74:7 apparatus 130:15, 131:2 appeal 1:4, 2:4 appears 54:16, 143:24 apple 142:19 application 4:28, 8:7, 8:8, 8:24, 13:14, 15:25, 22:21, 41:21, 43:7, 44:2, 54:7, 56:12, 75:18,	95:13, 96:17, 111:7, 111:14, 112:12, 112:16, 133:14, 147:1, 149:3, 152:25, 153:8 applications 50:5, 61:17, 112:5, 112:12, 119:15 applied 8:14, 9:1, 9:18, 13:24, 16:5, 71:22, 138:12, 139:11, 140:3, 140:17, 147:16 applies 128:3 apply 75:1, 75:11, 143:18 applying 8:6, 9:12, 11:21, 37:19 appreciate 6:19 approach 104:22 appropriate 77:8, 104:20, 105:7, 106:25, 107:13 architected 35:16 architecture 26:24, 27:18, 57:23, 61:8, 128:24, 140:23, 142:4, 142:11, 142:17, 142:18, 144:11, 144:14, 144:20 architectures 35:17, 50:2 area 149:21 areas 123:21
---	--	--	--

argument 117:10 arguments 35:11 arithmetic 115:16, 115:23 around 12:3 arranged 40:11 arrangement 40:14 array 34:6, 34:11, 34:20, 34:21, 34:22, 34:25, 35:4, 35:13, 35:20, 39:21, 39:22, 39:24, 40:3, 40:4, 40:7, 40:9, 40:12, 40:15, 40:16, 40:17, 40:24, 57:24, 130:16, 131:12, 131:19, 133:5, 135:2, 135:12, 135:20, 136:24, 137:9, 158:14, 159:2, 159:5, 159:15, 160:10, 162:4, 163:4, 164:6, 164:22, 165:14 art 8:20, 11:21, 12:5, 12:19, 12:20, 34:21, 36:5, 36:21, 42:9, 43:12, 43:23, 43:24, 53:19, 55:24, 57:1, 69:17, 69:18, 70:20, 71:4, 71:16, 72:15, 72:20, 73:8, 73:11, 73:13, 82:4,	83:20, 84:3, 95:2, 96:12, 99:25, 107:8, 109:2, 113:9, 114:4, 116:16, 119:9, 128:5, 129:22, 150:2 art's 69:4, 129:5 artificial 150:15 aside 71:5, 114:1, 137:25 asked 19:12, 19:15, 19:16, 28:15, 34:24, 38:6, 73:4, 99:4, 141:22, 145:10, 161:8 asking 20:12, 20:14, 26:20, 26:23, 49:3, 49:5, 71:6, 72:2, 72:3, 78:3, 78:5, 78:10, 99:4, 130:11, 151:21, 153:18, 153:25 asks 56:12 aspects 31:3 assembled 90:6 assembler 93:12 assembles 134:4 assembling 107:12 assert 21:9, 166:24 asserted 36:20, 69:15, 70:22	asserting 71:25 assign 27:13 assigned 26:12, 26:14, 124:13, 124:24, 124:25, 125:16, 126:20, 126:22, 126:24, 128:20, 155:14, 155:24 assigning 110:7, 130:4 assignment 127:21 assignments 157:12 assigns 102:11, 104:20, 105:6 associated 10:18, 28:17, 44:20, 44:21, 53:3, 53:4, 53:10, 61:24, 72:12, 74:21, 75:22, 75:24, 78:14, 78:24, 80:4, 83:10, 83:11, 122:14, 155:13, 159:6, 160:1, 160:17 assume 18:13, 50:7, 81:12, 112:14, 116:23, 153:5 assumed 87:12, 162:22 assumption 144:18, 144:21, 153:1 asynchronous 86:15, 86:16 asynchronously 85:19 ati 1:10, 2:10, 3:13, 5:7, 5:19,	170:3 attempting 10:3 attending 5:14 attention 19:20 attorney 172:24 attorneys 5:16 audible 73:23, 90:1 audio 5:23 autonomous 146:22 available 12:22, 47:15, 49:19, 60:24, 60:25, 110:8, 118:20 aware 42:3, 56:18 B b0 125:4, 125:14 b7 125:5, 125:15 back 10:2, 24:7, 29:14, 31:17, 31:23, 32:2, 37:14, 39:21, 51:18, 52:11, 52:15, 58:2, 58:21, 63:6, 74:17, 81:6, 85:23, 86:23, 89:9, 101:24, 102:2, 102:3, 107:10, 110:10, 115:4, 116:13, 117:4, 122:4, 123:4, 124:20, 125:18, 130:23, 138:7, 139:8,
--	--	---	--

145:12, 146:1, 146:10, 156:14, 157:24, 158:10, 163:11, 163:18, 164:21, 169:1 bad 74:8, 82:11, 84:16 bank 48:4 based 12:17, 13:2, 42:11, 45:20, 46:10, 46:20, 48:21, 58:18, 59:24, 60:8, 75:12, 83:3, 155:8, 155:13 basis 13:10, 48:12, 72:6, 138:16, 141:13, 143:13, 143:16, 143:17, 143:24, 144:5, 144:23, 145:20, 157:19 became 6:18 because 18:15, 19:15, 20:23, 30:12, 30:25, 33:21, 35:9, 38:4, 42:24, 44:17, 45:7, 45:23, 48:1, 50:10, 51:16, 54:23, 57:3, 57:7, 57:13, 57:20, 60:2, 61:6, 63:21, 65:21, 69:10, 71:21, 74:9, 78:20, 81:22, 83:7, 84:1, 84:8, 84:15, 84:20, 84:23, 85:4, 91:2, 98:4,	102:18, 103:14, 104:24, 106:13, 111:17, 113:8, 114:17, 114:19, 114:21, 116:10, 118:1, 119:10, 128:19, 133:8, 135:18, 136:19, 137:10, 137:15, 137:17, 138:5, 138:18, 141:8, 143:5, 144:24, 149:10, 150:11, 153:20, 154:16, 160:21, 167:22 become 12:22, 14:11, 27:24, 60:25 becomes 27:22, 27:25 becoming 105:3 been 10:23, 13:23, 16:25, 18:13, 19:16, 33:1, 39:10, 44:1, 46:1, 49:1, 57:24, 61:9, 71:3, 83:16, 108:16, 114:4, 119:8, 120:13, 127:17, 133:7, 136:16, 138:3, 140:10, 145:10, 157:2, 157:6, 168:6 before 1:4, 2:4, 2:23, 31:10, 44:16, 80:11, 82:14, 84:1, 90:13, 104:7, 115:20, 123:22, 140:10, 144:9, 150:22, 156:22, 157:17, 163:21, 172:8, 172:9	beginning 2:21, 5:16, 145:2 begins 5:4, 32:5 behalf 2:19, 5:18, 5:21, 6:2 behind 157:25 being 7:13, 9:7, 13:16, 21:2, 24:12, 30:12, 34:6, 36:4, 39:17, 47:8, 48:20, 65:20, 65:23, 73:4, 75:2, 75:18, 90:18, 90:24, 92:6, 92:14, 97:1, 98:22, 98:23, 99:1, 99:6, 99:16, 100:1, 110:11, 110:12, 110:13, 110:14, 110:16, 110:22, 117:11, 121:16, 122:9, 128:9, 128:20, 129:14, 130:20, 142:10, 142:21, 142:24, 144:11, 153:7, 156:25, 161:8, 165:5 belabor 134:13 believe 7:23, 9:9, 14:21, 15:2, 17:11, 23:2, 27:12, 33:6, 37:23, 51:1, 81:1, 117:25, 165:7 belonging 104:7 below 16:15	beneficial 134:17, 136:12 benefit 132:17, 134:15, 135:9, 136:7, 136:14 benefits 132:11, 136:6, 136:10 besides 9:12, 9:17, 50:24, 60:12 best 26:20, 49:7, 76:12, 157:13 better 62:25 between 10:9, 13:12, 14:5, 27:1, 28:1, 29:6, 33:19, 40:4, 40:9, 41:23, 42:15, 43:10, 43:17, 43:18, 52:4, 57:22, 59:19, 72:16, 86:25, 87:5, 128:8, 157:18, 158:1 bidirectional 44:25, 45:3 big 131:12, 135:2, 135:20, 136:24, 137:9 binary 114:17 binning 158:13, 159:23, 160:3, 160:4, 160:12, 160:14, 164:8, 164:23, 166:2 binnitization 160:3 bit 7:1, 46:19,
--	---	---	---

63:24, 74:3, 88:21, 104:4, 114:22, 124:21, 134:12, 141:8, 156:22 biu 117:17 black 139:6 block 25:20, 51:20, 54:19, 54:20, 54:21, 54:22, 55:12, 55:14, 55:15, 56:15, 56:16, 94:5, 94:25, 101:11, 127:16, 134:6, 137:6, 137:25, 139:25, 142:1, 142:18, 158:17, 160:22, 161:15, 164:15, 164:19 blocks 49:20, 55:25, 56:7, 123:24, 124:2, 124:4, 125:19, 126:8, 126:14, 127:11, 127:15, 127:25, 128:3, 128:10, 128:21, 129:2, 129:3, 129:9, 129:14, 129:25, 130:19, 130:20, 131:3, 131:4, 131:6, 131:9, 131:14, 131:18, 131:24, 132:5, 133:13, 133:22, 135:4, 135:8, 136:20, 137:12, 137:20, 138:13, 139:2, 139:12, 165:12, 165:18, 165:22, 166:1 blood 172:23	blue 143:1, 143:4 bmw 142:24, 143:1, 143:3 board 1:4, 2:4 bond 4:35 bookkeeping 25:8, 25:12 books 25:14 both 23:3, 30:24, 47:22, 55:2, 63:18, 109:13, 126:18, 148:4, 149:11, 154:25, 168:10 bottom 47:21, 103:20, 144:6, 166:22 box 69:20, 71:21 brake 146:23 branch 116:5, 116:7 break 31:6, 31:8, 62:24, 88:14, 89:3, 122:23, 136:12, 158:7, 163:10, 163:13 breaks 89:5, 136:21 bridge 41:18, 43:1, 45:5, 51:18, 51:24, 52:6 bring 59:14, 60:17, 61:2, 148:20, 148:23, 149:3, 149:6, 151:8, 151:18 bringing 148:19	brings 82:5, 82:7 broke 137:10, 137:25 broken 137:7, 137:8 brought 44:8, 59:25, 151:11 bucket 148:13 buffer 39:6, 39:10, 39:12, 81:3, 81:9, 81:19, 81:20, 82:2, 82:8, 82:18, 82:25, 83:1, 83:4, 83:5, 83:11, 83:14, 84:5, 84:6, 84:7, 84:11, 84:22, 96:6, 96:7, 104:12, 117:8, 117:14, 117:24, 118:3, 118:6, 118:9, 118:11, 118:14, 165:6 buffers 81:24, 82:13, 83:21, 84:4, 84:19, 122:6, 122:10, 122:14, 122:16, 122:21 building 10:24, 10:25 built 115:20, 115:22 bunch 75:16, 96:15, 147:19 bus 117:18, 125:11 bus-based 40:21 busses 40:6	busy 84:24 butler 3:25, 5:12 by; 79:5 bypass 137:2, 137:8 bypassing 51:25 C cache 91:23, 93:8, 116:5, 125:5, 125:12, 158:15, 164:14, 164:18 calculating 126:24 california 1:20, 2:21, 3:19, 5:1, 171:1, 171:8, 171:11, 172:3, 173:10 call 70:9, 73:2, 78:24, 79:4, 105:9, 108:15, 135:13 called 27:7, 33:17, 54:20, 55:14, 55:18, 79:3, 83:1, 113:13, 131:15 calls 70:10, 120:19 came 39:18, 85:3, 136:3, 136:19 camera 101:7 can't 45:13, 52:11, 82:16, 152:22, 154:15 cannot 6:23, 14:22,
--	--	--	---

14:24, 15:4, 15:6, 18:22, 19:6, 19:13, 21:10, 24:22, 42:2, 48:22, 50:1, 55:2, 78:8, 84:23, 101:12, 112:6, 150:14, 154:2, 154:24 capabilities 127:10, 127:14, 127:16 capability 43:25, 126:8, 134:11 capable 60:7, 96:2, 112:4, 116:17, 128:18, 130:7, 130:8, 130:20 captured 76:20 car 76:5, 76:8, 76:18, 76:19, 142:24, 143:1, 143:8, 146:22 care 156:4 carefully 24:23, 26:22, 46:3, 50:9, 57:18, 157:23, 164:3 carmel 3:17 carried 110:5 case 1:14, 2:14, 5:8, 13:9, 14:11, 21:24, 22:1, 33:7, 35:10, 37:19, 49:17, 69:9, 83:9, 135:15, 137:23, 139:10,	142:12, 142:25, 144:20, 144:21, 153:5, 153:7, 170:3, 170:6 cases 14:14, 24:15, 62:12, 152:16 caught 120:21 cause 82:12 causing 61:12, 100:22 cb 117:11 centered 101:7 certain 40:8, 60:23 certainly 13:17, 34:21, 62:13, 70:12, 77:10, 92:13, 122:19, 122:20 certified 2:24, 6:7, 7:12, 18:25, 172:2 certify 171:6, 172:4, 172:12, 172:14, 172:18, 172:22 cetera 51:9, 96:5, 96:9, 115:7 challenging 29:8 chance 102:8 change 30:2, 75:11, 145:15, 153:13 changes 142:9, 153:21 changing 167:17 channel 158:13, 159:19,	159:25, 160:17, 162:7, 163:5, 164:6, 164:22, 165:19, 165:23 channels 148:21 characteristics 30:7 charge 133:12 check 47:18, 104:5 child 27:2, 153:3 children 26:9 chipset 117:18 choose 41:20 circuit 86:12 circuitry 93:8 citation 95:18, 124:10, 125:2 citations 42:11, 124:7 claim 8:4, 12:25, 13:25, 15:20, 20:19, 20:21, 21:8, 21:15, 42:13, 58:25, 59:4, 59:8, 69:3, 69:15, 70:21, 78:13, 80:25, 113:12, 113:13, 119:1, 119:21, 119:25, 120:1, 146:3, 151:21, 161:23, 162:6, 162:14 claim's 158:17 claimed 32:16, 71:19,	71:25, 73:4, 73:7, 121:7, 121:9, 136:16 claiming 59:23 claims 32:21, 42:8, 70:22, 100:14, 118:9, 120:19, 123:12, 153:22 clarification 38:1, 63:17, 88:7, 93:18, 125:8 clarifies 108:6, 151:19 clarify 97:23, 156:23, 167:21 classification 149:10 clear 34:20, 48:5, 58:1, 67:6, 78:11, 92:24, 94:16, 95:15, 96:25, 99:17, 108:9, 109:17, 109:24, 112:9, 128:15, 136:4, 140:22, 141:18, 145:18, 156:20 clearly 21:22, 47:2, 48:14, 48:17, 56:20, 83:8, 116:16, 118:6, 161:13 clock 85:10, 85:19, 85:20, 85:21, 85:22, 86:2, 86:9 clocks 86:9, 86:18 close 45:2 closest 108:14
---	---	--	---

code 22:16, 23:4 codes 160:7 cohn 3:14 collection 34:12, 34:13, 34:17, 35:1, 50:16, 68:18, 69:7, 69:11 color 76:11, 143:8 column 46:24, 48:14, 52:19, 52:22, 53:24, 54:3, 56:25, 57:14, 58:19, 64:6, 64:9, 64:18, 64:22, 64:24, 65:6, 65:8, 65:12, 65:16, 78:22, 90:5, 93:19, 94:4, 94:22, 94:24, 94:25, 95:14, 95:15, 95:25, 99:8, 102:17, 102:21, 103:6, 103:20, 104:18, 105:21, 110:1, 115:14, 118:25, 126:6 com 3:10, 3:11, 3:21 combination 17:25, 68:6, 69:22, 70:23, 71:22, 72:17, 148:1 combine 149:23 come 19:13, 31:17, 35:9, 35:10, 45:25, 51:18,	64:1, 67:17, 71:17, 77:24, 82:15, 84:25, 93:3, 94:9, 99:16, 99:25, 108:14, 109:7, 111:20, 115:4, 120:4, 125:7, 134:1, 134:6, 137:16, 139:8, 146:10 comes 11:23, 13:15, 39:8, 39:13, 60:3, 60:16, 74:10, 100:6, 100:18, 106:1, 155:21, 156:1 coming 26:8, 26:11, 29:6, 67:2, 67:10, 76:8, 84:25, 92:20, 92:21, 97:25, 110:19, 111:3, 113:11, 125:11, 128:20, 134:23, 141:25, 142:1, 156:10 command 81:3, 81:9, 81:19, 82:2, 82:25, 83:1, 83:13, 122:6, 122:9, 122:14, 122:16, 165:5 commands 64:9, 65:13, 65:20, 66:9, 66:16, 66:23, 67:3, 67:9, 67:16, 88:4, 93:1, 93:22, 95:19, 108:13 commenced 104:8 commencement 172:8	commencing 172:16 comment 16:7 common 43:16, 56:13, 61:4, 82:16 communicate 41:19, 43:6, 43:17 communicates 44:4 communicating 42:1, 42:22 communication 19:9, 42:15, 42:17, 42:18, 43:20, 44:1 community 147:22 compiling 24:12 complainant 166:19 complete 95:12, 97:11, 157:13 completed 14:10, 155:13 complex 21:24, 25:15, 137:14 complicated 11:14, 51:21 component 32:25, 33:2, 92:1, 115:24, 116:17, 148:6, 161:4, 162:25 components 10:18, 26:3, 73:7, 82:23, 83:22, 92:2, 95:7, 112:20, 115:11, 117:19, 125:20, 158:15, 159:12, 159:15, 161:18, 163:24,	164:4, 164:12 composed 11:7, 12:7, 20:2 comprise 10:19, 50:15, 120:10, 120:19, 151:25, 155:10 comprises 103:22, 115:15, 120:2, 120:3, 130:15, 131:3 computation 33:20, 114:7, 114:8, 115:17, 116:5, 119:19, 151:13, 151:14 computational 10:17, 37:11, 68:17, 69:13, 93:16, 94:13, 102:14, 102:24, 114:12, 119:11, 119:12, 119:14, 119:19, 120:10, 131:5, 133:9, 147:3, 147:9, 147:12, 148:13, 148:14, 149:15, 150:6, 150:12, 150:14, 150:18, 151:3, 151:6, 151:12 computations 148:17 compute 114:2, 116:18, 119:18, 149:21, 149:22 computer 85:13 computing 61:7, 61:15, 115:25, 119:24, 120:1, 147:22, 155:15 conceive 28:9
---	---	--	---

concept 28:7, 77:24, 124:8, 143:19, 143:23, 146:6, 149:14, 154:6 concepts 86:21, 93:17 concerned 8:19, 101:14 concisely 134:14 conclude 48:22, 130:5 concluded 169:11 concludes 169:8 conclusion 100:1, 134:6 concurrent 147:20 concurrently 128:16 conditional 116:6 conditions 24:25, 75:14, 83:24, 146:20, 152:24 configuration 40:18, 73:6, 100:5, 100:10, 100:17, 112:13, 134:10 configure 60:8 configured 32:10, 44:1, 58:17, 59:24, 60:6, 96:7, 100:13, 109:19, 111:25, 123:13, 128:17, 157:12, 166:25, 167:17, 167:18, 167:25 configures 101:3 confirm 18:22, 19:13,	20:12, 20:13, 21:1, 49:8, 118:24 confuse 86:15 confused 80:7, 117:22, 155:17 confusing 143:10 connect 140:18, 140:20 connected 39:24, 40:6, 41:15, 41:18, 81:3, 81:9, 81:19, 81:21, 122:9, 165:5 connection 14:5, 34:14, 41:23, 52:4, 52:5, 52:14, 52:18, 74:8, 74:14 connections 40:4, 40:5, 40:18, 40:19, 117:16 connectivity 40:9 connects 34:1 consider 24:23, 26:22, 36:2, 46:2, 68:9, 70:22, 71:11, 72:9, 72:10, 75:23, 77:19, 80:12, 142:13, 149:14, 151:3 considered 9:14, 13:6, 17:3, 35:8, 47:14, 50:10, 71:9, 79:23, 148:19 considering 72:7	consistent 13:25 constant 96:6, 96:15, 104:12, 117:8, 117:13, 117:14, 117:24, 118:3, 118:4, 118:6, 118:9, 118:11, 118:14 constants 56:4, 74:20, 75:16, 75:20, 79:1, 93:20, 96:6, 96:16, 96:17, 96:19, 96:22, 98:13, 99:9, 99:13, 104:12, 116:12, 118:15, 149:1 constituent 153:11, 156:25, 157:5 constitute 33:8, 40:12, 89:19, 158:17 construct 17:4, 35:3 constructed 112:6 construction 8:4, 8:14, 8:21, 9:1, 9:10, 11:21, 13:25, 16:9, 20:19, 20:21, 21:8, 21:15, 34:5, 39:22, 69:20, 70:19, 81:13, 134:7, 161:9, 162:14, 162:20, 162:23 constructions 12:25, 15:20, 81:11 constructs 17:17, 22:2, 22:3, 22:5,	23:20 construe 162:17 construed 15:23, 162:15 consumers 83:25 consuming 105:3 contain 161:3 containing 82:2 contains 59:3, 59:7, 93:7, 160:22 contents 52:10, 58:13 context 9:7, 9:8, 9:11, 9:16, 9:22, 11:12, 11:20, 11:22, 23:3, 34:25, 36:4, 36:18, 36:20, 37:18, 37:22, 38:13, 50:7, 55:19, 59:10, 59:11, 59:13, 59:17, 59:18, 59:19, 59:20, 60:1, 60:2, 60:3, 60:7, 60:10, 60:21, 61:2, 61:12, 61:20, 61:22, 61:23, 61:24, 61:25, 62:4, 62:6, 62:13, 62:14, 62:19, 75:4, 76:17, 77:1, 77:3, 79:3, 79:5, 79:13, 79:14, 80:5, 82:6, 99:7, 103:23, 104:5, 104:6, 104:8, 105:24,
---	---	---	---

106:8, 112:25, 113:20, 113:21, 114:3, 118:16, 122:5, 150:1 contexts 104:2, 104:3, 136:10 contextual 96:21 continue 7:2, 18:24, 31:8, 136:22, 145:21 continues 96:11, 104:18, 143:5 continuing 164:20 contributes 33:24 control 23:19, 39:8, 85:12, 91:22, 92:11, 92:14, 92:20, 93:4, 93:5, 93:7, 93:9, 94:19, 100:22, 101:2, 102:22, 103:4, 104:7, 104:13, 104:19, 105:5, 105:17, 105:25, 106:6, 106:11, 106:18, 107:1, 107:11, 107:22, 108:13, 109:3, 109:13, 109:22, 110:7, 110:14, 111:4, 126:2, 126:21, 128:14, 134:3, 137:14, 138:24, 146:7, 146:11, 146:12, 146:13, 146:17, 147:1, 147:3, 147:5, 147:15, 147:17, 147:24, 147:25, 148:6,	148:14, 149:9 controlled 65:18, 65:25 controller 103:1, 117:9, 117:11, 146:17, 158:14, 159:4, 159:5, 159:15, 159:19, 159:25, 160:11, 160:17, 162:5, 162:7, 163:4, 163:5, 164:6, 164:7, 164:22, 165:15, 165:19, 165:23 controllers 26:5 controlling 99:2 controls 102:23, 106:23, 107:22 conversely 12:9, 155:12 converted 22:16 convey 35:3 coordinate 30:3, 30:4, 31:2, 77:7 coordinated 43:8, 57:24 coordinates 30:21, 75:11 copy 7:22, 7:24, 16:16, 21:13, 39:18 core 32:10, 32:17, 32:21, 32:23, 33:3, 33:5, 33:9, 33:12, 33:22, 34:3, 34:6, 34:16, 37:19, 38:14, 39:22, 40:11,	59:23, 63:13, 64:15, 64:16, 65:7, 65:12, 65:13, 65:17, 65:18, 65:20, 65:21, 65:24, 66:1, 66:4, 66:7, 66:14, 66:21, 66:22, 67:1, 67:4, 67:5, 67:9, 67:11, 67:13, 67:19, 67:21, 67:25, 68:13, 68:15, 68:17, 69:12, 70:15, 71:24, 72:25, 73:15, 82:22, 83:19, 84:6, 84:11, 84:13, 85:1, 86:25, 87:6, 87:7, 87:20, 87:21, 87:24, 93:15, 93:16, 94:13, 102:14, 102:15, 102:24, 103:11, 105:10, 107:23, 108:1, 108:10, 108:15, 119:13, 119:25, 120:7, 121:21, 123:13, 123:24, 124:9, 138:24, 165:13 cores 33:20, 64:18, 65:2, 68:18, 68:24, 70:16, 72:14, 84:21, 84:23, 84:24, 87:1, 87:17, 87:25, 121:23, 121:25 corp 1:7, 2:7, 3:3, 5:6, 170:3 corrections 170:14	correspond 23:23, 154:19, 154:21, 154:23 corresponded 111:13 correspondence 11:15, 27:1, 29:9, 72:16 correspondences 25:10, 25:11 corresponding 11:25, 29:25, 55:20, 68:16, 130:9, 134:10, 162:5 could 5:24, 8:23, 9:3, 9:11, 9:18, 10:11, 11:1, 11:4, 11:11, 11:15, 11:17, 11:18, 11:22, 11:24, 12:7, 14:6, 14:7, 14:11, 14:12, 14:14, 17:17, 18:9, 18:14, 19:17, 20:16, 21:22, 23:5, 25:17, 25:24, 27:15, 28:9, 29:11, 32:2, 35:1, 35:3, 35:21, 35:23, 36:7, 36:8, 36:22, 38:2, 38:10, 40:3, 40:5, 40:16, 40:18, 40:20, 40:25, 41:2, 41:24, 43:19, 43:21, 48:4, 49:11, 53:20, 54:4, 56:1, 56:10, 57:19, 60:20, 61:14, 63:9, 69:7, 77:15, 77:16,
---	---	---	--

82:9, 86:8, 89:12, 91:2, 91:11, 101:7, 111:24, 123:6, 126:19, 128:16, 128:19, 129:3, 130:2, 131:9, 134:21, 135:21, 136:9, 136:13, 136:21, 136:22, 137:1, 137:2, 139:4, 141:1, 145:14, 147:9, 148:8, 151:5, 154:17, 154:19, 155:2	crash 84:15 create 133:9 created 24:8 critical 138:20, 139:9 csr 1:24, 1:25 custom 147:25 cycle 39:8	32:4, 34:4, 41:7, 42:4, 58:22, 59:2, 68:12, 74:18, 79:15, 89:15, 101:17, 121:6, 124:23, 146:1, 146:2, 151:23, 154:5, 158:11, 160:10, 162:21, 163:23, 164:3, 165:10, 165:11 declarations 45:25 declared 6:11 decode 113:22, 116:2 decoded 23:20 decoder 112:22 decoding 39:15, 115:6 decomposed 13:17, 17:5, 18:3, 20:1, 20:7, 20:10, 21:4, 21:10, 21:17, 24:3, 24:20, 26:10, 29:23 decomposes 23:13 decomposing 18:18, 18:20, 19:22 default 153:1 define 35:20, 69:7, 78:3, 78:10, 79:20, 114:5 defined 79:6, 114:4 defining 66:10, 66:16 definitely 19:16, 84:13,	85:23, 85:24, 117:20, 160:23 definition 8:5, 8:6, 8:11, 8:16, 9:3, 9:12, 9:17, 16:4, 36:12, 37:18, 38:13, 40:11, 61:6 delivered 110:4 demolishing 11:1 depending 9:6, 11:12, 23:7, 25:23, 30:14, 54:7, 76:20 depends 23:17, 27:18, 27:19, 88:17, 114:22 depicted 51:11, 54:11, 55:11 depicts 44:19, 70:8 depo 170:10 deponent 170:12 depos 5:12, 6:5 deposition 1:18, 2:18, 3:26, 5:5, 5:13, 6:20, 7:10, 15:9, 169:8, 169:11, 170:1, 172:7, 172:9, 172:13, 172:14, 172:19 depositions 19:8, 45:24 describe 41:25, 42:21, 50:23, 57:10, 65:12, 81:7,
counsel 5:14, 6:3, 7:12, 102:2, 169:7, 172:13, 172:20 count 135:11 counter 104:17 county 171:3 couple 55:5 course 47:1, 83:2, 88:9, 92:21, 116:2, 117:15, 125:7, 147:5 court 6:4, 19:8, 141:9 covid 31:11 cpu 59:18, 61:22, 61:25, 100:23, 115:19, 115:22, 116:20, 117:15, 117:18, 117:20, 117:21 cpus 61:5	D data" 159:20 data-dominated 148:8 data-driven 104:22 data-intensive 148:8, 148:16 date 5:9, 170:10, 170:30 day 31:10, 31:14, 171:10, 173:4 ddr 148:21, 151:8 ddr4 148:22 deaf 156:3 deal 57:24 dealing 106:3, 106:19 deals 109:13 decide 116:6, 146:23 decision 95:5 declaration 4:9, 7:19,		

91:9, 105:23, 119:7, 126:18, 128:7, 144:10, 165:4 described 14:4, 24:18, 47:10, 90:23, 99:6, 104:1, 110:11, 110:14, 110:16, 115:11, 116:11, 122:9, 127:17, 131:2, 133:23, 137:24, 145:6, 158:4 describes 53:25, 70:8, 71:14, 76:17, 82:1, 90:18, 99:8, 111:6, 112:3, 117:23, 120:6, 126:13, 128:14 describing 56:19, 99:11, 125:3, 127:7, 127:19, 127:21, 133:4, 142:3, 143:7, 143:8 description 4:8, 40:7, 46:5, 48:6, 65:7, 76:2, 92:10, 95:1, 95:3, 96:25, 99:15, 109:18, 116:15, 125:10, 128:1, 128:15, 159:24 descriptions 46:11, 127:2 design 50:3, 83:6, 83:21, 84:9, 85:5, 85:6, 85:7, 85:9, 86:8, 86:13, 132:11, 133:7, 134:1, 135:16,	136:15, 136:17, 136:19 designation 53:15 designed 61:9, 148:3, 155:22 designs 85:16, 137:14 desktop 16:17 detail 25:13, 152:14 detailed 51:22 details 24:1, 24:18, 37:4, 51:20, 83:17, 87:11 determination 4:32, 4:34, 12:24, 24:24, 27:1, 95:4, 166:11 deterministic 111:22 device 17:21, 17:23, 75:3 diagram 51:21, 94:5, 94:25, 142:1, 142:18 dictate 85:11 diego 3:19 difference 135:11 different 10:18, 14:13, 14:16, 17:24, 19:6, 21:23, 22:3, 22:5, 23:8, 23:24, 23:25, 24:13, 26:4, 35:17, 35:18, 35:23,	37:10, 40:20, 42:8, 43:9, 50:17, 53:23, 55:25, 59:17, 59:18, 59:20, 60:8, 60:12, 63:21, 70:2, 70:10, 70:25, 74:25, 76:10, 76:11, 76:23, 76:24, 81:10, 86:5, 86:20, 91:4, 94:10, 100:2, 106:2, 106:18, 109:8, 109:20, 110:21, 110:22, 125:16, 126:9, 128:20, 129:4, 129:9, 129:25, 130:3, 130:4, 131:8, 131:9, 131:14, 131:15, 131:17, 131:18, 131:22, 132:14, 132:23, 133:6, 135:6, 135:13, 135:19, 137:13, 137:22, 138:1, 138:2, 139:16, 139:17, 143:4, 143:7, 146:14, 146:15, 149:20, 151:24, 152:12, 152:21, 154:23, 156:8 differently 150:23 difficult 24:14, 29:4 difficulty 28:17 direct 41:22, 43:19, 52:4, 52:5, 117:16 directed 110:13 directly 81:3, 117:12,	117:13 disagree 57:12, 57:13, 145:3, 145:4, 161:6, 162:12 disagreement 97:21 disambiguous 25:12 disappear 85:2 disclose 57:1, 58:4, 58:11, 152:3 discloses 58:5, 58:13, 58:24, 59:3, 59:8, 91:20, 146:3 disclosing 42:7, 92:6 disclosure 52:18, 55:5, 56:22, 59:11, 66:5, 72:4, 81:17, 92:13, 92:24, 93:22, 94:17, 98:1, 104:2, 109:2, 112:10, 115:13, 122:17, 122:18, 130:2, 140:22, 155:8 disclosures 56:23, 102:9 disconcerting 74:3 disconnect 38:11 discuss 18:19, 32:9, 32:10, 89:5 discussed 9:8, 28:10, 36:4, 67:16, 90:13, 125:4, 140:10, 142:10, 143:20, 144:12,
--	--	--	--

149:14, 152:13, 159:12 discussing 83:16, 115:6, 120:13, 140:25, 141:1, 159:14 discussion 16:18, 70:8, 108:16, 108:21, 153:19, 160:10, 162:13 dispatch 86:14, 86:16 dispatched 13:19, 84:23, 120:5 display 17:23, 30:4, 30:6, 30:15, 75:3 displaying 17:21, 29:17, 29:23 disputing 98:10 distinct 51:11, 81:3, 81:9, 82:24, 87:17, 87:21, 122:6, 122:9, 122:14, 124:4, 165:5 distinction 140:21 distinctions 150:15 distinguish 93:17 distribution 33:5, 64:10, 64:14, 64:20, 65:3, 66:3, 66:11, 66:15, 66:25, 67:19, 67:25, 68:15, 71:23, 73:21, 73:25, 83:18, 84:5, 84:12,	84:19, 84:20, 85:1, 87:1, 87:5 distrubution 63:12 divide 114:16 divided 129:13, 130:19 dividing 136:8 division 114:19 dma 125:11 doable 24:9, 24:10, 29:10 doctor 101:5 document 7:9, 7:13, 15:8, 16:18, 41:3, 80:10, 88:23, 123:7, 139:20, 166:7, 167:18, 167:20 doing 11:3, 17:22, 26:1, 26:20, 29:3, 39:12, 42:9, 43:22, 49:6, 61:1, 88:12, 92:20, 94:8, 97:10, 97:14, 97:21, 112:8, 116:17, 119:13, 120:7, 130:3, 138:2, 149:22, 151:13, 151:14, 151:15 dokhanchy 3:16, 4:5, 5:17, 5:18, 6:15, 7:14, 7:17, 15:12, 31:5, 31:16, 32:1, 37:13, 37:17, 41:5,	58:8, 62:24, 63:8, 80:9, 88:12, 88:20, 89:4, 89:11, 101:13, 101:19, 102:1, 102:5, 102:6, 122:22, 122:25, 123:6, 123:9, 125:13, 163:12, 163:20, 166:9, 168:12, 168:21, 169:5 done 8:23, 27:14, 43:12, 49:10, 86:14, 92:10, 97:11, 99:5, 99:6, 100:3, 112:21, 137:4, 137:5, 164:6, 165:7, 168:17 door 156:5 down 15:18, 16:16, 16:18, 16:21, 17:1, 27:20, 32:9, 37:14, 41:10, 48:2, 65:6, 66:16, 66:18, 66:19, 66:24, 72:13, 72:25, 80:17, 89:18, 92:7, 92:15, 94:9, 94:18, 100:3, 106:23, 113:5, 124:21, 136:13, 141:8, 144:25, 153:16, 166:15 download 7:22, 21:6, 21:12 dr 4:9, 5:5, 6:16, 7:18, 15:13, 16:11, 19:19, 31:6, 32:4,	41:6, 52:9, 57:9, 63:1, 63:11, 88:13, 89:14, 101:14, 102:7, 104:24, 107:17, 108:8, 110:12, 120:22, 123:10, 141:7, 141:13, 163:21, 166:13, 169:9 dram 148:18 drawing 47:21, 69:20, 71:22 drawn 30:12 driver 92:22, 93:4, 93:11, 94:20, 99:2, 100:23, 111:6 drives 110:25 driving 76:5 drop 85:2, 85:4 dropbox 7:23 drops 137:9 dual-entry 83:5 due 62:14, 150:9 duly 172:8 dumb 155:23 uplicative 74:6 duration 89:5 during 149:23, 151:5 dynamic 148:18
--	--	--	---

E			
e 66:8, 103:25, 104:14, 104:22 each 10:20, 23:25, 27:24, 32:15, 35:5, 35:18, 41:15, 41:20, 44:8, 45:6, 70:13, 78:14, 78:23, 81:2, 81:4, 81:9, 81:17, 82:25, 83:7, 83:10, 83:11, 86:20, 91:11, 92:18, 94:11, 104:3, 104:12, 104:16, 122:10, 124:12, 131:4, 131:6, 131:16, 131:22, 132:5, 132:6, 132:7, 132:18, 133:11, 134:13, 135:4, 135:17, 138:10, 139:2, 139:25, 159:6, 159:7, 160:1, 160:18, 161:17, 165:4, 166:24, 167:1, 167:25 earlier 17:9, 31:14, 68:3, 122:4, 143:24, 149:14, 157:7, 158:9, 159:13 early 31:10 easy 153:17 efficiency 22:10, 149:5 efficient 118:18, 136:3 efficiently 22:23	eight 124:23, 125:4, 131:15, 131:22, 132:3, 133:9, 133:11, 133:15, 135:4, 135:5, 135:6, 135:12, 135:13, 135:16, 135:18, 137:12, 137:20, 137:22, 138:1, 139:17, 142:6 either 7:22, 43:19, 52:17, 88:19, 136:16, 147:23, 149:8, 149:9 elaborate 100:10 element 25:20, 26:13, 26:14, 33:17, 35:24, 36:1, 37:20, 39:3, 40:15, 61:3, 86:9, 97:2, 134:24, 137:7 elements 26:6, 26:15, 26:16, 30:10, 34:7, 34:18, 34:22, 35:5, 35:14, 35:15, 35:21, 35:22, 39:23, 40:4, 40:8, 40:9, 40:10, 40:15, 40:24, 42:8, 68:17, 106:2, 120:10, 129:2, 129:12, 129:24, 130:17, 130:18, 131:13, 132:12, 134:17, 134:19, 134:22, 135:2, 135:4, 135:5, 135:18, 135:25, 136:25, 137:2,	164:3, 168:9 else 20:24, 27:11, 28:4, 105:15, 130:25, 145:6 else's 53:13 elsewhere 44:16, 81:7 embedded 146:18 embodiment 94:5, 104:16 embodiments 66:7, 104:15 employed 172:24 enable 42:18, 60:7 enables 135:18 encoding 39:16 end 14:15, 22:7, 39:7, 67:18, 67:25, 97:7, 97:20, 99:19, 99:20, 138:10, 156:13, 156:14 ending 2:22 ends 97:9 engine 10:17, 33:1, 68:4, 68:7, 69:9, 69:25, 70:1, 70:12, 71:12, 71:13, 72:17, 73:3, 73:5, 81:2, 82:22, 82:25, 120:18, 122:10, 131:5, 131:6, 161:24, 166:24, 167:25 engine; 81:4	engineered 111:22 engineering 24:7 engines 32:24, 33:6, 33:8, 33:12, 33:25, 47:5, 63:11, 63:19, 68:5, 68:14, 68:21, 68:25, 69:10, 69:25, 70:9, 70:24, 71:9, 71:10, 71:25, 72:21, 73:6, 80:22, 80:24, 81:8, 81:11, 81:13, 82:20, 82:24, 83:11, 89:19, 92:4, 100:15, 105:12, 105:13, 105:16, 109:4, 120:18, 121:7, 121:9, 122:5, 158:12, 161:25, 164:13, 164:17, 164:18, 164:21, 165:1, 165:5, 167:16, 168:4, 168:7 engines" 158:18 enough 14:7, 18:10, 18:15, 19:18, 66:4, 66:5, 83:25, 138:3 entirely 110:16 entity 100:6, 100:18 entry 91:25 environment 60:11 environmental 146:19

envision 136:5 envisioned 112:6 epu-controlled 125:11 equivalent 82:23 equivalently 34:15, 34:17 errata 170:1 erroneous 85:6 esq 3:5, 3:16 establish 29:9, 72:15, 161:2 et 51:9, 96:4, 96:9, 115:7 eu 91:22, 92:11, 92:14, 92:20, 93:4, 93:5, 93:7, 94:19, 97:9, 99:2, 100:22, 101:2, 102:10, 102:22, 103:3, 103:7, 103:22, 104:6, 104:13, 104:18, 105:5, 105:16, 105:22, 105:25, 106:6, 106:11, 106:17, 106:20, 106:25, 107:11, 107:22, 108:7, 108:12, 109:3, 109:12, 109:22, 110:6, 110:14, 111:1, 111:4 eudp 115:15, 116:16, 116:19, 117:8, 117:12, 118:2, 118:24, 119:7,	119:10, 119:23, 120:5, 120:9 eup 113:8 even 13:9, 21:25, 23:23, 26:22, 36:24, 52:1, 65:19, 72:15, 115:19, 115:20, 134:18, 136:24, 137:23, 145:19, 148:16, 161:13 evening 31:12 events 146:20 eventually 103:10, 106:24 ever 50:22, 52:9, 77:24, 115:20, 117:10, 120:9, 137:16, 138:5, 163:24, 167:13 everybody 169:6 everyone 113:3, 113:8, 115:17 everything 20:23, 33:11, 145:12 exact 119:22 exactly 17:16, 26:11, 92:19, 114:23, 133:21, 139:22, 152:11, 153:7, 153:23, 157:25 examination 4:5, 6:14 examined 6:12 example 10:24, 12:23, 17:20, 19:10,	19:22, 21:22, 23:1, 23:14, 23:23, 25:7, 26:17, 29:17, 30:20, 34:19, 37:11, 38:7, 38:10, 38:15, 39:6, 42:25, 43:8, 43:14, 49:1, 50:20, 53:4, 59:17, 60:22, 61:25, 65:8, 77:6, 84:3, 95:23, 102:15, 102:20, 104:1, 105:20, 106:5, 114:9, 115:1, 126:20, 128:25, 136:11, 138:8, 138:12, 138:17, 139:9, 139:10, 139:13, 139:21, 139:22, 140:2, 140:3, 140:7, 140:8, 140:9, 140:11, 140:12, 140:15, 140:16, 141:3, 141:4, 141:14, 141:17, 141:24, 141:25, 142:2, 142:3, 142:9, 142:17, 142:20, 142:24, 143:14, 144:5, 144:6, 144:11, 144:13, 144:17, 144:24, 144:25, 145:7, 146:12, 146:16, 146:21, 146:25, 147:19, 148:25, 152:20, 154:7 example; 97:8 examples 19:5, 30:1, 36:23, 38:23, 80:24, 91:4,	98:20, 139:14, 147:2, 147:21, 152:15 except 33:16 excerpt 48:13, 48:17, 49:7, 50:14, 51:1, 52:19, 53:11, 54:3, 56:25, 58:19, 67:7 excerpts 42:10, 163:11 exclude 33:25, 45:19, 122:15, 122:20 excludes 77:16 exclusive 55:22 excuse 8:8, 16:1, 18:18, 21:10, 68:4, 95:14, 124:23, 153:13 execute 23:21, 33:22, 60:8, 65:22, 65:25, 69:1, 116:9, 123:14, 125:20, 125:24, 156:13 executed 8:8, 9:24, 10:6, 10:12, 10:22, 10:23, 11:25, 13:15, 13:16, 13:20, 13:22, 13:23, 14:9, 15:25, 17:18, 22:6, 27:20, 29:7, 59:15, 62:15, 64:19, 66:11, 93:15, 102:14, 106:13, 121:21, 134:5, 146:18,
---	---	---	--

152:10, 157:1, 157:3, 171:10 executing 66:22, 67:12, 123:15, 124:8, 128:16 execution 10:13, 10:23, 13:19, 22:10, 60:17, 60:23, 70:15, 94:14, 95:11, 97:17, 97:20, 103:21, 103:24, 104:3, 104:13, 104:21, 105:7, 105:9, 105:22, 105:25, 106:2, 106:3, 106:15, 106:20, 109:9, 110:4, 110:9, 110:25, 111:19, 112:19, 113:22, 115:15, 116:8, 121:10, 130:5, 134:9, 148:6, 148:7, 157:9 exercise 71:22, 145:16 exhibit 4:8, 4:9, 4:14, 4:19, 4:21, 4:24, 4:26, 4:31, 7:15, 7:16, 7:18, 15:7, 15:11, 15:13, 21:7, 29:14, 32:3, 37:14, 37:15, 41:2, 41:4, 41:6, 58:23, 63:9, 80:8, 80:10, 80:19, 88:24, 88:25, 89:13, 89:14, 123:7, 123:8, 123:18, 165:10, 166:6, 166:8,	166:12, 166:16 exhibits 37:16 exist 57:2, 57:22, 83:15, 122:18 existed 30:11, 145:1 existing 11:1, 70:24, 142:6 exists 29:6, 43:25, 52:1, 52:18, 52:21, 56:16, 99:3, 121:22 exit 156:13 exotic 149:24 expect 84:2, 84:3 expense 157:14 experience 31:11, 77:25 expert 4:14, 71:15, 133:3 experts 35:11, 45:25, 50:12, 133:3, 137:16, 153:19 explain 15:23, 16:15, 17:2, 22:4, 22:25, 36:21, 37:12, 106:8 explanation 83:3, 95:15 explicit 92:10, 130:2 explicitly 21:18, 91:20, 127:10, 133:24, 139:4, 158:4 exponentiation 114:10	express 52:17, 98:1, 112:9, 115:13, 122:18 expressed 13:2, 98:17 expression 94:16 expressly 52:13, 99:18 extend 134:17 extension 132:16 external 46:13, 47:11, 51:4, 53:16, 57:5, 148:19, 149:2, 151:7 extracting 162:16, 162:24 extraction 114:22 extreme 14:11 extremely 101:14 F fabric 119:12 face 101:8 facilitate 168:8 facilitates 70:14 facilitating 66:2 facilitation 94:19 fact 13:11, 18:12, 41:19, 53:16, 55:3, 55:23, 76:18, 86:17, 95:6, 110:17, 111:10, 112:9,	113:3, 118:7, 126:18, 129:23, 129:25, 130:1, 131:11, 131:19, 132:3, 133:3, 137:18, 140:21, 142:13, 151:10, 152:7, 154:17, 157:11 factors 13:4 facts 48:21 fail 82:12 failed 137:2 fair 9:19, 9:20, 18:21, 67:22, 150:6 falling 99:20 false 137:24 fancy 149:24 far 8:18, 101:8, 105:2, 138:6 fashion 61:10, 86:15 fast 55:21, 56:2, 141:9 faster 149:5, 156:13, 156:14, 157:10 father 28:24 feasibility 22:10 feature 86:17 feed 127:25 feeding 117:8, 117:12
--	--	--	---

feeds 164:15 feel 7:22, 21:7, 46:4 ferris 3:14 fetch 113:22, 115:7, 116:3, 116:4 fetches 104:16 few 62:12, 168:17 figure 41:10, 41:24, 43:1, 43:2, 44:11, 44:12, 44:17, 44:18, 44:24, 45:8, 45:20, 46:5, 46:10, 47:1, 47:16, 47:20, 47:24, 48:1, 48:19, 48:22, 50:14, 51:7, 51:11, 51:14, 52:1, 52:25, 53:5, 53:24, 54:3, 54:11, 54:18, 54:19, 55:11, 55:13, 56:20, 56:22, 56:25, 57:14, 67:15, 67:16, 67:20, 80:18, 80:22, 81:2, 81:6, 81:7, 81:17, 83:8, 86:24, 87:3, 87:4, 87:11, 91:24, 91:25, 94:4, 94:10, 94:14, 94:25, 97:8, 99:14, 109:11, 110:10, 116:20, 117:7, 117:24, 118:1,	118:2, 118:4, 118:5, 118:12, 124:3, 124:22, 125:10, 131:21, 134:1, 138:9, 140:8, 140:9, 140:11, 140:13, 140:16, 141:17, 141:19, 141:24, 142:14, 142:16, 144:12, 144:13, 145:8, 160:9, 164:11, 164:12, 164:20, 164:21 figures 42:25, 58:19, 124:21 file 159:8 filings 35:12 final 97:11 finally 93:14, 94:12, 95:11, 116:11 financially 173:1 find 42:14, 42:19, 74:25, 103:16, 117:15 fine 33:15, 88:19, 101:19, 101:20, 108:20, 137:10, 141:4, 141:17, 143:1, 145:22, 154:4, 156:11 finish 62:3, 97:24, 103:14, 141:23, 157:9, 158:9 finite 62:5 first 32:15, 37:12, 44:15, 65:16,	67:7, 81:12, 81:20, 95:1, 107:21, 108:2, 115:2, 115:5, 115:19, 115:21, 120:2, 123:12, 123:14, 135:20, 136:23, 138:15, 138:18, 141:19, 142:11, 142:17, 144:20, 161:2, 161:13 fits 148:22 five 29:1, 62:25, 89:2, 168:20, 168:21, 168:22 fix 121:1, 134:21 fixed 27:13, 91:13 flag 104:4, 137:8 flipping 124:20 floor 3:7 flow 63:25, 85:12, 103:21, 105:22, 106:23, 113:5, 116:6, 126:2, 134:3, 137:13, 146:12, 165:9 flowing 43:10, 94:17, 94:18, 97:5, 99:19 flows 27:19, 97:4 fly 26:20, 49:6, 153:18, 154:2 focus 127:6, 154:5 follow 136:9, 168:13	follows 6:12 foregoing 171:8 form 8:17, 9:5, 12:16, 12:18, 14:3, 14:20, 15:1, 18:6, 19:2, 20:4, 23:15, 24:4, 24:21, 28:13, 32:22, 33:14, 35:7, 35:25, 36:15, 37:7, 39:4, 39:7, 40:1, 44:23, 45:12, 49:25, 50:25, 57:16, 60:13, 75:16, 75:25, 77:14, 78:1, 86:6, 93:25, 94:2, 95:21, 96:3, 100:19, 117:6, 122:11, 125:25, 128:11, 143:22, 144:8, 145:9, 145:15, 153:15 formats 76:24 formed 150:7, 150:19 forth 11:4, 11:6, 31:4, 38:22, 53:7, 56:4, 76:12, 124:20, 142:7, 145:20, 172:7 found 49:7, 113:6 foundation 11:2 four 28:1, 103:16 frame 76:14, 76:15
--	---	--	---

free 7:22, 21:7, 46:4	172:12, 172:14, 172:18, 172:22	111:11, 112:16, 153:3	glad 16:19, 37:3, 38:1, 58:1
frequently 25:13	G	generation 96:9, 96:23	glass 37:10
friday 1:21, 2:23, 5:2, 170:10, 172:16	gateway 34:1, 82:5	generations 29:1, 29:12	global 43:3, 44:13, 46:15, 47:2, 47:6, 47:11, 47:14, 47:20, 48:16, 48:21, 49:2, 49:11, 49:17, 49:22, 50:15, 51:5, 53:17, 53:20, 54:6, 54:21, 54:25, 55:2, 55:4, 55:13, 56:21, 57:5, 102:10, 104:19, 105:6, 107:3, 107:4
front 8:2, 20:6, 20:12, 21:3, 60:18, 67:18, 67:25	gave 19:6, 19:21, 36:23, 38:9, 107:18	generic 54:13	globally 49:19
front-end 33:4, 34:2, 63:13, 64:5, 64:8, 66:24, 68:14, 71:23, 73:18, 81:21, 82:7, 82:9, 82:21, 84:4, 84:12	general 12:3, 48:20, 50:2, 83:6, 102:22, 112:3, 114:2, 114:12, 115:17, 115:22, 126:21, 128:14, 144:24, 146:6, 146:10, 147:2, 147:5, 147:14, 147:22, 147:24, 148:5	generically 53:11	glovsky 3:14
full 45:14, 45:21, 48:13, 52:12, 57:21, 96:2	general-compute 113:14, 113:18, 113:25, 115:1, 120:3, 120:4	geometry 36:9, 36:25, 89:20, 92:3, 92:15, 93:13, 95:8, 95:24, 96:1, 96:5, 96:9, 96:13, 97:18, 98:12, 98:14, 98:15, 98:21, 99:8, 99:11, 104:22, 110:3, 120:14, 122:7, 122:13	go 7:8, 8:2, 13:4, 16:15, 16:18, 16:21, 17:20, 24:7, 28:22, 29:5, 29:14, 31:13, 39:11, 41:10, 44:16, 46:15, 51:18, 52:11, 52:15, 58:21, 65:5, 65:6, 68:12, 79:5, 80:11, 80:17, 89:4, 94:10, 98:6, 107:10, 107:23, 110:10, 116:4, 116:7, 116:24, 117:17, 121:18, 124:16, 126:6, 136:12, 138:7, 145:12, 145:16,
function 33:21, 33:24, 65:21, 70:14, 73:5, 91:13, 106:21, 106:22	general-purpose 36:6, 36:22, 38:2, 61:15, 61:17, 115:25, 116:17, 119:17, 119:18, 126:4	getting 51:19, 125:11	
functionality 91:14, 92:17, 112:14, 116:1, 116:14, 148:4	generally 17:3, 24:14, 24:17, 27:12, 28:22, 35:12, 59:13, 62:13, 83:6, 103:22, 108:11, 118:21	give 6:23, 17:5, 17:20, 20:7, 21:4, 29:16, 48:2, 55:10, 62:19, 101:6, 103:15, 139:9, 147:2, 152:1, 152:15, 152:19, 155:11, 156:10, 156:11, 156:12, 168:20	
functions 68:22, 69:24, 72:11, 104:19, 105:5, 114:11, 115:12, 119:5, 146:14	generate 11:24, 14:14, 95:10, 113:7	given 6:22, 44:19, 44:20, 45:8, 46:6, 56:9, 91:4, 92:19, 98:20, 100:16, 103:25, 112:11, 152:4, 172:10	
further 37:25, 96:7, 109:25, 134:18, 144:18, 145:1, 145:25, 159:3, 164:21, 169:3,	generated 30:1, 100:3, 110:2, 110:6, 110:12, 110:17, 111:2, 113:10	gives 70:13, 108:9	
	generates 11:23, 93:14,	giving 142:12	

146:1, 148:20, 148:22, 151:17, 157:23, 161:1, 164:11, 165:9 god 117:19 goes 93:12, 96:24, 109:10, 110:21, 156:5 going 29:1, 31:20, 31:23, 38:5, 39:21, 51:24, 51:25, 59:6, 63:3, 66:6, 69:8, 81:6, 88:15, 88:16, 88:17, 88:20, 89:2, 89:6, 101:21, 121:1, 122:4, 123:1, 141:9, 145:11, 148:18, 156:22, 157:7, 158:6, 158:8, 163:15, 166:15, 168:16, 168:23, 169:1, 169:9 gone 27:2 good 5:17, 6:16, 6:17, 74:15, 78:21, 85:5, 118:14, 136:2, 163:10 gotten 10:1 gpgpus 61:16 gpu 23:22, 25:16, 26:4, 61:12, 61:21, 62:18, 79:2, 92:23, 117:3, 117:4, 117:11, 117:21,	117:25, 118:7, 118:10, 118:20 gpus 61:6, 61:14, 61:17 grandfather 28:25, 29:2 granted 156:25, 157:6 granting 157:10 granularities 27:14 graphic 38:24, 89:24 graphical 97:25, 100:24, 107:7, 110:20, 112:18 graphics 30:18, 36:7, 36:8, 36:23, 36:24, 38:16, 39:1, 60:11, 61:12, 75:9, 76:4, 76:21, 76:22, 79:11, 79:20, 79:23, 79:25, 80:2, 80:16, 85:14, 85:18, 85:22, 93:11, 94:6, 94:7, 94:9, 94:21, 95:7, 95:12, 97:3, 97:7, 97:12, 99:18, 99:21, 102:12, 102:25, 103:1, 106:5, 106:6, 106:24, 107:7, 109:4, 109:16, 111:5, 111:6, 111:14, 112:3, 112:5, 113:19, 115:21, 117:3, 118:17, 118:21, 119:15, 120:2, 120:6,	126:3, 148:9, 148:17 great 29:2 ground 32:5 group 32:20, 35:1, 48:4 groups 43:9, 43:18, 129:15, 130:20 guaranteed 84:15 guess 35:12, 123:22, 128:12 guys 88:18 H hand 6:8, 14:7, 39:13, 68:12, 69:6, 70:9, 130:13, 173:4 handle 75:4, 128:2 handles 74:20, 74:23, 77:9, 78:25, 110:7 haphazard 84:14 happen 84:16, 151:5 happened 146:23 happening 25:8, 56:19 happens 27:19, 27:22, 61:5, 76:15, 106:5, 139:10 happens; 85:3 happy 150:20	hard 25:3, 28:2, 28:21, 45:19, 150:8, 152:24 hardware 11:25, 22:7, 22:11, 22:18, 22:23, 23:7, 23:9, 23:18, 23:21, 24:11, 25:8, 25:9, 25:23, 25:24, 26:1, 26:2, 26:3, 26:24, 27:20, 29:8, 61:7, 83:18, 147:23 hazards 82:11, 83:23 head 44:6, 152:22, 154:16 headache 7:1 heading 48:20, 127:20 hear 27:10, 74:4, 74:6, 156:3, 156:9 hearing 8:13 heavily 118:22 help 39:17, 99:2 helpful 152:19 helping 100:22 hence 83:10 here 5:4, 7:3, 8:5, 9:13, 9:15, 13:9, 17:16, 18:4, 20:11, 29:19, 35:3,
---	---	--	--

38:11, 46:11, 49:1, 53:10, 54:2, 60:1, 62:3, 64:6, 65:4, 65:11, 68:11, 71:15, 81:7, 91:4, 91:24, 100:21, 101:20, 106:17, 111:15, 111:21, 114:25, 115:5, 115:13, 119:21, 121:7, 121:18, 122:7, 128:1, 138:8, 138:20, 140:9, 140:14, 146:2, 154:17, 155:19, 156:7, 159:11, 162:2 here; 67:6 hereby 171:6, 172:4 hereinbefore 172:7 hereunto 173:3 herrington 3:4, 5:21 hesitate 150:11 hidden 30:11 high 11:22, 22:8, 27:14, 60:15, 60:18, 152:8, 155:15, 155:18, 156:2, 156:10, 156:11, 156:25, 157:1, 157:3, 157:6 high-level 22:13, 22:15, 23:3, 24:8, 25:21, 26:8, 27:6, 27:15, 27:22, 28:3,	76:3, 112:16, 113:4 high-performance 85:13, 86:12 high-priority 152:1, 152:9, 152:10, 155:11, 155:12, 155:20, 155:22, 156:5, 156:9, 156:16, 156:17, 156:18, 157:10 higher 28:22, 60:2, 60:4, 62:7, 97:2, 149:5 higher-level 17:4, 17:17, 28:19, 110:19, 111:3, 111:13, 113:11 highest 112:13 highlight 29:18 history 12:19, 42:13 hold 82:13, 83:24, 84:22, 85:4, 141:7 holds 84:7, 84:20 homogeneous 35:21 homogeneously 35:16 honest 156:21 honestly 52:5 honor 157:13 hope 38:24 hopefully 158:9 host 33:17, 33:18,	33:23, 33:25, 47:9, 50:19, 64:1, 67:17, 67:24, 81:21, 81:23, 82:1, 82:4, 82:12, 82:17, 83:4, 83:7, 83:9, 83:15, 84:10 hours 57:20 house 10:25, 11:1 houston 3:8 hundreds 19:4, 149:19 I id 48:2 idea 60:15 identical 131:25 identification 63:18 identified 32:23, 32:24, 33:1, 33:4, 33:8, 33:12, 71:9, 81:8, 92:3, 104:4, 162:3, 164:13 identifier 104:11 identify 5:15, 28:23, 53:8, 56:24, 71:21, 162:1 identifying 28:18, 28:19, 100:21 idle 136:1 ignorant 155:24 ignore 56:20, 56:22	ignoring 136:6 ill 6:19, 150:7 illness 6:22 illustrates 94:5 image 75:18, 76:2, 76:7, 76:13, 153:6 imagine 25:14, 51:16, 155:23 immediately 28:25 impact 30:22 implementation 83:19, 152:14, 153:2 implemented 83:18 implements 22:24 implied 40:13, 40:25, 52:14, 52:17, 133:25 implies 51:2, 116:16 imply 53:18, 65:19, 144:22 implying 13:5 important 13:24, 62:6, 140:21 imposed 40:2 impossible 100:5, 100:16 in-memory 149:21 include 8:9, 16:1,
--	---	---	---

47:7, 47:21, 55:7, 77:16, 104:9, 107:6, 150:4, 150:18, 162:16, 162:23, 164:21 included 54:20, 54:24, 61:18, 75:8, 88:11, 119:11 includes 30:19, 77:6, 93:8, 106:9, 108:1, 115:22, 138:9, 166:23 including 36:25, 46:13, 102:13, 103:23, 126:13, 158:15 incoming 70:15, 156:16 incompatible 86:13 inconsistency 111:15, 113:6 incorrect 18:13 increased 136:7 independent 86:19, 131:5, 132:3, 132:5 independently 81:19 index 158:24 indicate 52:20 indicates 48:14 indication 109:21, 132:4 indicative 48:10 indirect 51:23, 52:6 individual 110:6, 166:24,	167:16, 167:25 infer 40:25 inferred 40:14 infinite 62:10 inflow 102:23, 103:2, 107:22 information 12:22, 23:16, 24:5, 24:10, 25:1, 25:5, 26:10, 26:18, 26:24, 39:19, 43:6, 74:24, 80:5, 80:15, 81:22, 82:6, 90:18, 90:23, 91:2, 96:10, 96:21, 104:6, 104:9, 106:9, 112:19, 117:13, 118:16, 144:10, 158:24, 160:23, 161:16, 161:19 inherent 67:8, 67:13 initial 4:32, 12:24, 95:4, 166:11 injected 121:24 input 39:7, 39:10, 93:12, 104:10, 104:23, 106:10, 111:13, 147:7, 147:16 inputs 96:1, 96:14, 107:12, 108:6 inserted 84:11, 121:17 inside 81:24, 82:13, 82:18, 84:4,	84:5, 84:6, 84:20, 117:21, 118:10 inspection 172:18 instance 53:9, 104:23 instead 60:24 instituted 95:6 instruction 22:14, 91:6, 97:14, 98:24, 99:1, 103:24, 104:10, 104:16, 105:24, 106:3, 106:10, 106:12, 109:13, 109:21, 112:22, 112:24, 115:6, 116:2, 116:8, 116:9, 130:16, 130:17, 131:15, 131:17, 132:11, 132:13, 132:18, 133:1, 135:7, 138:12, 139:11, 140:17, 158:23, 158:24, 159:3 instruction's 92:21 instructions 9:23, 9:24, 10:5, 65:22, 65:25, 66:2, 66:23, 91:15, 93:23, 95:19, 105:17, 106:13, 108:7, 108:9, 108:10, 108:17, 108:22, 108:23, 108:24, 108:25, 109:3, 129:17, 131:7, 134:4, 134:5, 138:25, 140:3, 159:1 intellectual 4:26	intend 33:2, 133:19 intended 131:11, 133:20, 150:9 intensive 61:7 intention 50:4 inter 4:11, 7:4 interacting 147:19 interactions 57:22 interested 30:5, 173:1 interesting 149:24 interface 33:6, 33:17, 33:18, 33:23, 34:1, 41:22, 43:21, 45:1, 45:19, 47:5, 47:9, 50:18, 50:19, 63:13, 64:2, 64:16, 65:7, 65:12, 65:18, 65:21, 66:1, 66:7, 66:14, 66:25, 67:1, 67:12, 67:17, 67:20, 67:24, 68:1, 68:15, 68:19, 71:24, 73:15, 81:22, 81:23, 82:1, 82:5, 82:17, 82:22, 83:4, 83:7, 83:9, 83:15, 83:19, 84:11, 86:25, 87:7, 87:22, 87:24, 148:22, 148:23, 151:8 interfaces 82:12, 87:21
--	--	---	--

intermediary 39:14 intermediate 56:3, 109:15, 111:11 intermediate-level 27:16, 113:4 international 4:28 interpretation 16:9, 68:4, 82:19, 111:16, 111:17 interpreting 19:25 interprocess 43:20 introduce 7:8, 15:7, 41:2, 80:10, 88:23, 123:7, 166:6 inv 4:36 invalid 73:10 invalidate 70:20 invalidating 71:20 invalidity 4:16, 15:14, 21:9, 21:16, 42:13, 167:8 invention 12:21, 70:21, 71:20, 118:8, 128:25, 151:15 inventions 129:1 involve 38:19, 150:13 involved 31:1, 160:4, 160:24, 164:4 involvement 24:13	involves 113:21, 114:7 involving 155:3 io 148:11, 148:16, 148:18, 148:20, 149:5, 149:6, 149:15, 149:23, 150:4, 150:13, 150:15, 150:18, 151:1 ipr 1:14, 2:14, 5:8, 7:5, 7:19, 16:5, 32:5, 35:12, 163:23, 166:16, 170:6 issuance 138:24 issue 13:9, 19:13, 45:25, 167:6 issues 108:13, 153:20 itc 4:31, 15:15, 18:17, 19:21, 19:24, 21:8, 21:15, 162:15, 166:11, 167:14, 167:25 item 8:4, 60:23, 60:25, 72:8, 72:9, 72:10, 108:11, 108:14, 117:25, 151:8, 162:2 items 67:24, 68:6, 70:9, 71:8, 72:4, 89:19, 108:7, 110:12, 110:13, 125:4, 129:15, 130:21, 130:22, 131:8, 132:9, 133:6,	135:22, 162:3, 162:4 itself 11:13, 17:19, 18:16, 22:19, 25:8, 25:9, 25:23, 30:7, 30:23, 35:19, 41:1, 44:14, 47:1, 61:4, 65:19, 66:4, 75:19, 76:18, 80:13, 84:4, 94:24, 96:18, 102:15, 112:7, 112:12 J january 1:21, 2:23, 5:2, 5:9, 170:10, 172:16, 173:4 jeffrey 3:5, 5:20, 102:2 jerry 3:25, 5:12, 101:13 jiao 4:25, 89:14, 89:18, 90:5, 90:18, 91:20, 92:5, 92:13, 93:16, 94:4, 94:13, 94:15, 95:15, 101:2, 107:2, 108:22, 109:2, 112:10, 113:13, 113:20, 114:24, 115:6, 115:11, 117:5, 117:9, 118:19, 119:6, 119:12, 120:9, 120:19, 122:1, 122:4, 122:8, 122:14, 122:18, 122:19	jiao's 122:12, 122:19 job 1:26, 25:22, 67:5, 68:24, 133:17 johnson 3:5, 5:20, 5:23, 6:1, 8:17, 9:5, 12:16, 14:3, 14:20, 15:1, 18:6, 18:23, 19:2, 20:4, 23:15, 24:4, 24:21, 28:13, 32:22, 33:14, 35:7, 35:25, 36:3, 36:15, 37:7, 39:4, 40:1, 44:23, 45:12, 49:15, 49:25, 50:25, 57:16, 60:13, 62:2, 64:13, 74:22, 75:25, 77:14, 77:21, 78:1, 86:6, 89:1, 93:25, 94:2, 95:21, 100:8, 100:19, 101:20, 102:4, 117:6, 118:13, 122:11, 125:25, 128:11, 141:7, 143:22, 144:8, 145:9, 153:15, 168:16, 169:3 judge 166:23 jump 15:18, 59:6, 74:17, 123:12, 124:21, 129:7, 158:10 K karen 37:13
---	---	---	---

keep 25:10, 29:1, 37:15, 56:3, 60:25, 66:6, 88:15, 88:16, 88:17, 88:20, 89:1, 136:13, 137:1, 158:6, 158:8 keeping 25:9 keeps 156:1 kept 25:14 key 13:9, 128:25, 149:7, 161:18 kieran 3:26 kind 5:24, 10:16, 25:7, 25:24, 30:8, 39:15, 39:16, 42:14, 50:5, 67:9, 75:6, 82:17, 83:4, 121:20, 146:13, 157:16 kinds 38:23, 82:10, 83:22 know 24:16, 25:3, 25:7, 26:1, 29:4, 29:11, 29:19, 29:24, 31:10, 37:2, 42:24, 50:3, 65:21, 66:21, 66:23, 67:1, 70:7, 72:14, 72:18, 72:20, 73:12, 78:9, 81:10, 81:23, 82:1, 84:18, 85:17, 85:24, 87:4, 88:10, 88:13, 91:12, 93:7, 97:7, 97:20, 108:21, 109:23, 114:15, 116:3, 117:17, 127:1, 137:3, 137:6, 142:23, 142:24, 143:19, 153:8, 157:22, 168:16 knowing 24:1, 24:17 knowledge 12:20, 43:16, 69:5, 69:16 known 43:23, 61:16, 85:16, 90:8 knows 115:17, 117:19, 152:17 L labeled 41:11 lack 48:8 language 22:14, 22:15, 23:3, 23:19, 40:24, 76:3, 76:4, 119:20, 119:22, 162:6 large 14:8, 62:11 last 31:11, 66:13, 105:4, 126:14, 138:8, 155:6 later 91:9, 108:6, 121:1, 144:21, 144:25 laws 171:7 laying 11:2 leaning 101:8, 101:9 learning 61:17 least 43:23, 56:25, 58:19, 66:13, 84:18, 85:20, 94:8, 139:16, 139:17, 165:14, 168:8 left 114:16, 116:7, 146:24 let's 7:8, 15:7, 31:5, 31:9, 31:17, 33:25, 37:14, 37:15, 38:15, 44:17, 51:7, 58:21, 62:24, 65:11, 66:6, 70:7, 71:5, 74:17, 80:10, 88:23, 98:6, 101:1, 101:14, 101:17, 103:13, 105:1, 105:4, 107:23, 107:24, 113:12, 114:25, 119:21, 122:22, 123:11, 124:21, 127:6, 129:7, 130:13, 141:19, 142:13, 152:18, 152:19, 154:5, 158:5, 158:10, 161:2, 163:12, 164:11 level 11:23, 11:24, 22:8, 25:13, 27:21, 28:17, 95:18, 99:10, 103:23, 103:24, 105:24, 105:25, 106:3, 106:4, 106:19, 107:4, 107:6, 109:14, 109:15, 111:11, 112:13, 152:5 levels 27:3, 28:1, 28:9, 28:12, 29:5, 103:22, 105:23, 106:19, 107:5, 107:6, 110:21, 111:9, 151:24, 155:14, 155:24 levin 3:14, 3:27, 5:18 light 30:15, 75:13, 76:6 lighting 30:22, 31:3, 38:18, 75:14, 77:8, 90:10, 90:21, 91:5, 91:9 like"; 75:21 likely 81:24, 81:25, 151:17 limit 109:9, 112:6, 154:11 limitation 35:14, 35:19, 40:2, 40:13, 40:23, 145:1 limitations 145:1, 146:3 limited 144:20 limiting 162:19, 162:20, 162:22, 162:23 line 46:24, 46:25, 51:3, 64:6, 64:24, 65:8, 65:17, 78:22, 79:21, 80:3, 80:11, 90:5, 			
--	--	--	--

93:19, 94:22, 95:25, 96:4, 102:17, 102:21, 103:6, 103:21, 104:18, 110:1, 115:14, 124:10, 124:11, 141:14, 160:8, 161:13 linear 109:6 lines 64:18, 105:20, 118:25, 119:5, 157:25, 158:1 list 96:4 litigation 8:18 little 6:18, 6:25, 46:19, 63:24, 88:21, 101:6, 105:3, 117:22, 134:12, 141:8, 148:7, 155:17, 156:22, 164:16 lived 29:3 llp 3:4 ln 170:15 load 68:23, 72:23, 73:18, 73:21, 113:23, 116:11, 116:12, 120:14, 122:1, 159:4, 159:5, 160:20, 161:25, 162:5, 162:13, 162:16, 164:9, 166:25, 168:1, 168:6 loaded 39:17, 163:25 loading 70:16, 72:12, 73:14, 158:20,	160:24, 161:7, 161:9, 163:1, 164:5 loads 73:15, 159:16, 159:19, 159:23, 160:1, 160:11, 160:14, 160:17, 161:4, 162:2, 162:7, 162:10, 163:4, 163:5, 163:6, 163:8 local 7:23, 45:16, 45:17, 46:7, 48:2, 48:15, 48:24, 48:25, 49:9, 50:16, 50:20, 53:13, 53:20, 54:5, 54:23, 54:25, 55:2, 55:17, 55:18, 55:19, 56:6, 56:10, 56:11, 56:14, 57:3, 57:11, 58:5, 58:12, 58:14, 151:9 locally 79:2, 79:3 located 116:20 location 30:15, 39:11, 119:7 locations 121:15 logic 115:16, 115:23, 115:25, 116:9, 116:15 logical 14:5, 94:6, 94:7, 94:21, 97:7, 97:25, 99:18, 99:20, 102:12, 106:5, 106:24, 107:6,	109:4, 110:19, 111:4, 112:18, 134:7 long 74:10, 83:25, 98:4, 104:25, 109:23, 136:25, 162:21 look 21:12, 23:25, 25:18, 30:5, 33:16, 35:17, 38:15, 42:19, 43:2, 44:15, 45:14, 45:21, 46:4, 50:8, 51:7, 52:15, 55:9, 57:17, 59:21, 69:7, 69:19, 69:21, 75:12, 76:1, 78:21, 82:9, 101:4, 102:21, 107:23, 107:24, 117:2, 117:7, 118:4, 119:21, 129:21, 134:2, 141:19, 157:24, 168:18 looked 45:22, 46:2, 46:18, 93:18, 95:14 looking 9:8, 12:18, 16:22, 44:16, 46:8, 67:15, 71:11, 71:16, 72:3, 76:8, 80:17, 83:20, 84:3, 95:2, 96:13, 97:6, 99:25, 108:8, 108:19, 116:19, 129:22, 130:5, 133:25, 163:22 looks 9:25, 102:3,	102:7 los 1:20, 2:21, 5:1, 171:3 lot 23:16, 24:5, 24:10, 26:18, 26:22, 56:9, 67:2, 75:10, 91:10, 130:12, 149:18 lots 26:4, 30:20, 96:16 low 27:14, 60:16 low-level 23:19, 24:7, 27:7, 27:17, 27:23, 27:24, 28:4, 28:11, 86:11, 86:18, 110:17, 111:1, 113:5 low-priority 62:9, 156:6 lower 11:23, 11:24, 60:5 lower-level 10:20, 13:18, 28:18, 111:10, 111:12, 113:7 lower-priority 157:15 lowest 106:19, 109:14 lunch 88:14 M machine 22:16, 23:4, 24:12, 61:17, 147:6, 147:18 machine-specific 23:10 machines 147:19, 147:20,
--	---	--	--

148:5 made 21:1, 24:24, 24:25, 42:10, 144:21 magic 111:21 magical 111:23 main 3:6, 60:14, 60:19, 91:23, 118:7, 118:8 maintain 139:1 maintains 139:23 make 26:25, 50:1, 74:2, 82:9, 83:22, 83:25, 94:12, 97:16, 106:25, 109:7, 112:17, 120:25, 121:2, 136:20, 141:19, 144:2, 144:17, 145:15 makes 67:12, 68:20, 111:18, 152:25 making 6:19, 67:4, 143:2, 143:3, 153:21 manage 137:14 manageable 7:1 manager 124:5, 124:17, 125:3, 125:15, 126:7, 127:8, 127:20, 127:22, 128:8, 131:21, 138:9, 138:25, 139:23, 142:5, 142:18, 144:14, 158:15, 159:12,	160:20, 160:22, 160:24, 161:4, 161:16, 161:19, 162:10, 162:24, 162:25, 163:6, 163:8, 163:24, 163:25, 164:4, 164:14, 165:14 maneuver 146:24 manner 91:3 mantor 4:22 many 13:4, 19:5, 19:6, 26:14, 31:3, 54:9, 54:11, 54:17, 55:10, 70:24, 75:15, 92:18, 94:12, 98:23, 149:20, 150:16 many-to-one 11:16, 11:18 map 22:23, 24:2, 24:19, 77:4 mapped 22:18, 22:23 mapping 12:2, 25:6, 28:11, 104:11 marked 7:13, 7:16, 15:11, 41:4, 80:8, 88:25, 123:8, 166:8 marriage 172:24 massoud 1:18, 2:18, 4:4, 4:9, 4:14, 5:5, 6:10, 169:9, 170:12, 171:6, 171:17, 172:6, 172:15 mathematical 114:14	matrices 74:20, 75:16, 77:6, 79:1 matrix 151:14, 151:15 matter 5:6, 7:4, 57:19, 92:25, 99:23, 166:11, 173:2 maximize 61:3, 118:18 maybe 10:2, 18:15, 19:12, 20:16, 20:17, 23:6, 29:3, 29:4, 35:1, 56:23, 69:1, 75:11, 76:13, 78:23, 79:22, 88:21, 114:10, 116:12, 120:25, 121:3, 135:21, 147:25, 148:21, 152:16, 153:20, 158:3, 158:4, 158:9, 161:6, 161:8, 163:9 mean 10:14, 11:9, 16:14, 17:15, 19:4, 27:15, 28:15, 28:20, 30:1, 30:18, 33:2, 33:3, 34:17, 36:7, 36:17, 37:5, 37:9, 37:11, 37:12, 37:22, 38:2, 38:6, 38:12, 38:14, 41:17, 42:16, 45:13, 48:4, 51:16, 54:22, 55:16, 55:19, 55:22, 55:23, 56:5, 59:15,	61:14, 72:14, 73:3, 73:11, 73:12, 76:22, 77:15, 77:16, 77:22, 78:9, 78:12, 78:19, 79:10, 79:12, 85:8, 85:13, 85:17, 88:4, 98:18, 100:9, 107:3, 111:12, 114:8, 114:12, 114:13, 115:13, 117:1, 125:6, 127:1, 127:15, 137:11, 140:5, 140:9, 143:16, 145:13, 148:15, 149:17, 150:10, 153:16, 155:17, 155:19, 155:21, 159:24, 161:6, 164:1 meaning 17:5, 27:10, 34:20, 34:23, 55:21, 59:21, 112:23, 112:25, 114:6, 143:17, 161:7, 162:19, 162:22 means 19:25, 31:15, 38:9, 43:19, 47:25, 49:18, 49:21, 49:22, 56:1, 59:13, 61:7, 75:4, 85:10, 94:7, 100:11, 112:12, 112:15, 121:25, 130:16, 131:6, 132:7, 139:22, 144:17, 153:8, 157:7, 161:7, 161:10 meant 16:23, 18:7,
--	--	---	---

18:8, 86:7, 88:6, 137:19, 145:18, 145:19, 150:4, 152:23 mechanism 56:16, 86:10, 86:11, 86:19, 149:6 mechanisms 40:20, 104:5 media 5:4 memories 47:2, 48:7, 48:11, 48:24, 48:25, 50:16, 51:12, 52:24, 55:7, 56:6 memory" 51:5 mention 18:18, 69:8, 71:13, 107:11, 107:19, 127:10 mentioned 20:15, 27:5, 62:1, 62:20, 62:22, 73:14, 81:18, 129:8, 142:3, 154:8 microarchitecture 23:17, 24:11, 26:25, 35:22 microcode 158:25, 159:1 microinstruction 22:17 microinstructions 22:17 middle 127:9, 128:13, 140:25, 154:8 might 10:20, 11:13, 19:14, 19:15, 22:7, 27:2, 44:7, 45:18, 53:23, 71:9,	76:10, 152:7, 167:25 mind 89:1 mine 53:23 minor 148:7 mintz 3:14, 3:27, 5:18 minute 89:2 minutes 57:20, 62:25, 88:21, 168:17, 168:20, 168:21 mirror 136:9 mischaracterizing 139:20 missing 37:2, 83:20 mix 149:23 modify 46:19, 50:9 module 68:23, 100:7, 100:16, 100:18, 100:20, 112:2, 168:5 modules 69:11, 92:2, 112:20 moment 7:11, 15:10, 124:12 monitor 5:10, 31:25, 89:7, 89:10, 101:10, 101:11, 101:25 monitoring 26:6 more 10:11, 10:14, 10:21, 14:23,	17:18, 18:3, 18:5, 23:16, 24:5, 24:23, 26:22, 27:25, 28:22, 29:4, 29:5, 29:8, 31:13, 37:24, 45:7, 46:3, 48:10, 48:11, 50:9, 51:7, 51:21, 54:1, 54:10, 56:9, 57:17, 57:18, 57:19, 57:25, 61:5, 70:3, 83:5, 85:20, 92:9, 101:6, 105:2, 133:7, 133:20, 136:8, 151:25, 152:13, 155:10, 157:23, 162:20, 162:23, 164:3, 164:16 morning 5:17, 6:16, 6:17, 31:13 morrisson 3:26 most 62:17, 81:24, 81:25, 137:3, 151:16, 151:17 mountain 3:17 move 37:15, 58:21, 108:20, 145:23, 158:5 moved 167:20 much 31:9, 51:21, 139:3, 139:25, 145:24, 149:4, 149:5 muffled 5:24 multiple 10:19, 11:5,	11:6, 11:8, 11:10, 11:12, 11:19, 14:15, 17:6, 18:9, 18:14, 18:19, 20:1, 20:8, 20:10, 21:5, 21:17, 21:25, 22:17, 25:15, 27:3, 27:23, 27:24, 27:25, 44:19, 52:24, 59:16, 61:23, 61:24, 75:13, 96:3, 102:9, 102:20, 103:8, 103:25, 107:5, 111:8, 126:19, 128:2, 128:15, 128:18, 129:2, 129:9, 130:3, 130:9, 130:16, 130:18, 130:23, 131:1, 131:3, 132:7, 132:9, 132:12, 132:17, 134:9, 134:10, 134:16, 134:18, 139:5, 141:25 multiple" 18:5 multiplication 113:24, 114:20, 151:15 multiplied 75:17 multiply 114:16 multistranded 26:16 multitasking 127:3, 128:23, 130:1 multithreading 127:3, 128:22, 130:1 must 82:8, 84:22,
--	---	---	--

91:15, 91:18, 92:10, 92:25, 97:13, 97:14, 99:5, 99:23, 100:15, 152:16, 166:25, 167:16, 167:18 muted 102:3	28:23, 31:8, 31:15, 34:23, 35:5, 37:25, 39:24, 40:11, 42:14, 43:10, 46:2, 56:17, 57:18, 57:20, 61:8, 75:1, 75:11, 77:4, 89:2, 111:25, 114:5, 116:6, 131:13, 133:15, 137:11, 149:7, 151:18, 156:12, 168:7, 168:19 needed 24:10, 25:1, 26:25, 43:7, 44:2, 54:7, 70:3, 80:15, 86:11, 118:16, 118:17 needs 60:23, 75:19, 96:14, 96:15, 96:21, 96:22, 97:11, 101:15 negative 150:10 neighbor 40:5 neither 148:15 never 20:2, 49:4, 70:10, 111:18, 114:4, 136:16, 163:6 new 46:1, 50:10, 59:14, 60:17, 80:19, 82:14, 100:3, 139:21, 146:20 newer 95:10 next 37:15, 39:8,	76:14, 97:2, 99:10, 159:18 non-limiting 104:1 none 45:23, 84:21, 99:3, 137:15 noninfringement 167:11 nonsensical 111:17, 131:25, 150:16 normal 60:5, 60:17, 62:8, 153:1 normally 49:16, 49:21, 61:11, 79:5, 114:18, 153:5 nos 4:17 notary 172:3, 173:9 note 20:8 notes 102:19, 107:19 nothing 27:11, 28:4, 40:7, 54:10, 61:1, 77:15, 86:19, 108:12, 109:25, 136:1, 136:2, 145:14, 145:22, 145:24 notion 13:3, 13:13, 13:23, 61:16, 115:20 number 2:25, 4:29, 5:5, 5:8, 7:5, 13:17, 13:18, 14:8, 27:13, 55:10, 62:4, 62:5, 62:11, 80:22, 91:3, 94:22, 102:17,	104:9, 106:9, 108:4, 112:11, 114:17, 124:13, 134:21, 134:23, 146:14, 146:15, 170:6 numeral 48:9 numerals 53:3, 53:10, 54:12 nvidia 23:22, 85:15, 85:17, 85:22
N			O
name 29:2, 48:20, 70:12, 170:3 named 53:6 names 70:13 narrowed 144:25 natalie 1:24, 2:24, 6:4, 172:2 nature 119:10, 142:9, 142:19 near-memory 149:22 nearest 40:5 necessarily 11:15, 18:14, 40:25, 48:6, 48:23, 86:4, 140:18, 140:19, 148:1, 154:22 necessary 22:8, 42:19, 45:23 necessity 20:9 need 8:9, 15:25, 16:1, 21:6, 21:12, 23:16, 24:5, 25:5, 25:7, 26:1, 26:18, 26:21,			object 30:6, 75:2, 80:2 objection 8:17, 9:5, 12:16, 14:3, 14:20, 15:1, 18:6, 18:23, 19:1, 19:2, 20:4, 23:15, 24:4, 24:21, 28:13, 32:22, 33:14, 35:7, 35:25, 36:3, 36:15, 37:7, 39:4, 40:1, 44:23, 45:12, 49:15, 49:25, 50:25, 57:16, 60:13, 62:2, 64:13, 74:22, 75:25, 77:14, 77:21, 78:1, 86:6, 93:25, 94:2, 95:21, 100:8, 100:19, 117:6, 118:13, 122:11, 125:25, 128:11, 143:22, 144:8, 145:9, 153:15 objects 30:22

obvious 18:8, 42:7, 69:4, 69:15, 71:19, 119:9 obviously 24:6, 32:25, 43:5, 66:1, 82:7, 91:1, 96:12, 133:18, 133:19, 135:15 occurred 172:15 off-chip 47:6 office 1:1, 2:1, 5:8 offline 50:9, 56:24 often 61:5 oh 29:21 okay 6:25, 7:3, 7:8, 7:10, 7:18, 9:15, 11:7, 12:6, 14:24, 15:13, 15:18, 16:21, 21:20, 29:16, 29:21, 31:8, 31:16, 32:3, 33:10, 46:4, 46:9, 50:22, 57:13, 59:10, 64:17, 65:6, 65:15, 68:9, 74:13, 75:4, 77:12, 79:25, 80:17, 81:25, 89:12, 98:3, 98:10, 101:16, 101:18, 102:2, 103:19, 105:8, 105:19, 107:25, 116:19, 116:21, 117:5, 117:22, 118:11, 119:3, 120:12,	126:12, 130:25, 137:24, 138:22, 141:17, 143:1, 143:10, 143:23, 144:3, 144:4, 146:1, 149:13, 150:20, 153:10, 154:4, 154:11, 155:1, 158:5, 158:6, 158:20, 159:22, 160:19, 163:12, 163:21, 164:8, 164:11, 165:1, 166:10, 167:20, 168:12, 169:5, 169:7 on-chip 149:4 on; 77:9 once 90:6, 155:12 one-dimensional 40:16 one-to-many 11:15, 11:17, 12:1 one-to-one 11:14, 12:2 ones 62:16, 103:13, 107:10 only 13:6, 14:12, 14:14, 18:17, 21:1, 21:19, 23:8, 27:21, 28:2, 39:18, 42:6, 48:12, 49:12, 49:13, 51:14, 52:12, 53:2, 53:13, 55:14, 58:3, 58:10, 81:20, 87:25, 88:9, 95:3, 97:16, 99:11, 107:4, 112:7, 113:1,	118:1, 130:6, 131:12, 133:4, 133:5, 133:16, 133:22, 133:23, 134:7, 135:11, 135:21, 135:22, 135:24, 136:23, 137:21, 142:13, 144:19, 148:1, 154:12, 154:21, 157:4 ooo 169:12 open 8:1, 141:23 open-ended 36:16, 149:17 operable 129:13, 129:14, 130:18 operand 115:7, 116:3 operands 116:4 operate 132:13, 135:23, 149:4, 151:18 operated 106:18 operating 84:14, 107:7, 132:25, 136:13 operation 65:17, 65:24, 66:21, 86:12, 101:3, 109:12, 148:20, 150:18 operations 75:15, 90:9, 90:20, 91:4, 91:10, 113:23, 118:25, 148:11, 149:22, 150:4, 150:13 opine 26:20, 32:19, 49:3, 146:2, 153:9, 158:12,	159:19, 163:24, 167:12 opined 120:12, 159:22, 160:14, 167:12 opinion 8:16, 9:16, 12:6, 12:13, 12:17, 13:2, 13:4, 13:10, 14:1, 14:18, 28:10, 32:14, 32:15, 33:10, 42:3, 49:24, 64:3, 73:15, 93:1, 93:3, 95:3, 95:4, 100:5, 113:2, 113:12, 113:17, 155:9, 159:11, 160:15, 161:3, 163:2, 164:17, 167:7 opinions 12:14, 13:7, 42:10, 59:3, 59:7, 123:10 opportunity 138:4 option 135:1, 136:24 options 134:25 orange 142:19 orchestrate 134:8 orchestrating 93:10 orchestrator 106:7 order 90:19 ordering 62:14 ordinary 114:6, 162:19 organization 4:27, 135:12
---	---	--	--

organize 129:1, 131:14, 135:1, 135:3, 137:11 organized 129:24, 131:4 original 76:18, 145:2 orrick 3:4, 5:20, 6:1 others 88:18 otherwise 82:10, 84:24, 91:8, 97:13, 107:14, 154:23, 162:17, 173:1 out 59:14, 92:17, 96:24, 101:8, 110:5, 111:20, 138:19, 141:12, 156:5 outcome 173:1 outflow 102:23, 103:2, 107:23 output 39:11, 104:10, 104:11, 106:10, 109:10, 147:8 outputs 96:10 outputting 96:2 outset 99:5 outside 92:22, 117:11 over 19:23, 39:8, 58:9, 101:9, 103:15, 162:4 overall 85:12, 95:12 overwrite 84:1	own 36:12, 44:9, 45:9, 45:16, 46:12, 56:1, 68:24, 70:14, 83:7, 87:17, 101:17, 121:23, 131:5, 133:14, 137:13, 161:9 owner 1:11, 2:11, 2:19, 3:13, 5:19 P pacific 5:10 packets 104:23 page 4:3, 4:8, 15:18, 15:19, 15:22, 18:4, 29:16, 32:4, 59:22, 87:5, 94:4, 124:9, 124:11, 124:17, 125:2, 125:16, 126:7, 126:13, 126:16, 127:9, 128:13, 133:24, 138:7, 141:15, 144:7, 160:8, 160:9, 162:13, 166:22 pages 19:5 palettes 76:11 papers 149:19 paragraph 8:3, 32:9, 34:4, 51:3, 58:23, 59:1, 59:2, 59:6, 59:7, 74:18, 99:15, 115:6, 121:8, 126:15,	126:17, 127:18, 128:12, 128:13, 133:24, 138:8, 138:11, 138:23, 139:15, 140:6, 140:24, 140:25, 144:5, 144:9, 145:7, 145:19, 146:2, 151:22, 154:6, 154:9, 155:6, 155:7, 158:12, 159:14, 159:18, 159:25, 160:9, 161:14, 162:14, 165:11 paralegal 3:27 parallel 25:16, 32:15, 41:23, 42:15, 55:24, 55:25, 56:6, 56:14, 56:15, 68:1, 70:24, 135:4, 137:12, 165:2, 167:19, 167:22 parallelism 136:10 parameters 66:9, 77:11, 82:6 parent 25:21, 27:1, 28:18, 28:20, 28:23, 153:2, 153:4, 157:2 parentheses 45:1, 45:2 parenthesis 54:13 parenthetical 53:3 part 10:1, 12:11, 12:23, 13:14, 33:3, 33:20, 44:12, 46:15, 47:2, 47:14,	48:16, 49:2, 53:20, 54:6, 54:21, 54:24, 55:4, 55:13, 56:21, 59:18, 66:13, 66:18, 75:20, 76:16, 76:25, 80:15, 83:6, 86:22, 91:20, 94:6, 110:25, 111:4, 117:2, 117:3, 117:4, 118:3, 118:7, 118:20, 119:13, 150:5, 151:5, 151:14, 151:16, 158:11, 167:9 partes 4:11, 7:4 particular 8:19, 11:22, 23:6, 23:12, 23:18, 23:21, 25:19, 25:22, 40:12, 53:8, 53:21, 54:4, 56:8, 70:12, 76:19, 124:12, 130:13, 138:11, 138:16, 139:10, 139:21, 139:22, 140:15, 141:3, 141:14, 142:9, 142:12, 142:20, 142:25, 143:14, 144:6, 144:17, 161:7, 167:14 parties 5:13, 172:20, 172:23, 172:25 partitioned 137:20 partitioning 131:11 parts 49:11, 49:12, 49:22, 69:13
--	--	--	---

party 172:12 parvizi 6:4, 172:2 parvizi-azad 1:24, 2:24 pass 72:25 passed 64:5, 64:10, 64:15, 90:6, 90:18, 90:24, 91:18, 92:6, 92:15, 95:20, 104:6, 109:8 passes 39:2, 65:12, 92:11 passing 51:19, 72:13, 91:21 past 134:12 patent 1:1, 1:4, 1:11, 1:15, 2:1, 2:4, 2:11, 2:15, 2:19, 3:13, 4:12, 4:16, 4:19, 4:21, 4:24, 5:7, 5:18, 7:5, 8:19, 9:11, 9:17, 9:22, 12:4, 15:15, 32:12, 58:25, 59:4, 59:8, 59:22, 78:22, 79:13, 80:17, 80:21, 80:25, 89:20, 122:5, 123:12, 146:4, 166:25, 170:8 patents 8:18 path 51:23, 52:1, 52:6, 52:8, 97:19, 100:2,	115:15 pathway 67:11 pause 31:17, 74:10 pay 19:20 pcie 117:17 pct 4:26 pe 135:11, 158:23, 158:25, 159:3, 159:7, 159:8 peak 103:7 pedram 1:18, 2:18, 4:4, 4:10, 4:15, 5:6, 6:10, 6:16, 7:18, 15:13, 16:11, 19:19, 31:6, 32:4, 41:6, 52:9, 57:9, 63:1, 63:11, 88:13, 89:14, 101:14, 102:7, 104:24, 107:17, 108:8, 110:12, 120:22, 123:10, 141:7, 141:13, 163:21, 166:13, 169:9, 170:12, 171:6, 171:17, 172:6, 172:15 penalty 6:11, 171:7 pending 168:12 people 137:3 percent 117:21 perform 25:6, 38:17, 67:10, 68:21,	69:22, 69:24, 72:10, 77:8, 80:4, 80:16, 90:20, 90:25, 91:3, 91:11, 92:16, 96:8, 98:7, 98:10, 98:25, 112:11, 115:17, 119:5, 120:5, 126:3, 126:9, 126:14, 133:5, 146:14, 168:8 performance 26:6, 118:19, 168:8 performed 10:17, 22:20, 33:21, 72:11, 97:1, 98:20, 98:22, 98:23, 99:16, 111:7, 149:8 performing 90:8, 98:16, 98:18, 98:19, 99:10, 102:25, 115:12, 128:18 performs 68:21, 99:12, 106:21, 106:22, 118:25 perhaps 57:17 period 82:14 perjury 6:11, 171:7 person 12:20, 42:8, 43:11, 43:24, 53:19, 57:1, 69:4, 69:16, 69:18, 70:20, 71:4, 71:16, 72:7, 72:14, 72:20, 73:8, 73:11, 76:7,	82:3, 83:20, 84:2, 95:2, 96:12, 99:24, 107:8, 109:1, 113:9, 116:15, 119:9, 128:5, 129:5, 129:21 perspective 37:3, 46:12 pertaining 116:1 pes 137:9, 159:1 petition 4:11, 124:22 petitioner 1:8, 2:8, 3:3 pg 170:15 ph 1:18, 2:18, 4:4, 4:15, 6:10, 170:12, 171:6, 171:17, 172:6, 172:15 phenomena 56:14 phonetic 108:2, 137:24 phrase 75:7, 78:13, 138:16, 141:14, 143:13, 144:6 phrasing 155:16 physical 119:7, 147:10, 147:13, 147:14 picked 95:23 piece 22:7, 98:3 pipeline 85:13, 93:12, 94:6, 94:7, 94:8, 94:21, 97:3, 97:4, 97:5, 97:7,
---	--	---	---

97:25, 99:19, 99:21, 100:24, 102:12, 106:6, 106:24, 107:7, 109:5, 110:20, 111:5, 112:18, 130:5 pixel 17:25, 31:1, 36:9, 36:25, 38:18, 38:19, 66:9, 76:21, 89:21, 92:4, 92:16, 93:14, 93:24, 95:9, 97:18, 104:22, 110:3, 120:13, 122:8, 122:13 pixel-type 30:25 place 12:25, 19:7, 19:10, 20:15, 38:24, 39:20, 126:12, 162:6 placed 48:20, 91:25 plain 114:5, 162:18 plan 153:21 planet 5:12, 6:5 platform 23:9, 23:10 play 52:20 please 5:14, 6:7, 7:9, 8:2, 15:7, 17:1, 19:20, 22:4, 29:14, 31:8, 31:15, 32:2, 36:18, 37:2, 37:12, 37:16, 41:2, 47:18, 63:10, 65:5, 65:15, 78:11,	88:24, 89:12, 102:17, 103:18, 107:18, 116:24, 121:4, 122:23, 123:7, 150:24, 163:7, 169:7 pluralities 167:1 plurality 17:23, 32:11, 54:14, 59:24, 68:5, 69:10, 69:11, 69:12, 69:13, 78:14, 104:14, 120:18, 123:14, 123:16, 129:13, 158:17, 161:24, 165:13, 165:18, 168:3 plus 121:9 point 6:25, 20:18, 20:24, 23:10, 36:18, 37:2, 46:20, 46:24, 60:22, 62:3, 65:15, 66:20, 75:12, 91:25, 93:21, 95:17, 96:4, 106:17, 108:19, 114:24, 115:5, 121:1, 121:5, 123:21, 123:23, 125:3, 134:13, 138:19, 140:21, 141:19, 146:6, 153:9 point-to-point 40:21, 41:23 pointed 42:25, 52:22, 53:25, 57:14, 100:12, 105:1, 108:12, 108:20, 127:9 pointer 75:6, 121:15,	121:20, 121:21 pointers 74:25 pointing 100:17, 141:12 polygon 17:21, 17:22, 29:17, 29:23, 30:6, 30:7, 30:11, 30:12, 30:17, 30:20, 30:23, 79:11, 79:22 pool 91:22, 92:11, 92:14, 92:20, 93:4, 93:5, 93:7, 94:14, 94:19, 95:11, 97:9, 97:21, 99:2, 100:22, 101:2, 102:10, 102:22, 103:3, 103:7, 103:22, 104:7, 104:13, 104:19, 105:5, 105:16, 105:22, 105:25, 106:2, 106:6, 106:11, 106:18, 106:20, 107:1, 107:11, 107:22, 108:7, 108:12, 109:3, 109:13, 109:22, 110:7, 110:14, 111:1, 111:4, 112:19 popeo 3:15 portion 8:7, 15:24, 20:22, 21:3, 21:17, 22:20, 52:22, 57:14, 94:8 portions 30:10 posed 134:20	posita 34:21, 68:9, 71:7, 71:10, 72:3, 155:9 position 50:11, 52:16, 112:2, 167:5, 167:11, 167:24, 168:7 possessing 55:25, 56:7, 64:19 possibilities 62:11, 155:1 possibility 21:18, 43:5, 45:15, 45:20, 50:18, 52:20, 57:2, 57:9, 58:4, 58:11, 62:11, 67:6, 67:14, 99:3, 99:22, 122:15 possible 12:3, 24:15, 24:19, 25:3, 45:18, 46:17, 46:21, 50:2, 52:15, 61:13, 61:18, 98:23, 122:17, 133:21, 139:3, 140:1, 141:5, 149:23 potentially 9:6, 12:1, 25:25, 29:11, 45:6, 46:15, 54:4, 56:10, 85:20, 131:17, 134:8, 138:2, 149:11, 157:15 powers 114:20 pp 43:3, 43:14, 45:1, 45:16, 45:17, 46:12, 46:13, 46:14,
---	---	---	--

46:22, 46:23, 47:2, 47:7, 47:22, 47:24, 48:2, 48:3, 48:4, 48:6, 48:7, 48:9, 48:11, 48:19, 48:23, 48:24, 48:25, 50:16, 51:8, 51:15, 52:10, 52:23, 53:4, 53:5, 53:12, 54:5, 54:17, 54:20, 55:3, 55:6, 55:7, 55:10, 55:12, 55:14, 55:17, 55:18, 55:20, 56:21 ppu's 45:17, 53:19, 58:5, 58:12, 58:14 ppus 33:19, 41:11, 41:15, 41:17, 41:25, 43:5, 43:10, 44:19, 45:5, 48:10, 49:13, 50:17, 54:5, 54:9, 54:11, 54:14, 55:7, 56:11, 57:23, 64:1, 68:16, 69:13, 70:25, 71:24, 82:23, 87:12, 87:16, 87:20 practice 43:17 preempt 60:16, 157:8 preempting 62:8 preemption 60:15 preface 150:21	preferably 56:2 presence 122:20 present 3:24, 40:23, 104:1, 115:21, 137:23, 159:7 presented 35:11, 46:1, 49:4, 72:19 presenting 68:11, 70:19 presumed 162:18 prevalent 83:22 previous 124:17, 145:23 previously 48:1, 70:18, 143:20, 164:2 primary 63:22 primitive 66:8, 79:9, 79:10, 79:11, 79:18, 79:20, 79:21, 80:2, 96:2 primitives 79:23, 80:1, 80:4, 90:6, 90:13, 90:15, 90:17, 92:19, 93:19, 95:16, 98:13, 99:9, 99:13, 126:25, 148:25 prior 8:20, 11:20, 12:4, 12:19, 36:5, 36:21, 73:13, 114:3, 150:2, 153:19 priorities 60:9, 60:12, 152:12, 152:21,	153:3, 153:11 priority 8:9, 16:1, 55:21, 59:25, 60:3, 60:4, 60:5, 60:14, 60:16, 60:17, 60:18, 62:8, 62:14, 151:24, 152:5, 152:8, 152:12, 152:16, 155:14, 155:15, 155:18, 155:24, 156:2, 156:10, 156:11, 156:12, 157:1, 157:2, 157:3, 157:6, 157:19 probably 117:1, 141:10 problem 36:11, 102:5 proceeding 7:19, 12:15, 12:17, 15:15, 16:5, 32:5, 163:23, 166:16, 166:19, 167:9 proceedings 12:23, 19:9, 35:9, 45:24, 95:6, 167:15 process 13:15, 14:4, 25:20, 37:5, 37:9, 37:11, 38:6, 38:9, 38:12, 38:14, 38:19, 58:17, 103:7, 129:9, 129:15, 130:20, 146:12, 160:5, 164:5 processed 38:10, 66:8, 66:10, 66:17, 109:9 processes 37:21, 43:18,	90:8, 90:25, 149:12 processor 125:23, 131:20, 131:22, 133:16, 134:3, 135:3, 135:14, 138:3, 138:10, 146:18 processors 131:16, 131:17, 132:4, 133:10, 133:11, 133:16, 134:8, 135:6, 137:13, 137:23, 138:2, 138:24, 158:16, 164:20 program 17:4, 17:17, 22:13, 78:23, 96:18, 100:23, 104:17, 116:6, 139:1, 139:5, 139:24, 141:2 programmed 109:19, 111:25 programming 22:13, 22:15, 23:3, 24:13, 76:3 programs 66:11, 119:4 progresses 85:11 prominent 62:15, 62:17, 62:18 proof 162:2 properly 28:15 property 4:27 proposed 63:18, 69:22 prosecution 12:19, 42:12 provide 19:10, 30:1,
--	---	--	---

36:19, 37:3, 57:25, 106:1, 112:14, 118:15, 124:8, 144:10, 147:7, 148:4, 156:4 provided 61:9, 75:18, 83:3, 91:14, 99:1, 109:21, 156:15 provides 33:18, 109:3, 126:4 providing 92:1, 106:12, 112:19 pst 172:16, 172:17 ptab 12:24, 113:2 ptab's 13:25, 95:5 public 166:10, 172:3, 173:9 published 149:19 pulled 46:14, 49:1 pulling 15:9 pure 136:18 purpose 8:13, 115:22 purposes 162:20 pursuant 157:1 push 84:10 put 12:25, 32:2, 48:16, 141:18, 149:8 puts 47:2, 56:20,	108:9 putting 11:3, 131:17 Q question 13:5, 13:8, 14:18, 14:22, 14:25, 19:3, 19:12, 19:19, 19:20, 21:7, 24:6, 24:22, 25:25, 26:23, 27:11, 28:8, 28:14, 35:8, 36:17, 37:10, 37:21, 38:25, 45:8, 46:17, 47:17, 50:6, 50:10, 52:9, 54:22, 55:15, 58:8, 65:1, 65:2, 70:7, 71:6, 71:10, 73:8, 78:3, 78:4, 78:8, 78:11, 79:19, 86:5, 93:21, 94:1, 94:3, 95:17, 98:11, 99:7, 100:14, 100:25, 118:14, 121:3, 122:7, 127:14, 130:13, 130:14, 132:19, 132:23, 134:14, 134:20, 134:21, 138:15, 141:22, 143:12, 143:15, 144:23, 145:10, 149:18, 150:17, 150:22, 154:16, 157:16, 160:13, 161:8, 163:22, 167:23 questioning 80:12 questions 28:21, 153:17,	169:3 queue 83:6 queues 82:13 quick 31:16, 141:23, 168:17 quickly 46:21, 97:24 quote 34:6, 123:15 quoted 42:10, 52:19, 98:12, 107:10, 154:17 R race 83:24 raise 6:7 raised 28:7 raising 11:2 ram 148:18 rasterization 126:25 rasterizer 93:13, 96:10 rather 24:3, 31:13, 42:4, 61:6, 88:16, 109:18, 129:17, 140:8, 158:8 rationale 72:6, 86:22, 117:23 rdokhanchy@mintz 3:21 re-ask 99:7, 132:23 react 146:19 reactive 146:16, 146:19,	149:9 read 10:2, 46:25, 47:4, 48:13, 50:14, 51:1, 52:12, 55:5, 56:22, 57:20, 69:14, 71:1, 71:2, 83:16, 95:22, 101:9, 101:12, 102:15, 104:25, 105:5, 105:14, 120:24, 121:2, 128:4, 128:6, 129:19, 138:20, 139:23, 140:5, 144:9, 144:12, 145:3, 145:5, 155:7, 157:25, 159:9, 159:24, 160:6, 163:9, 163:10, 170:15 reading 53:22, 53:23, 53:24, 95:25, 99:24, 138:18, 172:18 reads 69:3, 69:22, 170:15 ready 84:21, 121:17, 153:9 realize 96:13 really 6:19, 20:23, 24:22, 26:21, 27:19, 33:22, 45:14, 52:15, 75:1, 91:23, 92:21, 98:5, 98:17, 101:8, 140:21, 142:20, 143:11, 145:13, 150:7, 150:9, 153:16, 154:15,
---	--	---	---

156:21 realtek 1:7, 2:7, 3:3, 5:6, 5:21, 6:2, 166:19, 166:24, 170:3 realtek's 167:5 realtek-amd_ohs@- orrick 3:11 realtime 10:1 reason 6:23, 21:21, 59:25, 117:24, 137:21, 150:11, 168:2 reasons 22:10, 60:10, 61:20, 116:10, 170:15 recall 16:22, 19:6, 42:2, 44:6, 44:7, 52:11, 58:18, 71:14, 82:3, 119:8, 120:20, 122:17, 127:2, 146:8, 152:6, 152:7, 160:16, 165:25, 166:5 receive 32:11, 33:23, 59:24, 64:19, 65:3, 68:22, 68:25, 72:21, 95:9, 111:3, 128:4, 155:22, 165:13, 165:18, 165:22, 166:2, 168:4 received 91:6, 91:16, 97:2, 100:1, 100:15, 155:14, 155:17, 156:18,	167:2 receives 39:1, 39:6, 66:7, 81:22, 96:1, 96:5, 96:20, 98:13, 98:14, 99:12, 99:13, 161:16, 161:19 receiving 66:15, 72:11, 96:14, 98:15, 98:16, 99:9, 108:17, 108:22, 108:23, 108:24, 112:1, 117:13 recent 6:22 recently 6:18 reception 66:19 recess 31:22, 63:5, 89:8, 101:23, 123:3, 163:17, 168:25 recited 80:25 recollect 57:22 recommended 4:34 reconfigurable 91:14, 92:17, 94:11, 112:11 reconfigure 91:3 reconfigured 100:13 reconvene 168:22 record 31:21, 31:24, 47:4, 63:4, 63:7, 89:5, 89:7, 89:10, 95:22, 100:20,	101:22, 101:25, 102:16, 105:1, 123:2, 123:5, 138:19, 138:21, 160:6, 163:16, 163:19, 163:22, 164:2, 166:17, 168:24, 169:2, 169:9, 172:10 redefine 34:24 redirect 168:13 redundant 132:1, 132:24, 136:15 refer 7:24, 53:2, 53:11, 53:12, 89:23, 101:17 reference 4:20, 4:22, 4:25, 4:30, 9:7, 32:7, 32:25, 41:7, 42:7, 42:12, 42:20, 43:2, 47:1, 50:7, 52:24, 57:18, 64:23, 69:23, 71:3, 79:14, 83:2, 88:13, 89:15, 103:4, 103:12, 107:2, 114:24, 115:2, 119:6, 123:11, 123:19, 129:6, 129:23, 130:1, 130:6, 130:22, 131:10, 136:18, 153:12, 157:24 references 12:5, 12:20, 36:5, 36:21, 52:23, 79:15, 107:18, 114:4, 160:2 referred 151:23	referring 26:3, 36:19, 89:25, 94:23, 130:25, 142:14, 142:15, 143:25, 144:7, 144:14 refers 74:19, 143:20 reflected 167:13 regarding 4:15, 7:5, 15:14, 15:20, 21:1, 28:11, 32:6, 78:17, 153:20, 161:2, 164:17 regardless 140:20 register 84:7, 159:7 registers 92:23, 104:10, 106:10, 160:1, 160:11, 160:17, 162:8 relate 118:12 related 142:17, 143:5, 172:22 relates 142:5 relation 21:2, 167:14 relationship 10:9, 13:12, 13:13, 128:8 relatively 135:10 release 39:9 released 39:18 rely 16:17, 41:7, 59:10, 89:15, 119:1
--	--	--	--

relying 92:9 remaining 68:17, 126:23 remains 33:7, 33:11, 140:22, 152:8 remedy 4:35 remember 20:17, 20:23, 116:23 remote 5:11 remotely 1:20, 2:20, 5:1, 5:14, 6:20 render 23:6, 23:12, 30:19, 42:7, 71:18, 76:9 rendered 12:14, 75:2, 75:19 rendering 13:7, 13:10, 17:22, 30:20, 75:12, 76:13, 76:17, 77:1, 77:3, 79:4, 80:5, 82:6, 103:23, 104:2, 104:5, 104:6, 104:8, 105:23, 106:8, 118:16 renders 69:3, 69:15 repeat 14:22, 57:8, 62:21, 121:3, 145:12 repetitive 132:20, 132:22, 145:12 rephrase 134:14, 150:20 report 4:14, 13:2,	15:14, 18:4, 18:17, 19:5, 19:20, 19:21, 19:24, 20:14, 20:20, 20:22, 21:9, 21:15, 27:7, 28:6, 42:4, 42:6, 42:16, 102:8, 103:17, 160:15, 161:2, 161:3, 161:14, 162:1, 162:9, 167:8 reported 1:24 reporter 2:25, 6:4, 58:7, 125:8, 141:9, 172:2 reports 19:7, 167:13 represent 5:15 representation 114:17 representations 23:24, 76:23 represented 172:13 representing 5:12, 6:5 require 75:15 required 68:22, 69:24, 92:16, 103:3, 112:14, 167:16, 168:9 requirement 11:10, 18:12, 32:11, 32:21, 86:16, 119:23, 123:13, 161:24 requirements 168:3 requires 25:23, 26:10, 40:8, 41:21,	43:7, 56:9, 113:13 research 149:19, 149:21 reservation 121:14 reserve 46:19, 50:8, 55:9 resolutions 76:10 resource 118:20, 118:22, 119:11, 119:16, 119:17 resources 26:5, 132:25, 133:8, 133:9, 135:10, 156:12 respect 73:6, 105:16, 150:9, 158:20, 167:10 respectfully 139:19 respective 102:23, 103:2, 104:12, 110:4, 129:15, 130:20, 131:8, 172:20 respond 74:1, 74:9, 74:11, 138:4, 150:8, 150:11, 157:12, 158:2 responded 120:23, 138:5 respondents 166:20, 166:23, 167:10, 167:24 responding 13:21, 116:1 responds 147:15 response 36:19, 37:25, 57:25, 58:1, 73:23, 90:1,	101:6, 110:18, 113:10, 129:21, 137:16, 137:17, 138:5, 146:15, 147:7, 147:8, 147:9, 147:10, 167:8 responses 74:2 responsible 131:23 responsive 26:21 rest 34:2, 95:1, 120:7, 135:25 restart 58:8 restate 5:24, 13:7 restructure 22:9 result 10:11, 10:14, 10:20, 11:1, 11:4, 11:5, 14:6, 14:8, 17:18, 17:23, 18:9, 18:11, 18:14, 19:17, 21:19, 21:25, 22:1, 26:17 results 11:10, 56:3, 111:19, 116:13, 134:15 retire 31:14 retrieve 151:1 reverse 24:7 review 4:12, 7:4, 102:8 reviewed 164:2 rewrite 22:9
---	---	--	--

rewritten 22:21 reza 3:16, 5:17 rise 17:6, 20:7, 21:4, 62:19, 152:1, 155:11 road 3:17 routed 26:12, 95:11 rpr 1:24 rsr 1:24 run 22:11, 60:18, 61:2, 61:14, 61:19, 82:10, 83:24, 84:22, 85:1, 96:23, 121:18, 121:19, 121:25, 128:9, 131:7, 131:9, 132:7, 132:18, 133:21, 135:7, 135:18, 135:21, 148:5, 157:7 running 25:15, 25:17, 25:19, 36:11, 43:9, 56:8, 59:14, 59:16, 60:22, 62:8, 62:15, 65:23, 84:13, 100:23, 111:14, 111:18, 112:15, 129:4, 130:6, 130:9, 133:14, 135:22, 137:21, 139:16, 139:18, 149:3, 155:3, 157:9, 157:14 runs 60:4, 131:24 S said 7:24, 8:25,	16:13, 16:23, 17:9, 17:15, 17:16, 18:5, 19:14, 19:16, 19:25, 20:2, 20:7, 20:11, 20:17, 27:9, 34:13, 34:25, 37:20, 38:2, 38:7, 86:24, 110:18, 118:23, 129:8, 129:20, 132:24, 133:2, 140:12, 144:9, 145:13, 157:5, 157:21, 157:22, 160:16, 163:8, 172:19 same 16:4, 17:19, 18:9, 18:10, 22:2, 22:12, 23:4, 25:18, 27:6, 28:11, 28:19, 28:20, 33:2, 35:3, 35:6, 35:22, 36:3, 38:5, 43:9, 45:18, 55:3, 59:16, 67:11, 68:23, 69:1, 70:1, 70:2, 70:16, 76:23, 77:21, 80:14, 104:15, 105:21, 106:23, 126:19, 128:3, 128:18, 130:3, 130:10, 132:8, 132:13, 132:18, 132:25, 133:6, 133:13, 134:15, 135:19, 138:12, 138:23, 139:11, 139:18, 140:3, 140:6, 140:16, 142:14, 142:15, 142:19, 143:6,	152:4, 152:11, 152:16, 153:4, 164:10, 167:8, 168:2, 168:10 san 3:19 sand 37:9 satisfies 32:20 satisfy 119:23 saving 155:4 saw 53:5 say 7:13, 10:3, 11:17, 16:19, 18:4, 25:1, 25:12, 25:18, 26:2, 27:12, 27:14, 33:25, 35:19, 38:5, 45:13, 46:10, 46:24, 48:22, 50:9, 52:9, 52:16, 53:21, 58:24, 64:18, 68:13, 72:20, 73:24, 74:19, 78:5, 79:22, 81:25, 83:2, 83:9, 88:3, 91:18, 93:2, 94:24, 100:14, 107:17, 113:20, 114:13, 117:5, 119:25, 121:8, 122:15, 125:20, 127:14, 129:16, 133:2, 134:22, 139:20, 140:12, 141:16, 143:19, 148:16, 150:11, 150:14, 151:7, 151:24, 158:13, 159:3, 160:21,	162:1, 162:4, 162:7, 164:1, 164:16, 165:11 saying 11:18, 14:13, 17:3, 18:2, 28:16, 29:10, 47:13, 48:13, 51:22, 52:2, 52:3, 52:8, 54:2, 54:10, 57:10, 66:14, 69:20, 74:7, 78:7, 83:14, 88:4, 97:6, 98:7, 99:5, 100:12, 119:24, 136:4, 137:17, 137:18, 139:4, 140:7, 140:8, 140:14, 141:5, 142:15, 142:16, 142:22, 142:25, 144:19, 145:4, 150:10, 153:6, 156:3, 156:7, 156:10, 156:24, 161:23, 166:23 says 39:9, 48:3, 51:6, 53:16, 76:4, 93:22, 94:4, 95:18, 102:22, 103:6, 105:15, 107:2, 107:22, 108:12, 109:22, 111:17, 112:22, 120:1, 121:17, 124:10, 129:12, 130:14, 138:8, 138:23, 139:8, 139:21, 140:5, 142:8, 144:13, 147:7, 154:20, 155:8, 156:11, 162:14 scaling 38:18
--	--	---	---

scenario 61:4, 61:5, 61:11, 141:24, 158:2 scenarios 60:19, 62:7, 62:16, 62:17, 62:18, 133:15 scene 30:16, 76:18, 76:23, 77:4 scheduler 91:24, 102:11, 104:20, 105:6, 107:3, 107:4, 107:5 scheduling 26:5, 68:23, 100:7, 100:15, 100:18, 100:20, 112:1, 168:5 screen 58:23, 75:19, 76:10, 123:18, 126:15, 129:11 screen; 77:5 scroll 16:16, 17:1, 89:18, 166:15 second 16:21, 61:11, 69:6, 70:18, 82:19, 83:12, 83:15, 120:3, 123:15, 127:5, 135:16, 142:8, 142:12, 142:20, 161:1, 161:21 section 4:33, 8:4, 15:19, 20:20, 20:21, 20:24, 20:25, 21:8, 21:15, 124:16, 126:7, 126:12, 127:6, 127:7, 127:8, 127:19,	162:14 see 18:3, 20:12, 21:22, 22:14, 24:23, 29:18, 29:21, 30:6, 41:10, 41:13, 41:17, 41:24, 45:4, 45:15, 47:19, 47:20, 48:19, 50:14, 52:12, 64:9, 64:21, 64:25, 65:7, 65:9, 66:19, 80:22, 81:2, 81:5, 90:11, 98:15, 101:9, 108:21, 115:8, 119:22, 121:11, 124:3, 126:18, 134:14, 135:9, 138:6, 145:6, 152:24, 157:24, 158:1, 160:2, 162:3, 166:13, 166:22, 167:3, 168:18 seeing 25:20 seems 46:21, 49:8, 50:14, 52:19, 78:18, 131:20 seen 45:24, 50:11, 50:13, 83:8, 85:21, 109:6 sees 72:4, 113:3 segment 79:21, 80:3 selection 92:18 self-driving 146:22 semiconductor 1:7, 2:7, 3:3, 5:6, 170:3	send 88:3, 88:4 sending 66:15, 66:18, 66:19, 100:24, 106:14, 147:17 sends 65:13, 87:25, 105:17 sense 68:21, 92:8, 120:25, 121:2, 144:2, 152:25 sensor 147:13, 147:16 sensors 147:11 sent 65:20, 93:5, 93:20, 93:23, 95:16, 100:3 sentence 35:13, 65:16, 66:14, 67:7, 95:1, 105:4, 120:25, 121:2, 138:20, 145:18, 155:7 sequence 9:24, 10:5, 23:18 sequencer 158:24, 159:4 series 41:11 serve 53:20 serves 86:9, 102:10 service 156:5 servicing 112:4 set 71:5, 78:24, 81:11, 92:2, 92:19, 93:14, 94:18, 96:21,	123:24, 124:1, 124:3, 137:8, 137:25, 172:7, 173:3 setting 114:1 seven 136:13 several 103:22, 105:23 shader 32:10, 32:16, 32:21, 32:23, 33:3, 33:9, 33:12, 33:21, 34:3, 34:6, 34:16, 37:19, 38:14, 38:17, 39:22, 40:11, 59:23, 68:13, 68:16, 69:12, 74:20, 78:23, 78:25, 89:20, 89:21, 90:7, 90:19, 90:20, 90:24, 90:25, 91:2, 91:7, 91:15, 91:19, 92:3, 92:4, 92:15, 92:16, 93:13, 93:14, 93:15, 93:20, 93:24, 95:7, 95:8, 95:9, 95:16, 95:20, 95:24, 96:1, 96:5, 96:10, 96:14, 97:17, 97:18, 98:12, 98:14, 98:15, 98:22, 99:8, 99:11, 102:14, 103:11, 104:9, 105:10, 106:9, 107:13, 107:16, 108:15, 110:2, 110:3, 119:3, 119:13, 119:25,
--	---	---	--

120:7, 120:13, 120:14, 122:7, 122:8, 122:13, 123:13, 123:23, 124:9, 165:12 shaders 89:23, 89:24, 91:10, 91:16, 91:21, 92:7, 92:12, 93:6, 95:24, 98:7, 100:2, 100:6, 100:10, 100:13, 100:17, 101:3, 102:13, 103:10, 105:17, 106:16, 108:10, 108:11, 108:13, 108:16, 108:22, 109:8, 109:16, 109:19, 110:18, 110:20, 111:2, 111:24, 112:10, 113:7, 119:4, 120:12, 121:9, 122:12 shading 31:1, 38:20, 77:8 shadow 96:9, 96:23 shape 76:20 share 47:3 shared 40:21, 41:21, 43:14, 43:21, 151:9 sheet 170:1 shift 113:24, 114:15, 114:16, 114:19, 114:21, 116:9, 116:10 shifting 114:10, 114:13, 114:14	shine 30:14 shiny 30:13 short 62:24, 88:15 shorthand 2:24 should 18:24, 25:12, 30:13, 42:16, 91:1, 91:6, 91:11, 127:17, 131:22, 138:1, 147:16, 154:20, 154:22, 168:6, 170:15 shouting 156:1 show 21:21, 30:13, 42:6, 42:14, 48:2, 71:15, 102:9, 163:7 showing 17:21, 20:5, 20:22, 20:25, 45:21, 54:12, 59:22, 76:14, 142:2 shown 21:3, 44:11, 44:12, 44:13, 44:24, 47:20, 54:17, 54:22, 54:24, 55:4, 55:12, 86:25, 87:3, 87:4, 87:12, 91:24, 94:9, 94:14, 138:8, 140:7, 140:8, 140:11, 140:12, 140:16, 141:17, 144:15, 145:8 shows 29:19, 47:16, 47:24, 117:8,	118:2, 142:6 sic 56:23, 104:2, 138:13, 160:8 side 49:4, 74:8, 74:15 signal 39:8, 86:9, 147:17, 148:9 signals 23:19 signature 170:30 signature-b7fzp 173:6 signing 172:19 simd 130:15, 130:23, 131:1, 131:5, 131:12, 131:19, 132:6, 132:11, 132:13, 132:18, 133:5, 134:16, 135:2, 135:21, 136:9, 136:24 simds 132:17, 134:18, 136:12 similar 87:13, 149:13 similarly 45:2, 160:19, 164:8, 166:1 simple 11:13, 14:7, 18:10, 18:16, 19:18, 22:9, 23:5 simplify 154:1, 154:4 simply 39:2, 132:16, 160:13, 164:16, 167:23 simultaneously 26:15, 103:8,	103:9, 126:9, 139:1, 139:5, 139:24, 141:1 since 43:23, 129:7 single 11:9, 11:11, 14:6, 14:12, 18:11, 18:13, 18:15, 18:20, 19:11, 19:17, 20:2, 20:16, 21:11, 21:19, 22:1, 22:16, 23:24, 24:19, 45:20, 52:23, 54:20, 59:18, 61:25, 68:7, 72:10, 82:22, 84:7, 96:3, 130:16, 130:17, 131:7, 131:19, 132:10, 132:11, 133:5, 134:16 single-entry 83:5 singular 131:20 sir 7:21, 8:10, 10:8, 12:8, 15:17, 16:12, 17:13, 29:15, 29:21, 31:15, 31:19, 32:8, 32:13, 32:18, 34:3, 34:9, 41:9, 41:14, 47:17, 47:19, 51:10, 51:13, 53:22, 54:10, 59:5, 59:9, 63:15, 64:4, 64:12, 65:10, 67:23, 68:2, 68:8, 73:17, 74:24, 79:19, 80:19, 80:23,
---	---	---	--

81:5, 87:10, 87:15, 87:19, 87:23, 88:2, 88:6, 89:17, 89:22, 90:12, 90:22, 105:11, 109:24, 111:23, 112:25, 121:12, 123:17, 123:20, 124:6, 125:1, 125:17, 125:22, 132:15, 133:2, 136:4, 139:7, 139:19, 145:21, 146:5, 150:7, 150:9, 156:20, 164:24, 165:16, 166:21, 167:4, 168:11 sit 9:15, 86:25, 136:1, 153:16 sits 92:22 sitting 87:5, 135:25 situation 59:23, 61:19, 82:11, 84:24 situations 60:21, 146:15 six 126:23 sizes 76:11 skill 69:4, 70:20, 71:16 skilled 12:20, 34:21, 42:9, 43:11, 43:24, 53:19, 57:1, 69:16, 69:18, 71:4, 72:14, 72:20, 73:8, 73:11, 82:4, 83:20, 84:2, 95:2,	96:12, 99:24, 107:8, 109:2, 113:9, 116:15, 119:9, 128:5, 129:5, 129:21 skinning 90:9, 90:21, 91:5, 91:8 skipped 138:20 slot 120:19, 121:22 slots 26:13, 106:1, 121:10, 121:13, 121:23, 121:24, 122:1 slow 84:14, 141:8 slower 56:7 slowly 98:6 small 14:8, 62:5, 112:7 smallest 8:22 smart 156:9 so-called 39:14 software 92:22, 93:4, 93:11, 94:20, 99:1, 100:23, 111:6 softwares 147:23 some 10:16, 11:23, 14:14, 16:25, 24:15, 30:1, 35:2, 39:7, 39:8, 39:15, 39:16, 39:18, 41:18, 45:17, 46:10, 48:24,	50:20, 51:6, 53:19, 56:11, 56:15, 56:23, 57:3, 60:20, 65:7, 66:6, 75:6, 76:9, 82:14, 83:4, 84:21, 91:1, 91:4, 93:18, 95:9, 100:2, 104:14, 110:17, 111:10, 114:10, 114:14, 116:13, 117:19, 121:20, 124:24, 126:1, 132:21, 136:5, 139:21, 140:2, 146:13, 146:18, 149:2, 149:22, 151:8, 151:10, 151:15, 153:21, 157:8 somebody 53:13 somehow 74:1 someone 133:25 something 7:1, 22:6, 26:7, 26:19, 30:19, 37:6, 37:8, 37:20, 38:7, 38:10, 38:14, 39:1, 45:22, 48:5, 49:4, 78:6, 92:10, 99:6, 105:15, 113:19, 124:4, 143:20, 143:24, 144:24 something's 146:22 sometimes 22:16, 24:9, 24:15, 27:20, 27:22, 27:25, 61:18, 74:1,	79:4, 114:21, 116:10 somewhat 145:11 sorry 9:25, 18:24, 34:13, 36:16, 47:19, 52:3, 59:1, 63:17, 64:22, 65:1, 74:8, 78:2, 80:11, 90:2, 94:4, 95:8, 102:4, 120:21, 120:23, 124:11, 126:6, 131:15, 132:8, 132:20, 139:16, 143:15, 159:13, 167:17 sort 18:13, 69:2, 70:6, 75:22, 100:23, 114:14, 118:15, 146:11, 167:22 sorts 112:22 source 75:13 sources 30:15, 75:13 space 47:3, 47:14, 54:6, 56:10, 111:15, 111:18, 158:1 spawn 13:18, 14:12 spawned 10:12, 10:21, 25:21 spawns 152:9 speaking 24:17, 167:21 special 144:21 specific 26:2, 37:24,
---	---	---	---

39:5, 70:14, 108:18 specifically 33:4, 92:6, 95:17 specification 45:14, 46:5, 52:12, 57:21 specificity 95:19 specs 45:21 speculate 78:2 speculation 136:18 spent 133:8 spirit 18:10, 164:10 ss 171:2 stand 169:8 standard 5:11, 43:22, 113:22 standby 61:1 start 19:22, 70:7, 94:25, 101:1, 103:15, 118:5, 155:5, 164:12 started 130:12, 140:6, 144:24 starting 15:19, 46:25, 51:3, 57:8, 65:8, 65:17, 78:22, 95:25, 103:21, 104:10, 106:10, 106:12, 115:14, 121:16, 121:18, 124:10, 124:11 starts 13:13	state 5:15, 8:5, 21:18, 25:3, 34:5, 43:23, 44:3, 55:24, 61:1, 66:9, 68:24, 70:17, 72:12, 72:23, 73:14, 73:15, 73:18, 73:22, 74:18, 74:19, 75:23, 77:11, 77:12, 77:19, 77:22, 77:25, 78:6, 78:7, 78:9, 78:10, 78:13, 78:17, 78:18, 78:20, 78:21, 78:25, 79:3, 79:9, 79:24, 80:5, 80:12, 82:5, 88:10, 98:14, 101:2, 120:9, 120:14, 122:1, 126:8, 128:24, 147:6, 147:18, 147:19, 147:20, 148:5, 149:13, 150:3, 155:4, 155:5, 158:21, 158:23, 159:5, 159:16, 159:19, 159:23, 160:1, 160:11, 160:14, 160:17, 160:20, 160:25, 161:4, 161:16, 161:25, 162:2, 162:5, 162:7, 162:10, 163:1, 163:4, 163:5, 163:6, 163:8, 163:25, 164:5, 164:9, 166:25, 168:1, 168:6, 171:1, 171:8, 172:3, 173:10	stated 21:16, 52:13, 57:4, 65:4, 70:18, 98:12, 99:18, 125:16, 129:8, 136:16, 155:2, 167:15, 167:23 statement 13:1, 14:13, 16:22, 17:2, 20:6, 21:1, 21:14, 86:16, 108:9, 113:6, 127:12, 143:2, 143:3, 145:15, 154:20, 154:25, 168:11 statements 14:16, 50:2, 128:23 states 1:1, 2:1, 4:31, 5:7, 20:13, 21:3, 90:6, 110:1, 127:20, 138:11 stating 48:18 stations 121:14 status 137:1, 137:6, 158:16, 160:21, 160:23, 161:15, 161:16, 164:15, 164:19 stay 44:17 stenographer 6:7, 7:12, 18:25 step 125:18 stephanie 3:27 steps 94:11	stick 130:13 still 17:11, 76:13, 84:8, 136:21, 138:3, 165:21 stop 38:4, 57:7, 104:24 storage 39:10, 79:2, 155:5 store 82:8, 104:14, 113:23, 116:11, 116:12, 158:25, 159:4 stored 39:10, 79:2, 79:3, 125:7, 125:12, 125:14, 149:2 storing 162:16, 162:24 straight 31:13 straightforward 153:2 strategy 167:10 stream 9:23, 10:5, 91:23, 112:24, 133:13, 138:12, 139:11, 140:3, 140:17 streaming 61:9 streams 97:15, 131:18, 135:7 street 3:6, 76:5 strip 96:4 strong 132:4 structure 39:14, 109:6
--	--	--	---

structured 87:13 structures 11:2 stucco 11:3 stuttard 4:30, 123:11, 123:19, 123:22, 123:25, 124:7, 127:2, 127:25, 128:7, 128:17, 128:21, 128:23, 129:1, 129:6, 129:22, 131:3, 131:10, 131:11, 133:4, 133:8, 133:21, 136:3, 136:17, 136:19, 137:19, 140:23, 141:15, 142:5, 144:7, 146:3, 148:3, 151:22, 152:3, 152:13, 153:12, 154:11, 154:13, 155:8, 157:17, 158:12, 161:1, 161:14, 164:11, 164:18, 165:4, 165:22 stuttard's 130:6, 156:7, 156:8, 157:6, 157:24, 158:2 stutzmann 3:27 sub-subtasks 27:4 subarrays 135:12, 135:17 subclassification 36:24 subcomponent 32:19, 44:4, 44:5 subcomponents 32:20, 42:22, 42:23	subintermediate-- level 27:16 submitted 167:8, 167:14 subsequent 111:9 subsequently 10:12 substance 161:22 substantially 103:8 substantiate 42:13 subsystem 33:19 subtasks 10:19, 11:5, 13:21, 27:4, 29:25, 152:20 subtraction 114:9 suddenly 111:19, 137:8 suffice 131:21 suite 3:18 sun 76:6 support 4:10, 25:23, 25:24, 61:23, 129:4, 149:11 supported 71:2 supposed 91:12 supposedly 76:8, 127:17 sure 17:2, 28:14, 29:11, 37:8, 45:13, 47:25, 48:8, 51:20, 62:21, 70:2, 77:22, 78:13,	78:19, 83:23, 84:19, 102:1, 107:21, 116:25, 120:21, 121:5, 127:13, 136:20, 139:13, 152:18, 153:24, 168:21 surface 30:8, 30:13, 30:14 surrounding 30:22 sutcliffe 3:4, 5:21 swap 60:3, 60:21, 61:2, 61:13 swapped 59:14 swapping 59:11, 59:12, 59:13, 59:19, 60:7, 60:11, 61:21, 62:5, 62:7, 62:19 swaps 62:13, 62:14 switch 60:2, 154:13, 154:17, 154:21, 154:24, 155:2, 157:18 switching 60:1, 154:7, 154:12 sworn 6:6, 6:8, 172:8 synchronization 26:6, 86:10, 86:18 synchronous 83:21, 84:9, 85:8, 85:16, 86:4, 86:7, 86:8, 86:14 synonym 34:25 system 25:16, 30:4,	38:24, 39:1, 40:21, 41:19, 41:22, 42:17, 43:4, 43:13, 43:14, 43:25, 45:19, 47:7, 47:8, 47:22, 49:21, 49:23, 50:4, 54:5, 56:7, 57:2, 57:23, 58:16, 60:6, 67:8, 69:14, 70:16, 71:14, 71:18, 82:11, 84:14, 84:16, 85:7, 85:14, 85:17, 85:18, 86:10, 86:17, 92:23, 94:13, 96:7, 111:22, 111:23, 112:4, 112:5, 114:17, 117:9, 118:3, 118:6, 118:9, 118:11, 118:15, 118:19, 119:12, 119:17, 119:24, 120:1, 122:12, 126:21, 127:4, 128:2, 128:17, 131:11, 133:8, 133:21, 135:10, 135:20, 136:2, 136:3, 137:19, 138:14, 140:23, 142:5, 145:2, 146:7, 146:11, 146:13, 146:16, 146:19, 147:1, 147:3, 147:5, 148:3, 148:5, 149:4, 149:9, 151:10, 153:12, 155:15, 155:21, 155:23, 156:8, 156:9, 156:14, 157:6, 157:11, 157:17,
---	---	--	---

158:2 systems 30:3, 55:25, 85:14, 85:21, 85:23, 112:20, 114:18, 147:11, 147:12, 147:23, 147:24, 147:25 systolic 40:17	127:3, 150:3, 152:19 talked 119:4, 122:4, 125:19, 127:7, 133:24, 148:25, 154:6 talking 23:1, 39:5, 44:18, 57:8, 78:20, 81:12, 98:5, 119:21, 130:12, 134:12, 136:15, 142:23, 143:11, 150:1, 163:3 talks 105:21, 142:20, 144:16 targeted 50:5 tasked 58:1 tasks" 78:14, 123:16 tasks; 98:18 taxonomy 28:3 teaches 161:15 teachings 122:19 technical 24:18 technician 3:26, 7:10, 15:9 technologies 1:10, 2:10, 3:13, 5:7, 170:4 tell 6:11, 10:2, 25:20, 27:9, 29:24, 46:6, 67:4, 82:4, 91:16, 91:19, 137:15, 149:18	telling 67:9, 96:18, 98:24, 107:13, 112:13 tells 55:6, 91:7, 146:18, 147:15 temporarily 82:8 ten 31:17 ten-minute 31:5, 122:23, 163:13 tens 25:17 term 9:9, 9:13, 20:8, 26:10, 34:16, 34:19, 41:1, 49:17, 71:13, 114:6, 162:13, 162:15, 162:17 terminology 35:2 terms 29:5, 76:3, 123:23, 126:1, 135:11, 155:4, 165:9 tessellation 38:22, 96:8, 96:23, 98:21 testified 6:12 testify 7:4 testimony 6:24, 27:6, 58:3, 58:10, 58:15, 58:16, 172:10 texas 3:8 text 100:12, 143:25 texture 30:23, 31:1,	74:19, 74:23, 75:1, 77:9, 78:25 textures 38:20, 74:25 th 3:7, 173:4 thank 6:3, 6:21, 7:14, 31:19, 37:5, 46:8, 74:16, 88:6, 90:4, 122:25, 141:11, 145:24, 150:24, 151:20, 163:14, 168:15, 169:6 that; 88:9 themselves 5:15, 40:19, 43:6, 80:6, 111:2, 149:15, 151:3, 151:6, 157:20 theory 63:11, 120:18 therefore 21:24, 33:7, 35:17, 42:9, 46:17, 60:25, 69:3, 91:15, 92:5, 97:24, 142:16, 150:8, 157:2 therein 109:9 thing 13:6, 30:2, 31:2, 38:5, 61:22, 69:20, 70:1, 70:2, 85:2, 96:24, 108:18, 109:23, 118:8, 119:14, 133:23, 143:6, 147:14 things 19:6, 24:13,
T			
ta 4:36 ta- 15:15 table 20:5, 104:11 tag 157:11 tagged 39:19 take 21:12, 22:8, 30:2, 31:5, 31:9, 37:14, 38:23, 62:24, 65:11, 86:23, 88:14, 98:3, 105:4, 107:19, 114:25, 117:4, 122:22, 136:7, 140:19, 155:1, 158:6, 163:12 taken 2:19, 19:8, 42:16, 50:11, 172:9 takes 43:3, 85:11 taking 5:16, 23:4, 39:20, 102:18, 104:25, 125:18 talk 45:5, 63:24, 64:17, 74:18, 106:15, 124:22,			

30:20, 68:10, 70:10, 70:11, 71:11, 72:7, 77:3, 84:16, 103:16, 108:4, 108:15, 112:8, 130:12, 137:15, 138:2, 143:7, 148:12, 149:20, 149:24, 155:4 think 10:16, 27:9, 34:19, 36:11, 37:24, 38:4, 62:22, 70:6, 82:16, 84:2, 86:24, 91:20, 94:3, 101:13, 117:1, 117:2, 117:10, 118:23, 120:25, 134:12, 136:5, 136:6, 136:9, 136:11, 141:9, 141:16, 145:11, 145:18, 148:3, 152:12, 152:23, 152:24, 153:17, 156:22, 156:23, 157:16, 157:21, 157:23, 163:7, 167:20, 168:19, 169:5 thinking 55:8 thought 37:1, 145:17 thousands 25:17 thread" 20:3 three 28:1, 68:10, 69:8, 69:19, 69:21, 69:23, 70:9, 70:10, 70:23, 71:8, 71:9, 71:11, 72:4, 72:7,	72:16, 72:17, 77:5, 81:17, 95:24, 103:16, 106:18, 107:18, 107:19, 120:12, 121:9, 139:15, 139:17 three-dimension 76:9 three-level 109:12 through 26:8, 27:2, 29:7, 40:20, 41:12, 41:18, 41:21, 43:13, 43:19, 43:21, 44:25, 45:5, 45:18, 50:18, 51:18, 51:24, 52:6, 63:25, 64:1, 64:16, 64:18, 64:24, 67:11, 92:1, 92:14, 92:22, 93:11, 93:12, 94:10, 94:20, 97:7, 97:17, 97:25, 98:6, 100:24, 103:10, 106:23, 109:8, 110:19, 110:22, 111:3, 111:8, 112:17, 112:21, 117:17, 125:5, 125:15, 125:23, 135:7, 145:16, 148:20, 148:23, 151:7, 163:9, 163:10, 168:18 throughout 35:10 time 5:10, 5:11, 8:1, 12:21, 16:25, 20:25, 23:11, 25:4, 25:18, 31:11,	31:15, 31:21, 31:24, 43:9, 45:7, 55:3, 55:8, 57:18, 57:19, 59:16, 62:23, 63:4, 63:7, 68:23, 69:2, 70:16, 70:21, 82:14, 84:15, 89:7, 89:10, 101:22, 101:25, 103:18, 103:25, 104:15, 105:3, 109:25, 112:5, 123:2, 123:5, 124:12, 126:19, 128:3, 128:19, 130:4, 130:10, 133:6, 133:14, 135:19, 139:3, 139:18, 139:25, 142:8, 151:17, 163:10, 163:16, 163:19, 168:10, 168:24, 169:2, 169:4, 169:10 times 55:6, 73:4 tired 31:10, 31:12 tm 125:5 today 5:11, 6:4, 6:24, 9:15, 17:10, 33:10, 167:12 today's 5:9 together 46:14, 48:16, 49:1, 70:11, 71:11, 131:18, 134:4 token 80:14 told 96:22	took 19:10 top 30:12, 44:6, 64:9, 91:25, 94:3, 109:7, 116:23, 125:7, 125:9, 152:22, 154:16 topic 46:1 topology 96:3 total 134:21, 134:23, 134:24 totality 10:22, 12:18, 58:18, 129:6, 129:22 touched 70:6 toward 31:12 towards 101:7 trace 29:11, 66:24 track 25:9, 25:10, 26:7, 137:1 trademark 1:1, 2:1, 5:8 traffic 76:6 transactions 43:13 transcendental 114:10 transcript 120:24, 168:18 transfer 66:2, 70:15, 93:10, 151:2 transferring 158:25 transform 74:20, 79:1
---	---	---	---

transformation 22:25, 39:15, 77:6 transformations 31:2, 38:18, 75:10, 75:15, 77:7, 90:9, 90:21 transformed 22:22, 110:22 transition 88:12 translate 23:8 translated 111:8 translation 27:3, 27:21, 28:2, 29:6 transmission 86:3 transmits 65:13 transparent 30:9 travel 67:18, 67:19, 67:20 trial 1:4, 2:4 triangle 79:11, 79:21, 80:3, 96:4 tried 150:21 tries 42:6 trigonometry 114:11 trivial 25:2, 25:22 true 49:14, 51:15, 51:16, 110:24, 171:8, 172:10 truth 6:12 truthful 6:24	try 19:20, 29:18, 30:19, 36:13, 43:17, 45:7, 72:15, 75:9, 76:8, 76:12, 84:10, 132:20, 141:11, 156:22 trying 21:21, 49:6, 71:15, 76:1, 88:17, 92:8, 114:19, 114:21, 114:23, 120:24, 141:16, 144:22, 161:20, 161:25 turn 8:3, 101:7, 113:12 two 18:1, 43:22, 48:21, 57:1, 59:17, 62:20, 62:22, 63:21, 68:12, 69:1, 81:10, 86:20, 93:17, 103:25, 114:16, 114:20, 124:8, 134:24, 140:18, 140:20, 143:4, 148:1, 152:20, 153:3, 153:10, 154:23, 162:3, 168:3, 168:9 two-dimensional 40:17 type 40:8, 104:9, 106:9 types 79:23, 79:25, 130:3, 148:4, 149:11 typical 146:21 typically 49:17, 49:18,	75:20, 78:24, 80:15, 82:12, 148:18 U u-1 41:12, 46:14, 46:23, 48:3, 51:17, 51:19, 51:25, 52:4, 52:7 ulc 1:10, 2:10, 3:13, 170:4 unambiguous 25:11 unambiguously 116:16 uncommon 56:13 under 6:11, 16:7, 24:25, 48:20, 83:12, 147:7, 149:8, 152:24, 158:1, 162:13, 162:22, 171:7 undergoes 39:14 underlying 23:7, 23:18 underneath 30:10, 119:12 understand 6:18, 7:3, 8:15, 16:8, 28:5, 28:14, 43:12, 43:25, 46:18, 47:17, 52:2, 52:8, 53:22, 57:21, 71:8, 79:19, 82:20, 83:21, 88:3, 94:1, 94:3, 100:4, 100:9, 107:9, 113:9, 113:15, 127:13, 140:19,	142:22, 143:11, 143:17, 145:3, 145:4, 153:22, 155:9, 156:23, 163:2, 166:18, 167:9 understanding 9:21, 12:18, 17:14, 34:10, 42:12, 54:2, 71:17, 92:9, 109:17, 109:18, 129:5, 144:4, 165:24 understood 23:20, 47:10, 51:2, 51:4, 57:4, 71:3, 89:24, 113:8, 151:19 undertaking 25:2 unfortunately 16:16 unit 8:6, 8:23, 9:1, 10:13, 13:14, 15:24, 22:19, 32:16, 33:4, 33:5, 34:2, 55:20, 61:1, 63:12, 63:13, 63:14, 64:6, 64:8, 64:11, 64:20, 65:3, 66:3, 66:12, 66:15, 66:25, 67:1, 67:19, 68:10, 68:14, 68:15, 69:19, 71:23, 73:18, 73:21, 73:25, 81:21, 82:7, 82:9, 82:21, 83:18, 84:4, 84:6, 84:12, 84:13, 84:19, 84:20, 85:1,
---	---	--	---

87:1, 87:6, 92:14, 93:9, 94:14, 94:19, 95:11, 97:17, 97:20, 100:22, 102:10, 102:22, 103:4, 103:21, 104:3, 104:7, 104:13, 104:19, 105:5, 105:9, 105:22, 106:2, 106:12, 106:14, 106:15, 106:18, 106:20, 107:1, 107:11, 109:3, 109:22, 110:25, 111:4, 111:19, 112:19, 115:15, 115:23, 117:3, 125:12, 151:10, 158:13, 158:16, 159:8, 159:23, 160:3, 160:4, 160:14, 164:9, 164:14, 164:19, 164:23, 166:2 united 1:1, 2:1, 4:31, 5:7 units 13:19, 26:5, 26:7, 41:24, 42:16, 49:20, 69:8, 69:23, 70:23, 72:16, 72:17, 79:2, 81:18, 104:21, 105:7, 110:4, 110:9, 115:16, 121:10, 136:8, 160:12 unmarked 7:24, 8:2 until 37:15 updated 25:13 use 20:8, 28:2,	35:3, 53:2, 53:9, 56:11, 60:19, 60:20, 61:14, 82:10, 84:1, 96:19, 104:4, 118:10, 119:18, 149:5 user 17:21, 17:22, 111:15, 111:18, 126:4 user's 75:2, 76:9, 77:5 uses 158:24 using 26:9, 28:5, 36:12, 53:10, 80:4, 86:8, 86:18, 104:21 usually 26:7, 26:8, 29:8, 74:10, 75:10, 148:16 utilization 61:3 v valid 69:14, 71:1 value 84:8 values 82:14, 83:24, 83:25, 116:12 variety 28:9, 153:11 various 13:19, 28:12, 42:8, 96:8, 109:4, 110:2, 110:8, 112:20, 118:17 vector 76:21 vectors 75:17	version 166:10 versus 5:7, 23:4, 23:9, 23:22, 98:25, 109:23, 128:14, 135:9 vertex 17:24, 23:6, 23:13, 31:3, 36:8, 36:25, 38:8, 38:16, 66:8, 89:20, 90:7, 90:17, 90:19, 90:24, 90:25, 91:2, 91:6, 91:23, 92:3, 92:15, 93:13, 93:20, 93:24, 95:7, 95:16, 95:20, 97:17, 104:11, 104:22, 110:2, 120:14, 122:7, 122:12 vertex-type 30:24 vertices 90:8, 96:1, 96:3, 126:25 via 2:20, 40:6, 47:5, 47:9, 64:19, 65:18, 65:20, 65:25, 165:14 video 5:10, 5:13, 31:24, 76:14, 89:7, 89:10, 101:25 videoconferencing 2:20 videographer 3:25, 5:4, 5:11, 5:23, 6:3, 31:20, 31:23, 63:3, 63:6,	88:18, 89:6, 89:9, 101:5, 101:16, 101:21, 101:24, 123:1, 123:4, 163:15, 163:18, 168:23, 169:1, 169:7 videotaped 5:5 view 54:3, 55:4, 58:18, 69:4, 69:16, 70:20, 71:2, 71:21, 73:10, 74:9, 74:11, 75:5, 75:7, 75:12, 98:2, 128:22, 128:23, 129:23, 145:19 viewing 30:16 violation 4:33 virtual 80:22 virtually 172:15 virtue 156:24 voice 5:14 volume 1:19, 2:18, 96:9, 96:23 w wait 61:10 waiting 60:24 waived 172:19 walls 11:3 want 9:25, 15:18, 50:1, 57:7,
---	--	--	--

60:1, 62:21, 63:24, 64:17, 72:20, 78:2, 88:14, 88:15, 93:17, 108:21, 110:10, 112:23, 115:4, 116:7, 116:10, 123:10, 127:5, 132:22, 134:13, 141:21, 143:12, 145:16, 145:20, 146:23, 158:6, 160:6 wanted 29:13, 48:1, 53:1, 53:8, 136:19 wanting 51:17 waste 132:25 wasteful 132:1, 133:7 way 12:3, 18:10, 19:14, 22:12, 22:22, 28:16, 35:20, 39:24, 40:12, 41:16, 42:1, 43:11, 43:12, 52:16, 54:16, 60:1, 63:24, 63:25, 66:24, 67:4, 67:8, 67:12, 69:2, 69:6, 71:18, 74:2, 78:21, 85:4, 87:13, 88:19, 92:25, 94:12, 97:6, 97:16, 99:23, 106:25, 109:7, 111:5, 111:24, 112:7, 112:17, 112:18, 113:8, 118:18, 132:23, 140:6, 141:18, 144:10,	150:21, 167:23, 172:25 ways 24:6, 43:22, 51:6, 62:4, 62:6, 132:21, 149:25, 150:16 we'll 31:16, 67:15, 89:5, 124:20, 146:1, 146:10, 161:1, 168:22 we're 31:20, 31:23, 36:11, 89:6, 101:8, 101:13, 134:12, 140:24, 140:25, 151:22, 163:15, 163:18, 168:23, 169:1 we've 83:16, 93:18, 114:25, 120:13, 125:18 well-defined 34:23, 85:12 well-known 55:23, 62:6, 147:21 went 26:13, 151:7, 151:11, 162:4, 163:21 whatever 27:18, 71:19, 76:22, 82:6, 114:11, 151:18 whenever 121:17 whereas 31:2 wherein 132:12, 132:17 whereof 173:3 wherever 39:11 whether 9:16, 24:23,	35:15, 48:10, 52:13, 53:12, 54:7, 82:1, 86:13, 122:6, 152:11 white 139:6, 158:1 whole 20:14, 60:15, 61:16, 99:15, 108:3, 118:6, 119:14, 155:7 wilt 4:20, 32:6, 32:16, 32:20, 32:24, 41:7, 41:25, 42:6, 42:10, 42:17, 42:19, 42:21, 43:2, 43:13, 43:25, 44:3, 45:15, 46:4, 46:7, 46:25, 48:14, 50:6, 50:7, 50:22, 51:3, 52:9, 52:19, 53:1, 53:25, 56:18, 57:2, 57:10, 57:18, 57:21, 58:4, 58:5, 58:11, 58:12, 58:16, 58:20, 58:24, 59:3, 59:7, 59:11, 63:12, 64:22, 69:3, 69:7, 69:23, 70:8, 70:10, 70:19, 71:3, 71:7, 71:11, 71:14, 71:16, 71:20, 71:21, 72:4, 79:14, 81:6, 81:7, 81:12, 82:1, 83:2, 84:5, 85:8, 86:3, 102:9,	112:3 wilt's 71:18, 118:9 windows 11:3 withdrawn 32:14, 34:15, 79:8, 86:23, 90:16, 93:2, 115:10, 150:25, 159:13, 164:12, 164:15 within 11:19, 26:13, 33:1, 44:13, 64:2, 71:24, 87:8, 87:18, 87:24, 87:25, 110:8, 117:19, 121:10, 121:22, 132:12, 148:12, 152:4, 154:12, 159:5, 159:15, 160:22, 161:15, 162:4, 162:10, 162:25, 172:3 without 24:1, 24:17, 35:14, 54:12, 84:11, 85:19, 148:6 witness 4:3, 5:22, 6:2, 6:6, 18:24, 31:7, 31:19, 63:2, 88:16, 122:24, 125:9, 163:14, 172:7, 172:11, 172:20, 173:3 wo 4:29 word 8:11, 8:16, 9:3, 9:21, 15:22, 19:25, 34:10, 36:12, 37:21, 39:21,
--	--	---	---

114:1, 143:25, 146:10 words 28:5, 156:17 work 6:20, 8:7, 8:23, 9:2, 10:17, 10:18, 15:24, 22:20, 28:23, 33:5, 63:12, 64:10, 64:14, 64:20, 65:3, 66:3, 66:11, 66:15, 66:25, 67:18, 67:25, 68:14, 71:23, 73:21, 73:25, 83:17, 84:12, 84:19, 84:20, 84:25, 87:1, 87:5, 97:10, 97:11, 119:14, 119:19, 120:8 worker 49:19 workers 49:19, 49:23 working 85:18, 109:16 workload 13:14, 22:19 works 31:6, 67:8, 71:18 world 4:26, 30:3, 50:3, 96:16, 137:22 worst 85:2 worth 74:5 wouldn't 129:17, 134:15, 149:8, 149:14, 151:3, 160:21 wrap 169:5	write 56:3, 151:9 write-after-write 83:23 write-back 113:23 writing 92:23, 139:6 written 19:5, 22:13, 35:13 wrong 82:21 wrote 163:3, 163:4, 163:6 Y yeah 19:2, 56:24, 74:4, 95:22, 101:11, 102:4, 114:15, 120:23, 125:6, 127:19, 155:5, 158:10, 161:20 yourself 57:8 Z zoom 2:20 0 0 45:2, 48:15, 51:8, 53:6, 87:13, 87:18, 87:22, 87:25 0) 51:15 00 4:29 00687 1:14, 2:14, 5:8, 7:5, 170:6 0204 51:15	053 15:14 06 168:24 07 31:21 1 1 32:5, 51:8, 89:10, 101:22, 101:25 10 31:18, 31:21, 31:24, 64:18, 64:24, 89:10, 124:17, 130:17, 173:4 100 117:21 102 124:5, 127:8, 158:15, 160:20, 161:16, 162:10, 165:14 1022 125:11 1024 125:5, 125:12, 158:16, 164:14, 164:19 1026 138:24, 158:16, 164:20 103 160:22 1030 158:17, 161:15, 164:15, 164:19 1036 161:17 104 43:4, 43:14, 47:8, 47:22, 158:14, 159:5, 159:15, 162:5, 164:22, 165:15, 166:15, 166:22	1044 158:24, 159:4 1045 159:4 105 41:18, 43:1, 158:25 106 123:24, 124:2, 126:8, 128:10, 139:2, 165:22, 166:1 1061 159:8 1069 158:13, 159:23, 164:9, 164:23, 166:2 108 158:14, 159:19, 159:25, 164:22, 165:19, 165:23 11 63:4, 63:7, 114:25, 115:14, 118:25, 119:5, 124:9, 125:2, 125:16, 126:6, 126:7, 126:13, 126:16, 127:9, 128:13, 133:24, 138:7, 141:15, 144:7 1103 160:1 114 117:3, 117:4, 117:21, 118:7 116 137:9 118 117:17 12 2:22, 5:2, 5:10, 89:7, 135:22, 135:24, 137:10, 172:16 122 117:18
---	---	--	--

123 4:26 125 158:12 126 116:20, 117:18, 117:21, 159:14, 161:14 127 159:18, 160:9, 162:7 128 134:22, 134:24, 135:1, 135:3, 136:25, 137:9 129 165:11 13 115:1, 118:25 1318 4:36, 15:15 135 146:2 136 151:22, 154:6, 154:9 137 155:6, 155:8 14 65:8, 65:17, 119:5, 166:7 14125 1:25, 2:25 15 4:14, 46:25, 80:25, 104:18, 119:25, 120:1 1596 3:20 16 124:10, 135:5, 135:17 166 4:31 17 31:18 18 2:22, 15:8,	169:2, 169:10, 169:11, 172:17 19 31:11, 31:24, 51:3, 58:25, 59:4, 113:12, 113:13, 119:1, 124:11, 146:4 1980 43:23 1d 40:3 2 2 52:25, 118:4, 123:2, 123:5 20 59:8, 59:22, 88:21, 90:5, 93:19, 123:2, 151:21 200 118:6 2003 166:16 202 32:16, 47:11, 48:15, 51:4, 54:12, 54:23, 57:5 2023 1:14, 2:14, 5:8, 5:9, 7:5, 170:6 2024 1:21, 2:23, 5:2, 170:10, 171:11, 172:16, 173:4 204 43:4, 43:15, 44:8, 44:9, 44:10, 44:12, 44:21, 45:1, 45:9, 45:10, 46:7, 47:7, 47:22, 47:24,	48:3, 48:6, 48:19, 48:23, 51:8, 51:14, 52:23, 53:6, 54:17, 54:20, 55:3, 55:6, 55:11, 55:12, 55:14, 55:17, 102:24, 107:23, 108:1, 108:11, 108:14, 108:23 206 47:9, 64:2, 67:17, 67:24, 81:22, 81:23, 82:1, 83:4, 83:7, 83:15, 102:22, 104:7, 104:19, 108:6, 108:9, 110:7 208 64:15, 64:19, 65:2, 65:13, 65:17, 65:20, 67:13, 67:21, 87:1, 87:6, 87:17, 87:25, 94:6 21 103:6, 162:13 210 33:5, 63:13, 64:11, 64:14, 64:20, 65:3, 66:3, 66:12, 67:19, 67:25, 68:6, 68:15, 70:11, 72:4, 73:21, 81:8, 81:13, 81:18, 82:21, 87:1, 87:6 212 32:23, 33:4, 34:2, 63:13, 64:6, 64:8, 67:18, 67:25, 68:6, 68:14,	70:11, 72:5, 73:18, 81:8, 81:13, 81:18, 81:21, 82:21 214 45:1, 47:6, 47:9 22 160:8 26 102:21, 107:21 29 15:19, 95:25, 123:5 2d 40:3, 126:22, 128:14 3 3 44:11, 57:14, 64:6, 67:20, 99:14, 163:16, 163:19 30 15:18, 15:22 300 3:18 302 47:5 303 33:6, 63:14, 64:16, 65:7, 65:12, 65:18, 66:7, 66:14, 67:12, 67:20, 68:1, 68:6, 68:15, 70:11, 71:24, 72:5, 73:15, 81:8, 81:14, 81:18, 82:22, 83:19, 86:25, 87:5, 87:8, 87:21, 87:22, 87:24, 109:5 304 89:20, 90:7,
---	---	---	--

95:16, 95:20, 108:24, 110:12 306 89:21, 95:24, 96:1, 96:5, 96:10, 108:24, 110:13 308 96:11 31 18:4, 29:16 310 89:21, 108:24, 110:3, 110:13 314 3:20 32 32:4, 63:4, 104:15 320 44:13, 47:6, 47:12, 47:21, 48:21, 54:21, 55:4, 55:13 33 141:14 337 4:33, 4:36, 15:15 35 110:1, 141:15 3580 3:17 36 101:22 37 89:7 38 101:25 381 8:19, 9:11, 9:16, 9:22, 11:20, 12:4, 12:19, 15:14, 32:12, 36:5, 36:20, 42:12, 58:25, 59:4, 59:8, 59:22,	69:3, 69:16, 70:22, 71:5, 71:6, 71:19, 71:20, 72:1, 72:3, 73:5, 73:6, 73:7, 73:10, 73:12, 73:13, 78:22, 79:6, 79:13, 80:17, 80:20, 80:21, 80:25, 89:20, 113:21, 114:3, 122:5, 123:12, 136:17, 146:4, 150:2, 166:25 3; 87:4 3d 126:24, 128:14 3j6ptabdoCKET@or- rick 3:10 4 4 2:22, 168:24, 169:2, 169:10, 169:11, 172:17 40 3:7, 34:4 41 4:19, 8:3, 110:1 414 97:10, 103:7, 103:22, 104:13, 105:22, 111:1 420 104:3, 104:13, 104:21, 105:7, 105:9, 106:15, 108:7, 108:15, 108:23, 110:4, 110:9, 110:13, 110:25, 120:10, 121:10 43 160:8, 163:16	44 63:7, 160:8, 162:14 48 115:2, 115:3 5 5 117:24 510 117:14 512 117:25 514 115:15, 116:17, 116:19, 117:8, 117:12, 118:24, 119:8, 119:10, 120:9 519721 1:26 53 78:22 54 78:23 55 163:19 5th 5:9 6 602 117:9 609 3:6 61 64:6 62182 4:29 6400 3:9 65 103:21 658 3:9 67 160:9 6a 117:7	7 7,742,053 4:17 713 3:9 74 74:18 76 32:9 77002 3:8 8 8,151,095 4:19 8,318,774 4:24 8,854,381 1:15, 2:15, 4:13, 4:17, 4:21, 7:5, 170:8 80 4:21 858 3:20 88 4:24, 58:23, 59:1, 59:2 89 59:6, 59:7 9 9 2:22, 5:2, 5:10, 172:16 92130 3:19 98 121:8
---	---	---	--



Planet Depos®
We Make It *Happen™*

Transcript of Massoud Pedram, Ph.D., Volume II

Date: January 8, 2024

Case: Realtek Semiconductor Corp. -v- ATI Technologies ULC (PTAB)

Planet Depos

Phone: 888.433.3767

Email: transcripts@planetdepos.com

www.planetdepos.com

Transcript of Massoud Pedram, Ph.D., Volume II
Conducted on January 8, 2024

1 (173 to 176)

173

1

UNITED STATES PATENT AND TRADEMARK OFFICE

2

BEFORE THE PATENT TRIAL AND APPEAL BOARD

3

REALTEK SEMICONDUCTOR CORP.,

Petitioner,

v.

ATI TECHNOLOGIES ULC,

Patent Owner.

4

CASE NO.:

5

IPR2023-00564

6

U.S. Patent No.

7

7,742,053

8

DEPOSITION OF MASSOUD PEDRAM, PH.D

9

VOLUME II

10

REMOETLY IN LOS ANGELES, CALIFORNIA

11

MONDAY, JANUARY 8, 2024

12

REPORTED BY: NATALIE PARVIZI-AZAD, CSR, RPR, RSR

13

JOB NO.: CSR NO. 14125

14

519722

174

1

UNITED STATES PATENT AND TRADEMARK OFFICE

2

BEFORE THE PATENT TRIAL AND APPEAL BOARD

3

REALTEK SEMICONDUCTOR CORP.,

Petitioner,

v.

ATI TECHNOLOGIES ULC,

Patent Owner.

4

CASE NO.:

5

IPR2023-00564

6

U.S. Patent No.

7

7,742,053

8

DEPOSITION OF MASSOUD PEDRAM, PH.D, VOLUME II

9

TAKEN ON BEHALF OF THE PATENT OWNER

10

REMOETLY VIA ZOOM VIDEOCONFERENCING, IN

11

LOS ANGELES, CALIFORNIA, BEGINNING AT

12

9:04 A.M. AND ENDING AT 3:03 P.M., ON

13

MONDAY, JANUARY 8, 2024, BEFORE

14

NATALIE PARVIZI-AZAD, CERTIFIED SHORTHAND

15

REPORTER NUMBER 14125.

175

1

A P P E A R A N C E S

2

FOR THE PETITIONER, REALTEK SEMICONDUCTOR CORP.:

3

ORRICK, HERRINGTON & SUTCLIFFE, LLP

4

BY: JEFFREY JOHNSON, ESQ.

5

609 MAIN STREET

6

40TH FLOOR

7

HOUSTON, TEXAS 77002

8

(713) 658-6400

9

3J6PTABDOCKET@ORRICK.COM;

10

REALTEK-AMD_OHS@ORRICK.COM

11

FOR THE PATENT OWNER, ATI TECHNOLOGIES ULC:

12

MINTZ, LEVIN, COHN, FERRIS, GLOVSKY AND

13

POPEO, P.C.

14

BY: REZA DOKHANCHY, ESQ.

15

3580 CARMEL MOUNTAIN ROAD

16

SUITE 300

17

SAN DIEGO, CALIFORNIA 92130

18

(858) 314-1596

19

RDOKHANCHY@MINTZ.COM

20

ALSO PRESENT:

21

JERRY BUTLER, VIDEOGRAPHER;

22

KAREN MORRISON, DEPOSITION TECHNICIAN;

23

STEPHANIE STUTZMAN, PARALEGAL WITH MINTZ,

24

LEVIN

176

1

I N D E X

2

WITNESS

3

MASSOUD PEDRAM, PH.D

4

EXAMINATION BY MR. DOKHANCHY

5

EXAMINATION BY MR. JOHNSON

6

E X H I B I T S

7

EXHIBIT NO.

8

DESCRIPTION

9

PAGE

10

EXHIBIT 1

11

DECLARATION OF DR. MASSOUD

12

PEDRAM IN SUPPORT OF

13

PETITION FOR INTER PARTES

14

REVIEW OF U.S. PATENT NO.

15

7,742,053

16

EXHIBIT 2

17

ORDER NO. 31: CONSTRUING

18

TERMS OF THE ASSERTED CLAIMS

19

(OCTOBER 18, 2022)

20

EXHIBIT 3

21

INITIAL DETERMINATION ON

22

VIOLATION OF SECTION 337

23

AND RECOMMENDED

24

DETERMINATION ON REMEDY

25

AND BOND, ADMINISTRATIVE

26

LAW JUDGE CAMERON ELLIOT,

27

(JULY 7, 2023)

28

EXHIBIT 4

29

UNITED STATES PATENT NO.:

30

7,742,053 B2

31

EXHIBIT 5

32

STUTTARD REFERENCE

33

EXHIBIT 6

34

WILLIAMS REFERENCE

35

PAGE

36

178

37

320

38

187

39

188

40

193

41

206

42

222

43

269

Transcript of Massoud Pedram, Ph.D., Volume II
Conducted on January 8, 2024

2 (177 to 180)

177 1 REMOTELY IN LOS ANGELES, CALIFORNIA 2 MONDAY, JANUARY 8, 2024, 9:04 A.M. 3 4 THE VIDEOGRAPHER: Here begins media 5 number 1 in the videotaped deposition of Dr. Massoud 6 Pedram in the matter of "Realtek Semiconductor 7 Corporation v. ATI Technologies ULC" in the United 8 States Patent and Trademark Office, case number 9 IPR2023-00564. Today's date is January 8, 2024, and 10 the time on the video monitor is 9:04 a.m. Pacific 11 Standard Time. 12 The remote videographer today is Jerry 13 Butler representing Planet Depos. All parties of 14 this video deposition are attending remotely. 15 Would counsel please voice identify 16 themselves and state whom they represent starting 17 with the taking attorney. 18 MR. DOKHANCHY: Reza Dokhanchy from Mintz, 19 Levin on behalf of the patent owner, ATI. 20 MR. JOHNSON: Jeffrey Johnson from Orrick, 21 Herrington & Sutcliffe on behalf of the Realtek 22 entities and the witness. 23 THE VIDEOGRAPHER: Thank you. 24 The court reporter today is Natalie 25 Parvizi, also representing Planet Depos.	179 1 BY MR. DOKHANCHY: 2 Q. Okay. Dr. Pedram, I hope you're feeling a 3 little bit better. 4 A. I do. I still have a little bit of 5 coughing and feel cold, but otherwise, I'm okay. 6 Q. Okay. You understand that you're here to 7 testify today regarding case number IPR2023-0564 8 regarding US patent number 7,752,053; is that 9 correct? 10 A. Yes, sir. 11 Q. Is there any reason why you cannot give 12 truthful and accurate testimony today? 13 A. No, there's not. 14 Q. Okay. I want to talk about your 15 background a little bit. 16 Do you have any experience designing 17 graphics processing systems? 18 A. I've done many different research projects 19 over the years. 20 I remember back in early 2000s, I had a 21 big army contract -- US Army contract in which we 22 designed the so-called "Land Warrior" system. The 23 Land Warrior system that we were designing back then 24 included a head-mounted display in which a soldier 25 could overlay graphics, objects over the real scene
178 1 The witness will now be sworn. 2 THE CERTIFIED STENOGRAPHER: Please raise 3 your right hand to be sworn. 4 5 MASSOUD PEDRAM, Ph.D., 6 having declared under penalty of perjury to tell 7 the truth, was examined and testified as follows: 8 9 MR. DOKHANCHY: So before we get started, 10 I just want to make a quick note that I previewed 11 off the record that during last Friday's deposition, 12 Exhibit 7, the ITC initial determination, I stated 13 that was public. That version is not public. It 14 has Respondent's confidential information redacted 15 but not AMD's. So it does have AMD's confidential 16 information in that Exhibit 7, Friday's deposition. 17 So I ask that -- you know, Dr. Pedram, if 18 you're not on the protective orders for that case, 19 please delete any copies you may have. We didn't 20 ask anything confidential or discuss anything 21 confidential during that exhibit, so we don't need 22 to mark that transcript confidential. So just a 23 note about the exhibit itself. 24 25 EXAMINATION	180 1 and then make determination that's what the best 2 action is. And my research group -- I was the PI, 3 the principle investigator, on that project 4 involving some other faculty as well. But my team 5 was very much involved in the design of all the 6 components and the assembly into a head-mounted 7 display. 8 I remember, in fact, meeting with some 9 two-star generals at the time. And when the case 10 was field-tested in -- I think it was 2003 or '04, 11 they were quite happy about what we had done in 12 putting together a graphics system that would allow 13 this kind of functionality for the "soldier of the 14 21st century," they called it. 15 That's one, I guess, initial involvement 16 in graphics processing system designs. 17 We also worked on doing automatic 18 adjustment of the backlights in various kinds of 19 display systems. And when you do backlight 20 adjustment, of course, there's going to be a quality 21 degradation on the image, and you have to do image 22 processing contrast enhancement techniques to try to 23 recover the backlight intensity that's lost. We 24 wrote a number of papers on that. We got a patent 25 on it and so on so forth. This was 2005, '06, '07,

Conducted on January 8, 2024

181 1 '08 time frame. 2 I also have worked on various kinds of 3 parallel processing systems in terms of their 4 architecture and their optimizations to maximize the 5 throughput and so on, and I had a visiting student 6 in my group for about one and a half year whose 7 focus was entirely on GPU design, and we worked 8 extensively, at that time, on GPU architectures and 9 then the GPU design. 10 (Reporter clarification.) 11 THE WITNESS: Graphics processing system 12 perhaps. 13 BY MR. DOKHANCHY: 14 Q. To clarify that last point, I think you 15 did start out saying you have worked in various 16 "power processing systems" and, later, you described 17 GPU work. 18 A. Well, parallel processing systems. I'm 19 sorry. Parallel processing systems. 20 Q. Got it. Okay. Okay. 21 So on that army-related contract, I 22 believe you're saying the systems that you worked on 23 included a display component; is that right? 24 A. It was not just display. It was 25 overlaying graphics objects on tops -- on top of the	183 1 have, I think, some components of this in my lab, 2 but we -- we did it on an LCD. Later on, in 2010, 3 we switched to OLED. But at the time, it was an 4 LCD, backlit LCD, in the lab, but then, in the 5 field, it was a type of transparent thing that was 6 used. 7 Q. And what was your role in connection with 8 that project? 9 A. Oh, I was the principle investigator. I 10 was the main investigator. We had a team of maybe 11 ten postgraduate students and post-docs involved and 12 some other faculty, and I would visit the army labs 13 in -- I remember it was in -- somewhere in New 14 Jersey. We would visit there very frequently, 15 almost two, three, four times a year to showcase 16 what we had done and what we have developed and that 17 kind of thing. 18 Q. Were you responsible for designing any of 19 the subsystems of that project? 20 A. Oh yes, of course. I was the principle 21 investigator; right? So we designed multiple 22 different things at that time, including components 23 and -- yeah. 24 Q. Which subsystems did you personally 25 design?
182 1 image that a soldier in the field sees through the 2 head-mounted display. So it was a combination of 3 actual viewing by the soldier under different 4 lighting conditions; nighttime, daytime. I remember 5 we got a special infrared camera that, sort of, 6 would allow the soldier to see night and then look 7 at what we can do during nighttime. So it was a 8 substantial project, and it's well documented. 9 Again, it was really field-tested, and 10 later on, I heard that it was used actually in the 11 field in Iraq in 2006 or so time frame. So that was 12 a major success story from my research team. 13 Q. So that display. Was that on a screen or 14 on somewhere else? 15 A. It's a head-mounted display. So there's a 16 sort of a screen, but it's not your desktop screen. 17 It is head -- head-mounted; right? So there's a 18 smaller screen there. But it's still -- you can 19 think of it as a screen. 20 Q. Was that a sort of LCD screen or some sort 21 of -- was it more like an augmented reality, where 22 the screen was clear? 23 A. It was -- I think we built different types 24 of systems experimentally in the lab and in the 25 field. In the lab, when we did it -- and I still	184 1 A. So this all related to, again, display 2 systems; right? Head-mounted display systems, at 3 the time, but also processing the information that 4 we receive to overlay objects -- the information 5 that we have. So for example, for range-finding for 6 identifying the targets, for deciding what the 7 potential firing angle should be. Stuff like that. 8 So a lot of processing is needed in the 9 system like this. 10 Q. Have you ever designed a GPU? 11 A. I've not designed a GPU from scratch. I'm 12 quite familiar with the architecture of the GPUs 13 and -- but I've not designed from scratch a GPU, 14 although I could do it if I -- I'm asked to do it. 15 I mean, we normally do things depending on 16 the funding that we have. I've certainly designed 17 full-blown multiprocessor systems, but graphics 18 processing -- well, I want to add a little bit of 19 clarification here. 20 I'm currently doing a project with Defense 21 Advanced Research Projects Agency, DARPA. The 22 purpose of this is to build a unified shader 23 architecture for signal -- advanced signal 24 processing. It's not graphics processing as such 25 but signal processing; radar antenna signal

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 98 of 182

Conducted on January 8, 2024

185 1 processing, beam-forming. Stuff like that. And as 2 part of that, we are developing the architecture of 3 such a system from ground up. 4 I recently also concluded a project for 5 DARPA. Again, DARPA is Defense Advanced Research 6 Projects Agency. That lasted for about three years, 7 in which one collaboration with some of the faculty 8 at USC and other places. We built a highly parallel 9 processing engine comprising of many, many different 10 parallel lengths (phonetic) doing computations at 11 the same time to enable so-called "fully homomorphic 12 computation." 13 This is computation that works on 14 encrypted data. It's not directly graphics 15 processing but the concepts are very similar to what 16 you would do in graphics processing. Different 17 types of threads come in. You have to process them 18 in very high throughput in parallel. So there's lot 19 of similarity in architecture and performance 20 considerations of such systems to graphics 21 processing systems. 22 My team is a design team; right? So over 23 the years, we have designed many, many different 24 systems. 25 Q. Have you designed any systems or --	187 1 MR. DOKHANCHY: All right. Let's go ahead 2 and have introduced, as Exhibit 1, document 22, 3 please. 4 (Exhibit 1 marked.) 5 DEPOSITION TECHNICIAN: Okay. One moment. 6 THE WITNESS: Again, Mr. Dokhanchy, I do 7 have all the documents also on my desktop. Nothing 8 is marked or highlighted in any way. It's just PDF 9 that I'm going to refer to if -- unless it's a new 10 document, if that's okay with you. 11 BY MR. DOKHANCHY: 12 Q. Sure. What documents do you have up on 13 your screen? 14 A. So currently, I have AMD response, my -- 15 my declaration, decision granting institution, the 16 '053, Stuttard, Williams, and Whittaker. 17 Q. The AMD response. Is that the patent 18 owner preliminary response from this proceeding? 19 A. It's -- correct, yes. Absolutely right, 20 yes. 21 Q. Okay. And all of them are unmarked, clean 22 copies; is that correct? 23 A. Yes, sir. 24 Q. Okay. Then, that's fine. 25 Up on the screen right now is Exhibit 1.
186 1 withdrawn. 2 Have you been personally responsible for 3 the design aspects of any systems or subsystems 4 regarding how they process graphics commands? 5 A. Yeah. Of course, again, as part of the 6 Land Warrior project I mentioned, a lot of graphics 7 processing was done. I mean, so definitely there, 8 we did a lot. 9 Furthermore, when we, later on, looked at 10 dynamic backlight scaling technology and the patent 11 that we received for it and so on, we have to do a 12 lot of graphics processing to recover the quality of 13 the signal that is lost due to backlight escaping 14 itself through image processing techniques. 15 Furthermore, since at least 2019, 2018 16 time frame, I've been very heavily involved in using 17 GPUs for doing all kinds of image classification, 18 object tracking, object recognition through images, 19 through vision transformers, pollution on neural 20 networks, and so on. 21 So I'm actually quite -- my team included, 22 of course -- quite good at working with GPUs and 23 GPGPUs, not only to do graphic processing tasks but 24 also to do general purpose processing tasks, such as 25 machine learning and AI.	188 1 Dr. Pedram, is that the declaration of yours from 2 this proceeding regarding the '053 patent? 3 A. Yes, sir, it is. 4 Q. Okay. I'll take us down to paragraph 42. 5 And starting at paragraph 42, you give 6 your claim constriction opinions; is that correct? 7 A. Yes. 8 Q. And you say, in paragraph 42, "I 9 understand that several of the claim terms have 10 already been construed by the ITC and the Board in 11 decisions regarding the '053 patent. For the 12 purposes of this declaration, I have applied the 13 narrower of the available constructions"; is that 14 correct? 15 A. Yes, sir. 16 Q. All right. Let's introduce, as Exhibit 2, 17 document 29, please. 18 (Exhibit 2 marked.) 19 BY MR. DOKHANCHY: 20 Q. And when you say -- before we get to that, 21 when you say, in your declaration "construed by the 22 ITC," you're referring to the 1318 investigation 23 between AMD and various respondents, including 24 Realtek; is that right? 25 A. I believe so. This is the one that Judge

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 99 of 182

Conducted on January 8, 2024

189 1 Elliot presided at the ITC. 2 Q. All right. Okay. Let's get up Exhibit 2. 3 And Exhibit 2 is the order regarding claim 4 construction in that ITC matter; is that right? 5 A. Yes, sir. 6 Q. And if we go down, starting on page 15, 7 you see that the ALJ is addressing several terms, 8 one of them being, in the '053 patent, the term 9 "select a command thread from either of the 10 plurality of pixel command threads and the plurality 11 of vertex command threads." 12 Do you see that? 13 A. Yes, sir. 14 Q. Okay. And on the next page, 16, towards 15 the bottom, you see here, the ALJ held, "But the 16 claim itself requires selection of" -- quote -- "a 17 single command thread" -- end quote -- "from at 18 least four such threads, two or more pixel threads 19 and two or more vertex threads. So the selection 20 must be based, at least in part, on considerations 21 other than merely the 'command thread type.'" 22 Do you see that? 23 A. Yes. 24 Q. Okay. Did you apply that meaning of the 25 claim term in your declaration?	191 1 as the case arises: I disagree with that point of 2 view, but I believe that the prior art references 3 would invalidate -- render obvious the assertive 4 claims of the '053 if, regardless of the particular 5 construction that's being discussed here, whether 6 it's one out of two threads or one out of four 7 threads, two of each type, I believe under either of 8 these, the prior art would invalidate '053. 9 Q. Okay. Not the question I asked, but I 10 appreciate that answer. 11 My question is simply: When you were 12 determining the meaning of the claim terms, did you 13 include the requirement stated here, at the bottom 14 of page 16, that the claim itself requires selection 15 of a single command thread from at least four such 16 threads, two or more pixel threads and two or more 17 vertex threads? 18 MR. JOHNSON: Objection. Form. 19 A. You have to show me the specific part of 20 my report. I don't remember everything that I put 21 in my record, in every paragraph. So if you point 22 to me -- to a particular part of my report that 23 relates to this and say, "You made this statement. 24 Is it also consistent with the view in which a 25 command thread must be selected from among four, two
190 1 A. This is not a construction. The 2 construction is above -- if you go up, with what was 3 construed is there, this is just Judge Elliot 4 providing further, I guess, justification for his 5 construction above. 6 I applied the construction that was handed 7 down to us, which meant that you select one command 8 thread from one or more -- not necessarily one, 9 even, but a command thread from among a plurality 10 command -- plurality of vertex commands threads or a 11 plurality of the pixel command threads. That's the 12 construction I apply. 13 Q. So is the answer to my question no? 14 A. I do not agree. I'm sorry. I do not -- I 15 respectfully, to Judge's hint here, I disagree that 16 the language of the claim or the -- in view of the 17 specs and its prosecution history actually requires 18 that a command thread be necessarily chosen from 19 among four threads, two of each type. I disagree 20 with that. I -- I don't think that's the right 21 construction. 22 Q. Okay. So you did not apply the meaning of 23 the claim term that I read; correct? 24 MR. JOHNSON: Objection. Form. 25 A. So my -- my opinion, and we can discuss it	192 1 of each type?" then I can respond. But you're 2 asking a very general question. 3 I can tell you yes, I was aware of this 4 hint here, and I did consider it as I was writing my 5 report and my declaration. 6 Q. So my question is a little bit different. 7 You have a section on claim construction 8 in your report where you say what you understand the 9 claim terms to mean and you say what constructions 10 you have applied and you say that you applied the 11 narrowest construction put forth by the ITC report. 12 So my question is just simply: In your 13 construction of the -- selecting a command thread 14 term, did you include the requirement stated by the 15 ALJ at the bottom of page 16 of the claim 16 construction? It's a yes-or-no question. 17 MR. JOHNSON: Objection. Form. 18 A. Yeah. With due respect, to me, this is 19 not yes-or-no question. I cannot provide a 20 yes-or-no response. 21 My response here is that I went by the 22 narrower of the claim constructions. And if you go 23 up, in the same table that you're showing me, there 24 is a construction for that phase, and that 25 construction does not include this requirement.

Conducted on January 8, 2024

193 1 This requirement is provided in the justification 2 section for the construction itself, but this is not 3 the construction. 4 So I applied that construction, but I'm 5 aware of this further requirement here. And when 6 appropriate, I've discussed it, and I will be glad 7 to discuss it today when appropriate. But that's 8 the response I could provide. I understand it's not 9 yes or no, but that's the best I can do, sir. 10 (Exhibit 3 marked.) 11 MR. DOKHANCHY: All right. Let's mark, as 12 Exhibit 3, document 33, please. 13 BY MR. DOKHANCHY: 14 Q. And document 33 is the initial 15 determination in the 1318 ITC investigation; is that 16 correct, Dr. Pedram? 17 A. Yes, it is. 18 Q. And if we go to page 37 -- starting on 19 page 32, the ALJ gives some further claim 20 construction opinions in the ID; correct? 21 A. Yes, sir. 22 Q. And -- or rulings, rather, than opinions. 23 On page 37, he states, "Thus" -- and this 24 is about a third of the way down the page: "Thus, 25 the intrinsic evidence strongly supports finding 'at	195 1 know that Realtek and other respondents in the ITC 2 actually argued for this interpretation of the '053 3 claims; correct? 4 MR. JOHNSON: Objection. Form. 5 A. I'm sorry. I believe that's what they 6 did. But again, this is quite common, as I 7 understand and as I've seen in my interactions and 8 litigation experience that I've had, that lawyers 9 tend to make arguments in the alternative, 10 especially with respect to noninfringement and 11 invalidity. And so, it is something that's happened 12 routinely. 13 But regardless of that practice that I've 14 seen, it's always been my opinion -- and you can 15 check all of my deposition testimonies, all of my 16 trial testimony, everything -- that I do not agree 17 with this perspective and this point of view, 18 defining the complete storage of a thread as 19 being -- as a required element of any claims of the 20 '053. 21 Q. Okay. And so, Realtek argued for this 22 construction. The ALJ found it to be correct. 23 And sitting here today, you disagree with 24 that; is that fair? 25 A. I disagreed with it even back then. I
194 1 least one memory device comprising a first portion 2 operative to store a plurality of pixel command 3 threads and a second portion operative to store a 4 plurality of vertex command threads' means storing 5 complete command threads at any given point in time, 6 and not partial command threads." 7 Do you see that? 8 Is that a yes? 9 A. Yes, sir, I do. I do see it, yes. 10 Q. Do you agree with the ALJ's statement? 11 A. It has never been my opinion that storing 12 complete command threads at any given moment in time 13 is a requirement of the claim language. I've always 14 opined that although it is quite possible that the 15 memory contains complete threads at different points 16 in time, it is not a requirement of the claim to 17 have that, and a partial command thread storage in 18 memory would suffice. 19 So I respectfully disagree the finding of 20 Judge Elliot in this case. And I know some of this 21 was based on noninfringement positions and so on 22 with the arguments in the alternative, but as far as 23 I'm concerned, I've never agreed to this meaning of 24 the -- of the storage of a thread in '053 patent. 25 Q. And I think you alluded to this, but you	196 1 disagreed with it in my opening report. There, I 2 disagreed with it. In my deposition testimony, I've 3 always disagreed with this. I've been consistent on 4 this point, sir. 5 Q. Okay. Okay. The -- claim 1 and claim 5 6 of the '053 patent require that the arbiter select a 7 command thread; is that right? 8 A. Yes, sir. 9 Q. Does the term "select" require a choice to 10 be made by the arbiter? 11 MR. JOHNSON: Objection. Form. 12 A. The question is vague. I'm not sure what 13 do you mean by "a choice to be made." A choice in 14 respect to what? So maybe you can provide a little 15 bit more detail or context. 16 Q. Sure. So in the phrase "select a command 17 thread," which is the action to be performed by the 18 arbiter, does the arbiter need to make a choice 19 about the command thread in the process for 20 selecting it? 21 MR. JOHNSON: Objection. Form. 22 A. Sorry. I don't understand. "A choice 23 about the command thread." What does that mean? 24 An arbiter selecting a command thread 25 means that it looks at command threads that are

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 101 of 182

Conducted on January 8, 2024

197 1 available and chooses one, selects one. That's the 2 meaning a person skilled in the art would understand 3 the function of the arbiter to be in this context, 4 also consistent with the '053 specs. 5 Q. If something merely receives data without 6 taking a role, or "choosing," as you said, one of 7 them, did that thing select the data that it 8 received? 9 MR. JOHNSON: Objection. Form. 10 A. Again, not a very precise question for me 11 to respond. 12 I can give you an example that if an 13 arbiter receives a number of command threads and 14 chooses one from among them based on priority or 15 resource availability or what have you, it's made a 16 selection. If an arbiter takes -- receives these 17 command threads and what it does is that it chooses 18 one and says, "You go on bus 1," and she chooses 19 another one and says, "You go on bus 2," that's a 20 selection. The arbiter is doing its function of 21 selecting and assigning to different buses these 22 command threads. 23 If a realtime command thread comes that 24 has higher priority than a background thread, the 25 arbiter selecting the realtime thread and sending it	199 1 not have a fixed answer one way or the other. 2 There's a possibility that -- and we have 3 seen it also in '053 -- that something comes through 4 a FIFO, goes into another FIFO, gets back inserted 5 into something or the initial FIFO, and so on. 6 So there could be interactions between 7 FIFOs as -- as these things are flowing through 8 them, and again, there are selections taking place 9 at that point. 10 Q. You added a lot of detail that was not in 11 my question. Please stick to my question. I 12 specifically said "with no further details," so let 13 me ask you the question again. 14 Data comes into a FIFO data, leaves the 15 FIFO in the order that it came in, and arrives as a 16 processing block. Without knowing anything further, 17 at that point, has the processing block selected the 18 data that it received? 19 MR. JOHNSON: Objection. Form. 20 A. Again, this question doesn't have them in 21 the right language consistent with the '053 because 22 the arbiter function in the '053 is not met by its 23 processing blocks. So you are talking about a 24 processing block receiving tasks or threads to 25 perform and then supposedly the processing block
198 1 forward is a selection. 2 All kinds of selections would be done 3 based on different criteria, including priorities or 4 resource availability, and the arbiter is performing 5 its functions in that way. 6 Q. So let's take: Data comes into a FIFO and 7 data leaves the FIFO as it came in and arrives at a 8 processing block. Did the processing block in that 9 scenario, with only those data details that I've 10 given you, select the data that it receives? 11 MR. JOHNSON: Objection. Form. 12 A. There's not enough information to respond 13 to this question because when something comes into a 14 FIFO, it may be held, for example, in the FIFO for 15 some time for any number of reasons. So although 16 the order in which it goes out is the same as the 17 order in which it came in -- apparently, in your 18 question, that's the case -- then because still 19 there's an intentional delay in processing of the 20 FIFO entries, there could be a selection taking 21 place. It is possible. 22 But again, really, there should be 23 details. I don't know what details are, and there 24 could be any number of situations in which this 25 seemingly simple question would get complicated and	200 1 itself is selecting data. This is nothing that I 2 found in the '053 or any of the prior art, so it's 3 completely out of context. 4 For me at least, you to have a notion of 5 an arbiter and say, "But this arbiter doesn't do 6 anything." And whatever you want to say, that's 7 fine with me, but I -- I just haven't thought about 8 a situation in which supposedly the receiver of the 9 thread itself when it has received it had something 10 to do with this reception. 11 Q. I can clarify. I was just using 12 "processing block" in a generic sense, not in the 13 sense of the Stuttard reference or anything else. 14 So let me -- I can use the word "arbiter" if you 15 want. I just meant a block of logic. So I'll ask 16 the question that way. 17 A. Please do. 18 Q. Okay. So in the scenario in which data 19 comes into a FIFO and then data leaves the FIFO in 20 the order that it came in and arrives at a logic 21 block and nothing further has happened, at that 22 point this time, has the logic block selected the 23 data that it received? 24 MR. JOHNSON: Objection. Form. 25 A. I'm sorry. You have not changed your

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 102 of 182

Conducted on January 8, 2024

201 1 question. You just replaced the term "processing 2 block" with the "logic block." My -- 3 Q. That was your problem with my question, so 4 that's -- 5 A. No, sir, that's not my -- respectfully, 6 that was not my problem with the question. My 7 problem with the question was: There are two 8 different entities here. One is a processing block, 9 or that you like to call the "logic block," and one 10 is an arbiter. You have to -- in your question, you 11 have to have both elements present. There's an 12 arbiter element. There's a processing block 13 element. And then, you pose a question in view of 14 FIFO and whatever else you have in mind, but you 15 have to have those two elements because that's the 16 context you're talking about. That's the context of 17 '053 and that's language of the claim with the 18 specification. 19 Q. Dr. Pedram, I'm talking about the word 20 "select." I don't have to use the word "arbiter" if 21 I don't want to. I'm asking about your 22 understanding of the word "select." If you can't 23 answer my question, say you can't answer my 24 question, but I don't need to use the word "arbiter" 25 in my question if I don't want to. I'm talking	203 1 make no other assumptions. In fact, it's possible 2 that information stays in the FIFO for some reason; 3 some period of time, it's delayed. That is a 4 selection happening there: Somebody causing that 5 FIFO or not to be processed and a -- a -- a thread 6 to be kept there for some reason before they are 7 handed downstream. 8 But you're saying don't make no such 9 assumption, so therefore, I'm going to make an 10 assumption that the data comes from one end of the 11 FIFO without any delay. Nobody causing anything to 12 stay in the FIFO. It just goes through the FIFO, 13 one tick of the clock at a time. So if the data 14 entered in the FIFO of depth 4, four clock cycles 15 later, that data or thread comes out the other end 16 of the FIFO and then that goes to an arbiter that's 17 there and, supposedly, arbiter has nothing do with 18 the processed flow into the FIFO itself. 19 Who put the data into the -- or the thread 20 into the FIFO? When they were put in? How long 21 they stayed there, if anything? None of it has been 22 done with the knowledge or involvement of that 23 so-called "arbiter." And then, you're saying, at 24 that point this time, did arbiter make a selection 25 about what got to the arbiter?
202 1 about the word "select." So just tell me you can't 2 answer my question; otherwise, please answer it. 3 I'll ask it once again, if you want me to, or you 4 can just go ahead and answer it. 5 A. I cannot answer the question as asked. 6 Q. Okay. Now let's try with the word 7 "arbiter." 8 In the scenario in which data comes into a 9 FIFO and then leaves the FIFO and arrives at an 10 arbiter, at that point in time, has the arbiter 11 selected the data that it received? 12 MR. JOHNSON: Objection. Form. 13 A. These are not well-posed questions. I 14 understand you can ask anything you would like to 15 ask, but generally, an arbiter looks at the tasks in 16 some medium, such as maybe a memory or set of 17 buffers or what have you, and then makes a selection 18 of one of these tasks or threads and passes them on 19 to processing blocks or causes them to be 20 transferred to appropriate processing block. An 21 arbiter may not itself receive the data. It's just 22 looking at the data and the threads, I mean. 23 And so, the question you're asking me now 24 is that the FIFO is in front of the arbiter. The 25 data goes through a FIFO channel, and you're saying	204 1 Under these restrictive set of assumptions 2 you made, that arbiter essentially had nothing do 3 with what arrived at its inputs, then yes, arbiter 4 has nothing to do with what arrived at its inputs. 5 The question itself provides the answer. 6 So I'm not saying anything about the 7 arbiter as such or the function of the arbiter. I'm 8 just saying, in the hypothetical that you presented 9 to me, the way you presented it to me, and the way I 10 understand it, the so-called "arbiter" in that 11 question has nothing to with the inputs coming to it 12 because that's the way you presented it. So fine. 13 Yes, I understand the question, and fine, I 14 acknowledge it. In that case, that's the situation. 15 Q. And in that situation, the arbiter did not 16 collect the data; correct? 17 A. I'm not sure why such an -- why such an 18 entity would be called an "arbiter" in that case. 19 Q. Dr. Pedram, just answer the question. 20 MR. JOHNSON: I had an objection to form. 21 A. Because you used the term "arbiter," that, 22 to a person skilled in the art and in view of the 23 '053 context, has very specific meaning, and that 24 meaning is inconsistent, incompatible with the 25 example you provide, yet you insist on that example,

Conducted on January 8, 2024

205 1 which is, in the context of '053, incorrect use of 2 the term and the meaning associated to the term per 3 constructions done by the ALJ or the PTAB, per a 4 person of skill in the art's understanding of what 5 the term means. 6 So out of context, with incorrect claim 7 constructions applied, then I would say yes, the 8 so-called "arbiter" that you define has nothing to 9 do with what arrived at its inputs. But that's not 10 the -- that's not what a person skilled in the art 11 looking at '053 would understand the function of the 12 arbiter to be. 13 Q. And if that so-called "arbiter" had 14 nothing do with that arrived at its inputs, it did 15 not select the data; correct? 16 MR. JOHNSON: Objection. Form. 17 A. If the so-called "arbiter" had nothing do 18 with what inputs arrive at its input terminals, 19 then, of course, it had no selection of choice at 20 that point. 21 Q. Does the selection of commands, command 22 threads in the '053 patent have to be from the 23 threads that are in the claimed memory? 24 MR. JOHNSON: Objection. Form. 25 A. Let's look at the '053 claim language.	207 1 thought about this. I need more time to consider 2 this question in view of the specs that we have. 3 What the language of the claim itself 4 states is that there's a memory device that stores 5 the different command threads -- pixel and vertex 6 command threads -- in different portions of itself, 7 and that there's an arbiter that's coupled to the 8 memory twice that can select one or the other of 9 these command threads based on relative priorities. 10 There's no explicit disclosure that the 11 arbiter is required to make a selection at new 12 particular point in time; i.e., for example, when 13 the threads are still residing in the memory device. 14 It may or may not, and I don't know. I have to look 15 at this carefully and -- and think about it to be 16 able to respond one way or the other. At this 17 point, I don't know. 18 Q. Do you have any particular understanding 19 as to whether that is required when rendering your 20 opinions in your declaration? 21 A. I do not recall in any of the reports I've 22 seen from AMD experts in the ITC case or in the AMD 23 declaration for these proceedings that this point 24 was raised and I did not respond to it and I did not 25 think about it, therefore.
206 1 Maybe we can focus on -- 2 MR. DOKHANCHY: That's fine. Let's 3 introduce, as Exhibit 4, document 23. 4 (Exhibit 4 marked.) 5 BY MR. DOKHANCHY: 6 Q. Okay. Document [sic] 4 is the '053 patent 7 issued in this case; is that correct, Dr. Pedram? 8 A. Yes. 9 Q. All right. Let's scroll down to claim 1. 10 As to the element in which the arbiter 11 must be operable to select the comment thread from 12 either the plurality of pixel of command threads and 13 the plurality of vertex command threads, does 14 selection need be made from command threads that are 15 in the memory device, which are described in the 16 prior element as "storing command threads"? 17 A. I've not thought about this question 18 before. This is a fine point you're raising about 19 when the arbiter is making a selection between the 20 command threads. Do the command threads have to 21 reside in the memory that stored them in the first 22 place or that the selection, which could happen 23 downstream in some bodement when the data is in 24 flight out of the memory element itself? 25 I don't -- so again, I really haven't	208 1 Q. Is it your view that if you didn't see an 2 argument raised by AMD experts in the ITC case, it's 3 not something that you needed to opine on in your 4 declaration? 5 A. No, no. Sir, I didn't mean it in that 6 way. Of course, with respect to noninfringement, 7 that would have been my position, but with respect 8 to invalidity, that's not what I'm saying. 9 What I'm saying is: Because you never 10 raised it and I never found it necessary to consider 11 this distinction, my review of the '053 specs in 12 view of the claims being asserted, the explicit 13 express language of the claims, as understood by a 14 person skilled in the art and therefore, at least 15 myself, and in any of the prior art references that 16 I've analyzed, this question never came to my mind. 17 So I didn't find it a material question to 18 think about and respond to. Now you're raising it, 19 and I -- I can think about it and get back to you. 20 Q. Why is it that you didn't think it was 21 necessary to determine if there's a requirement 22 regarding from where the arbiter selects commands in 23 identifying the prior art disclosures that 24 invalidate the claims? 25 A. Because the claims should be interpreted

Conducted on January 8, 2024

209 1 in the broadest possible term; right? In the claim 2 language itself, claim 105, they don't have this 3 somewhere in it; right? So they're just talking 4 about -- these are Aratus (phonetic) claims. 5 We are talking about graphics processing 6 system, having a memory device that must satisfy 7 certain requirements and an arbiter that must 8 satisfy certain other requirements, and one of those 9 other requirements is that it's coupled to the 10 memory device. 11 There's no statement in that claim or case 12 that, in fact, the arbiter must do the selection 13 while the command threads are still residing in the 14 memory device. Or maybe not; they are not residing 15 in memory device. They are in flight or transfer 16 somewhere else. So I wouldn't narrow down the claim 17 of -- the scope of a claim in an unnecessary way. 18 Again, that requirement must be there or might not. 19 I have to think about it. 20 But what I can say -- you're asking me why 21 I didn't do it in the first place: Because I did 22 not, so far, find it relevant to anything, any 23 analysis I've done to make that determination. 24 Q. And we see in claim 2 and in claim 5 the 25 concept of command processing engines.	211 1 MR. JOHNSON: Objection. Form. 2 A. My reading of the claim language is that 3 the arbiter is operable to select a command thread 4 for either of the plurality of pixel command threads 5 or the plurality of the vertex command thread. 6 It's -- this either means that this claim also reads 7 on a system in which the arbiter only selects one 8 type of command. Example: Selects a pixel command 9 thread all the way time and sends it to the command 10 processing engine that, of course, is capable of 11 processing the pixel command threads, and executes 12 it. 13 And so, there is no requirement in the 14 claim language itself that the command processing 15 engine be capable of simultaneously processing pixel 16 command thread and vertex command threads because 17 there's no requirement for the arbiter to 18 necessarily select a command -- pixel command thread 19 and then subsequently select a vertex command thread 20 and vice versa. None of that is required, and so I 21 do not agree. 22 Q. So you had a word in there that wasn't in 23 my question, and that's about simultaneous 24 processing, so I want to be clear about my question. 25 My question is regarding the capability to process
210 1 Do you see that? 2 A. Yes. The single command engine processing 3 engine implied -- I believe it's command processing, 4 the validity of command processing engines. 5 Q. You agree that a command processing engine 6 must be capable of processing both pixel command 7 threads and vertex command threads? 8 MR. JOHNSON: Objection. Form. 9 A. Not necessarily, no. My reading is not 10 that. 11 Q. All right. Let's go up to the abstract. 12 And you see in the abstract, the inventor 13 describes the invention, and states, in the last 14 sentence, "The selected command thread is provided 15 to a command processing engine capable of processing 16 pixel command thread and vertex command threads." 17 Do you see that? 18 Is that a yes? 19 A. Yes, I see it. Yes, sir. 20 Q. In light of that, do you think it is a 21 claim requirement that the pixel -- withdrawn. 22 Do you think it is a claim requirement 23 that the command processing engine is capable of 24 processing pixel command threads and vertex command 25 threads?	212 1 pixel command threads and vertex command threads as 2 we see here in the abstract. 3 So my question is: Do the claims of '053 4 patent, particularly claim 2 and claim 5, require 5 that the command processing engine be capable of 6 processing pixel command threads and vertex command 7 threads? 8 A. My answer to that is: No. 9 Q. So turning back to a point that you just 10 raised. 11 It is your view that if in operation a 12 system only ever selects pixel command threads -- 13 and when I say "a system," the arbiter in that 14 system -- and that arbiter never selects any vertex 15 command threads, that could infringe this claim; 16 true? 17 And as to the memory device, would a 18 device or could a device infringe if the memory 19 device only ever stores pixel command thread and the 20 memory device never stores vertex command thread? 21 MR. JOHNSON: Objection. Form. And I had 22 an objection -- form -- to the previous question as 23 well. 24 A. I do believe that the memory device must 25 be capable of storing a plurality of pixel command

Conducted on January 8, 2024

213 1 threads and a plurality of vertex command threads, 2 so that is a requirement. There's no either/or 3 there. 4 So the language of the claim there -- 5 claim element is very clear. It must comprise two 6 portions; one to store a plurality of pixel command 7 threads and the other to store a plurality of vertex 8 command threads. 9 Q. Okay. So your view is that there's a 10 requirement to store both types of command threads 11 but no requirement to select both types of command 12 threads; is that fair? 13 MR. JOHNSON: Objection. Form. 14 A. There is -- correct; there's no 15 requirement that the arbiter of the graphics 16 processing system of '053 will have to necessarily 17 select a pixel command thread at one time to be 18 followed by selection of a vertex command thread at 19 a different time. It could only select one type of 20 command threads, and that would read on the claim 21 language. 22 Q. Does the arbiter need to have from its 23 available command thread at any particular point in 24 time both pixel command threads and vertex command 25 threads from which to choose?	215 1 from a plurality of pixel command threads and is 2 never to select from a plurality of vertex command 3 threads, could the arbiter beat the claim 4 limitations? 5 MR. JOHNSON: Objection. Form. 6 A. The point is not about the ability of the 7 arbiter to select between the two pluralities of 8 commands but it is about the fact that it may choose 9 to select from only the pixel command threads one 10 application and then, in a later application -- in a 11 different application, it may select from the 12 plurality of the vertex command threads only. 13 I'm not making a statement about the fact 14 that arbiter can not do it. It can do it if it 15 chooses to, but it may not do it, and if there is a 16 system in which the arbiter of the system decides 17 that it wants to only choose from pixel command 18 threads, that system will still read on claim 1 and 19 5 of the '053. That is the statement I'm making. 20 Q. If the arbiter is only operable to select 21 from pixel commands and is not operable to select 22 from vertex command threads, could that arbiter meet 23 the claims? 24 A. No, it doesn't. 25 MR. JOHNSON: Objection. Form.
214 1 MR. JOHNSON: Objection. Form. 2 A. So there's no requirement in the claim 3 element related to memory device that the pixel 4 command threads and the vertex command threads must 5 be simultaneously present in the memory of the 6 device. 7 The only requirement here is that there's 8 a first portion of the memory that's operative to 9 store a plurality of pixel command threads and a 10 second portion operative to store a plurality of 11 vertex command threads. The two portions may not be 12 filled in at exactly the same instance in time. 13 That is not a requirement of the claim. 14 Q. So I don't think my question included 15 that, but let me phrase it a little bit differently. 16 Does the arbiter need to have the ability 17 to select from both a plurality of pixel command 18 threads and a plurality of command threads? 19 MR. JOHNSON: Objection. Form. 20 A. The arbiter must be operable to select a 21 command thread from either of the plurality of pixel 22 commands threads and the plurality of vertex command 23 threads based on relative priorities. That's the 24 only requirement. Nothing else. 25 Q. If the arbiter is only ever able to select	216 1 Q. So the arbiter needs to be operable to 2 select from both pixel command threads and vertex 3 command threads; correct? 4 MR. JOHNSON: Objection. Form. 5 A. Again, that's not what I'm saying. The 6 requirement is that it is operable to select a 7 command thread from either the plurality of pixel 8 command threads and the plurality of the pixel [sic] 9 command threads. There is no requirement here that 10 it is operable to select from both pluralities. 11 Either one or the other plurality is the 12 requirement. 13 Q. So if your view is that it is either one 14 or the other, then if the arbiter only selects -- or 15 is only operable to select from one and not the 16 other, can it meet the claims? 17 MR. JOHNSON: Objection. Form. 18 A. The other is not what I'm agreeing to. 19 All I'm saying, consistent with the claim language 20 and view of the specification that the system -- 21 that the graphics processing system that satisfies 22 all the other claim requirements and, in addition, 23 it has an arbiter, that this arbiter, which is 24 coupled to the memory device, is operable to select 25 a command thread from either plurality of pixel or

Conducted on January 8, 2024

217 1 vertex command threads, but it chooses to only pick 2 the pixel command threads. That's its choice. 3 You spoke of choice earlier; right? So 4 its choice to only pick pixel commands threads, for 5 whatever reason, right, and pass it on to the 6 command processing engine. That system will still 7 read on asserted claim 1. I could say on claim 1 of 8 the '053. 9 Q. I think you're talking about the 10 selecting, and I'm talking about the "operable to 11 select from" term. So that's, I think, the 12 confusion here. 13 My question is: If the arbiter is only 14 operable to select from one type of command thread, 15 can it meet the command limitations? 16 MR. JOHNSON: Objection. Form. 17 A. And my -- I'm going to probably repeat 18 what I said before. 19 My understanding is that -- that the -- 20 the arbiter of this claim must be operable to select 21 a command thread from either plurality of pixel or 22 vertex command threads, and that if it is operable 23 to select this command thread, it should be able to 24 recognize pixel versus vertex command threads and 25 then make a determination that I only want to go	219 1 command processing engines, of course, operable to 2 receive and process the command threads, but only 3 the received command thread. 4 And reception of the command thread by a 5 particular engine is determined -- dictated by the 6 smart arbiter sitting there that would assign the 7 right command type -- command thread to the engine 8 1, which is pixel command thread, and assign the 9 appropriate command thread, which is the vertex 10 command thread, to engine 2. In such a system, 11 engine 1 cannot do vertex processing. 12 Q. Sorry. Were you finished, Dr. Pedram? 13 A. No. I'm just saying -- what I'm trying to 14 say is: In that case, engine 1 is doing -- I 15 believe I said engine 1 is doing pixel processing. 16 So engine 1 cannot do vertex processing and vice 17 versa. Engine 2 cannot do pixel processing. 18 So there are two specialized engines. One 19 engine is for pixel processing. One engine is for 20 vertex processing. The arbiter decides which 21 thread, pixel or vertex, to send to which engine, 22 and there's no -- no problem there. And such a 23 system would read on claim 5. 24 Q. Is there any discussion in the '053 patent 25 specification regarding what you call "specialized
218 1 with pixel command threads because that's the 2 plurality I want to choose. 3 So this being operable to select a command 4 thread. It is -- applies to selecting from either 5 the plurality of pixel command threads or from the 6 plurality of vertex command threads, and that means 7 you're recognizing which one is the pixel command 8 thread and which one is a vertex command thread, and 9 you're choosing. 10 Q. And then, whichever type is chosen, in 11 claim 5, the command processing engines must each be 12 operable to receive and process whichever command 13 thread was chosen by the arbiter; correct? 14 A. That's not quite the correct reading of 15 this. The correct reading of the last element of 16 claim 5 is that there is a plurality of command 17 processing engines sitting there. The system -- 18 there are -- there's two command processing engines 19 in that plurality. The arbiter picks pixel command 20 thread and sends it -- sends it to command 21 processing engine 1, and it always does that. It 22 always send its pixel processing commands to command 23 processing engine number 1. And similarly, it 24 always sends the selected vertex command threads to 25 engine number 2. So now engine -- the plurality of	220 1 engines?" 2 A. Not that I recall. But the claim, it 3 should -- is what it is. When you look at the 4 claim, it's written in a particular way and it has a 5 particular scope and claim about what it covers. 6 Claim 5 that we just discussed simply discusses a 7 plurality of command processing engines, each 8 operable to receive and process the command thread. 9 But in a system in which the -- one of 10 these processing engines only receives a pixel 11 command thread, then it can be specialized to do 12 pixel processing, and another engine that only 13 receives vertex command threads can be specialized 14 to do vertex processing. And this is -- this system 15 will read on claim 5 of the '053, obviously, to read 16 on it, and there is no -- no discrepancy or no 17 issues there. 18 MR. DOKHANCHY: All right. Let's take a 19 ten-minute break. 20 THE VIDEOGRAPHER: We're going off the 21 record. The time on the video monitor is 10:26 a.m. 22 (Recess.) 23 THE VIDEOGRAPHER: We are back on the 24 record. The time on the video monitor is 10:43 a.m. 25 MR. DOKHANCHY: All right. Let's

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 107 of 182

Conducted on January 8, 2024

221 1 introduce, as Exhibit 5, document 25, please. 2 DEPOSITION TECHNICIAN: Counsel, when I 3 open this document, it says it contains unapplied 4 redactions. In order to save it, it needs to 5 permanently remove that content. 6 MR. DOKHANCHY: That's Exhibit 25? 7 DEPOSITION TECHNICIAN: Document 25? Yes. 8 MR. DOKHANCHY: Sorry. Document 25. That 9 is strange. Let me see if I can upload a different 10 version of that. I should be able to do it quickly. 11 All right. Do you see an Exhibit 1005? 12 DEPOSITION TECHNICIAN: Let me refresh 13 here and see if we can get that. 14 Yes. Give me a minute just to get it 15 downloaded. 16 It is giving me that same notice. I can 17 share it on the screen. I just won't be able to 18 save it without removing that content into that 19 Dropbox file. 20 MR. DOKHANCHY: Let's try a different one. 21 Do you still have access to Exhibit 6 from Friday's 22 deposition? 23 DEPOSITION TECHNICIAN: Yes, I should. 24 MR. DOKHANCHY: Can we mark that as 25 Exhibit 5 for this deposition.	223 1 correct? 2 A. Yes, it is. 3 Q. All right. And let's jump back to your 4 declaration. 5 So in paragraph 62 of your declaration, 6 you are explaining how the memory device element of 7 Stuttard -- or of the '053 patent is disclosed in 8 Stuttard; is that correct? 9 A. Yes, sir. 10 Q. And you state, in the second sentence, 11 that processing blocks 106 receive instructions from 12 "thread manager 102 containing cache unit 1024 13 process both pixel and vertex data" and then you 14 have two citations; right? 15 A. Yes. 16 Q. All right. So I want to go to Stuttard at 17 that location. So the first one is on page 20. 18 Okay. So on page 20, starting at line 21, 19 there's a description of figure 9 in Stuttard; 20 correct? 21 A. Yes. 22 Q. And figure 9 of Stuttard depicts a 23 flowchart illustrating data processing using 24 Stuttard system; correct? 25 A. Yes.
222 1 DEPOSITION TECHNICIAN: Okay. Pulling 2 that one up now. 3 MR. DOKHANCHY: I don't think that was 4 Exhibit 6 from Friday's deposition. 5 DEPOSITION TECHNICIAN: Oh, I'm sorry. 6 That was document 6. 7 You mean the marked Exhibit 6? 8 MR. DOKHANCHY: Yes. 9 DEPOSITION TECHNICIAN: Got it. 10 And you wanted to re-mark this as 11 Exhibit 5 for today? 12 MR. DOKHANCHY: Yeah. If you can remove 13 the previous stamping and mark it as Exhibit 5 for 14 today. 15 DEPOSITION TECHNICIAN: Okay, yeah. 16 Sorry. This is going to take just a second. It's 17 in a different spot. 18 MR. DOKHANCHY: Okay. 19 (Exhibit 5 marked.) 20 DEPOSITION TECHNICIAN: Okay. Does it 21 look correct, Counsel? 22 MR. DOKHANCHY: Yes. Thank you. 23 BY MR. DOKHANCHY: 24 Q. Dr. Pedram, Exhibit 5 is the Stuttard 25 reference that you rely upon in your report; is that	224 1 Q. Okay. And so, your first citation was 2 page 20, line 23 to page 21, line 7 in terms of your 3 evidence for processing blocks 106 processing vertex 4 and pixel data. 5 So my first question is: In your first 6 citation here, is this describing pixel operations 7 or vertex operations or both? 8 A. So the citation -- or the excerpt from 9 line 21, I guess, of page 20 to line 7 of page 21 of 10 Stuttard discusses vertex processing threads. 11 Q. Okay. And if we go to figure 9 of 12 Stuttard. 13 What portions of figure 9 does that 14 correspond to? 15 MR. JOHNSON: Objection. Form. 16 A. I mean, this would generally correspond to 17 vertex data processing -- geometry processing-type 18 things. 19 Q. So the blocks in figure 9 saying "vertex 20 data to graphics system," "geometry processing," 21 those two; correct? 22 MR. JOHNSON: Objection. Form. 23 A. Definitely those blocks. Perhaps also 24 "data transfer between neighbors" is also included. 25 Q. Okay. What about "data output to memory

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 108 of 182

Conducted on January 8, 2024

225 1 plus sorted"? Is that within the vertex 2 processing -- 3 A. Probably not. This is part of the 4 binitization process. 5 Q. Okay. What is binitization, or binning, 6 as it is sometimes called? 7 A. Yeah. This is a line, an image to be 8 segmented in different segments corresponding to 9 different set of data items, being put in bins that 10 then are passed to some of the processing blocks for 11 processing. 12 Q. Now let's go to your second citation, 13 which is page 22, lines 4 to 32. 14 Does this citation describe vertex 15 processing or pixel processing or both, or neither? 16 A. It is pixel processing. 17 Q. And if we go down the figure 9. 18 What elements in figure 9 does the pixel 19 processing correspond to? 20 A. Everything below the binitization part, 21 including data read to PEs, PE/MEE data transfer, 22 and then, finally, PE/LEE data processing. 23 Q. And so, going back to page 20, Stuttard, 24 I'd like to walk through the flow of figure 9. 25 So starting on line 25 of page 20 of	227 1 saying. 2 Q. At this point in the process, is there a 3 division such that some processing blocks or 4 processing elements are responsible for particular 5 regions of the screen? 6 A. Yes, it can be. Yes, it is possible. 7 Yes. 8 Q. You see at page 21, line 15 to 21, the 9 reference states that in the later step, the 10 "primitive data is output from the PEs to the local 11 memory in order that it can be sorted by region." 12 Where it says that, it's talking about 13 region of the display; correct? 14 MR. JOHNSON: Objection. Form. 15 A. I responded. I said yes, the region 16 refers to region in the -- on the screen. 17 Q. Your audio is not being caught every time 18 you're answering. 19 A. No. I know. But I'm not sure why. 20 Q. Okay. Well, just bear with us. 21 And then, it states that the -- in 22 line 20 -- excuse me -- 18, the "binning unit sorts 23 primitive data by region since the data is generally 24 not provided by the host system in the correct order 25 for region-based processing."
226 1 Stuttard, the host system prepares data concerning 2 the verticis of the primitive graphical images to be 3 processed and displayed by the graphics system. 4 And then, the vertex information is 5 transferred to the graphics system ending in 6 line 30; is that correct? 7 A. Yes, sir. 8 Q. And then, starting at line 32, the data is 9 loaded into the processing elements of the graphics 10 system so that each PE contains data for one vertex. 11 Is that the next step? 12 A. Yes, sir. 13 Q. An then, the PEs being discussed here. 14 Are those processing elements within one processing 15 block, 106, in Stuttard, or is it talking about all 16 processing elements in all processing blocks? 17 A. It can be either. In one case, if the 18 size of the data is small, it's going to be done 19 only in one parallel processing block. 20 Q. Okay. So it can be either. 21 Are you saying it can be the case that the 22 data is loaded into the processing elements of just 23 one processing block or could be the processing 24 elements of multiple processing blocks? 25 A. That is correct, sir. That's what I'm	228 1 Do you see that? 2 A. Yes, I do. 3 Q. So these lines of Stuttard are saying that 4 in the earlier step we looked at, on page 20, when 5 the host system provides vertex information, it is 6 not provided in order such that processing blocks 7 can be devoted to particular regions; correct? 8 Is that a yes? 9 A. I'm thinking about the question. 10 I understand that what's being described 11 in Stuttard, that the data is generally not provided 12 by the host system in the correct order for 13 region-based processing, but it's not clear exactly 14 if that sentence is the same as what you are saying, 15 which is that the -- the vertex information is not 16 provided in an order such that processing blocks can 17 be devoted to particular regions. 18 So I mean, the issue being, is it a matter 19 of improved efficiency to do this kind of ordering 20 or is it a matter of necessity? That is, no other 21 possibility exists and you must do it this way if -- 22 if the image is big enough and there are multiple 23 regions in it and so on and so forth. 24 So it can be, but I'm not sure. 25 Q. Let's go back up.

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 109 of 182

Conducted on January 8, 2024

229 1 So in page 20, lines 25 through column 2 [sic] 21, line 1, this is describing where the host 3 system provides vertex to processing elements; 4 correct? 5 That was a yes? 6 A. Yes. 7 Q. And at this point in the process, the 8 vertex information has not been processed yet; 9 correct? 10 A. Correct. 11 Q. And as we saw in page 21, at that stage, 12 the data is not in the correct order for 13 region-based processing; true? 14 A. Yeah. I'm not sure what that means 15 exactly. The sentence you're repeating, that the 16 vertex information is not sorted properly for 17 region-based processing; right? 18 So the binning unit source primitive data 19 by region and then there's a statement that this 20 sorting is useful because the host system does not 21 provide discriminative data information in the 22 correct order for region-based processing. All that 23 is correct. 24 So that's my understanding, and beyond 25 that, I'm not able to opine about the details.	231 1 processing on that particular data; correct? 2 A. I mean, it describes the function of the 3 binning unit. That is correct. And it says that it 4 sorts primitive data by region and that the data 5 initially provided by the host is not in the correct 6 order for region-based processing. So if you want 7 to do region-based processing, you better sort data, 8 and that we all agree. I'm not in disagreement with 9 that statement. 10 Q. And so, the vertex processing that happens 11 before binitization in Stuttard is not region-based 12 processing; true? 13 MR. JOHNSON: Objection. Form. 14 A. That appears correct; that the vertex 15 processing that happened before this binitization, 16 of course, did not make use of the binitization 17 process and the region-based processing. 18 Q. And when I say "region-based processing," 19 you understand that I mean the allocation of 20 processing blocks to particular regions of the 21 display; correct? 22 A. I believe that's what you mean, but with 23 the caveat that it is not true that one region only 24 gets assigned to a processing block. Depends on 25 what you want to do in the region in way of the
230 1 Q. All right. And why are you not able to 2 opine about the details? 3 A. Because that's the level that is 4 discussed; right? So the binning unit sorts 5 primitive data by region. My understanding here is 6 that if you are trying to do, for example, rendering 7 of a large image, the full image, for example, you 8 might partition this into regions, and there are 9 primitives in these regions -- in these primitive 10 data are going to be sorted by region. 11 And then, potentially, one or more regions 12 will be assigned to one processing block. This may 13 exhaust all of the processing blocks but may not. 14 Maybe two of the processing blocks will be 15 sufficient to handle the regions. Maybe four of 16 them is needed out of the eight possible processing 17 blocks in the example system of Stuttard. 18 So there is no statement here, as I 19 understand it and as a POSITA would understand it, 20 that because you're doing region-based processing 21 that you cannot perform any other task at the same 22 time, some of the processing blocks may be idle and 23 can be used to do other types of processing. 24 Q. What Stuttard says here, on page 21, is 25 that before binning, there's no region-based	232 1 region processing -- or region-based processing. So 2 it is possible that depending on the number of 3 processing elements in a given processing block, one 4 or more regions are assigned to the same processing 5 block or that the number of regions is small. For 6 example, there's only two regions and therefore, 7 only two of the processing blocks are occupied 8 during this region-based processing. Six of the 9 other remaining processing blocks are available to 10 run other tasks such as pixel processing. 11 Q. The region-based processing and assignment 12 of processing blocks to regions only happens in the 13 pixel processing phase, which is after binitization; 14 correct? 15 A. So pixel processing in this linear 16 graphics pipeline of Stuttard happens after 17 binitization, but that's with respect to a given 18 task. There may be another task in a different 19 stage of its processing at the same time. 20 Q. Okay. Looking at page 21, lines 3 to 7. 21 After the processing elements received the 22 vertex information, Stuttard states the received 23 data is then processed to transform it from the host 24 system reference space to the required screen space. 25 Do you see that?

Conducted on January 8, 2024

233 1 A. Yes. 2 Q. What is that describing? 3 A. This is essentially known as "vertex" or 4 "geometry" processing in which you are doing 5 coordinate transformation as described from the 6 host-system reference space to the screen space, and 7 this involves making solid computations, 8 transformations, such as lighting and shading, et 9 cetera. 10 Q. And so, before the data is processed to 11 transform it into its location and screen space, it 12 exists in host-system reference space; is that 13 right? 14 A. Yeah. 15 Q. What is host-system reference space? 16 A. We call it -- what we are discussing here 17 is that -- there's a -- an application describing a 18 graphics scene, for example, in a -- as part of a 19 computer program inside the host-system memory, and 20 our job is to take this description of the graphics 21 scene and visually display it on a two-dimensional 22 screen that the user is looking at. And so, the 23 transformation needed for recording its use in the 24 host system to the coordinates of the screen space. 25 Imagine this application is written	235 1 may be hidden on the user screen because of the 2 lighting or shading conditions and viewing angle, so 3 it may be behind something else. 4 So the primitive actually exists. It is 5 in the host-system reference, but you don't see it 6 in the display because of the viewing angle or 7 lighting or shading or what have you. 8 Q. And is that location information that is 9 the result of that processing what the bidding unit 10 uses to sort the primitive data by region and allow 11 region-based processing? 12 MR. JOHNSON: Objection. Form. 13 A. I'm not sure exactly what -- at least I 14 don't recall what the information the binning unit 15 uses to sort primitive data by region. 16 Q. Well, maybe more simply, before the vertex 17 information is processed, the system would not be 18 able to sort that information by region because it 19 doesn't know where the ultimate primitives will be 20 located there? 21 A. In Stuttard's system, binitization happens 22 after vertex processing based on disagreement there, 23 from my perspective. 24 Q. But also it could not happen before vertex 25 processing because the location information is
234 1 generically for all kinds of user display devices 2 with all types of resolution and so on. So it's 3 part of the job of the graphics processing engine to 4 take that single description of the graphic scene in 5 host-system memory and so-called "reference space" 6 and translate it appropriately to a different space. 7 So this can be small or large or different 8 number of pixels, could have also the display itself 9 is dependent on the viewing angle, the light source, 10 the characteristics of the surfaces in terms of 11 their textures, and so on. 12 So that's what the graphics processing 13 involves. 14 Q. As a part of this step described in the 15 first paragraph of page 21 of Stuttard, does the 16 system determine the locations on the screen where 17 the primitives will be located? 18 A. Yes. 19 Q. And that's the first time in the process 20 where that information is determined? 21 MR. JOHNSON: Objection. Form. 22 A. Yes, that's -- that's -- I'm not sure the 23 first time, but the -- it is a time at which this 24 information will be calculated and -- and presented 25 or used. The fact is that a particular primitive	236 1 unknown until the vertexes are processed; true? 2 A. That, I disagree because the -- in a 3 pipeline system, it is possible that as soon as some 4 information becomes available, the binitization 5 based on that information is going to be done and 6 the pixel processing can start after that. So it is 7 not necessarily true that you must finish all of the 8 vertex processing on a scene before you can do any 9 binitization or before you can do any pixel 10 processing. 11 If the region is -- I mean, if the screen 12 is large enough that you could start processing -- 13 doing vertex processing on some quadrant, some 14 portion of the image and while you're doing that, 15 you're making process and then you have all the 16 information that you need to do binitization on that 17 quadrant or that super region of the -- to avoid 18 confusing term with the region here on the -- on 19 that portion of the image -- and then it start doing 20 binitization there and define regions and start 21 doing region-based processing, pixel processing on 22 that portion without getting to the other sides, 23 yes, depends how the application is written, depends 24 on how the system is doing it for this to happen or 25 not. The possibility certainly exists that while

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 111 of 182

Conducted on January 8, 2024

237 1 you're doing vertex processing in -- on one portion 2 of an image, you're doing pixel processing or 3 another portion of that same image at the same time. 4 Imagine a graphic scene that's being 5 updated, right, and it's being updated from one 6 frame to the next frame on the -- on two of its 7 primitives, right, and one of the primitives is 8 being updated based on its graphics processing and 9 then it's going to start doing -- doing its pixel 10 processing and when its -- you're doing pixel 11 processing on the updates of one object on the 12 scene, you are doing vertex processing on the update 13 of the second object on that scene so that you have 14 two different tasks running on two different 15 objects. Within the same scene, one of them is in 16 its pixel processing stage, one is its vertex 17 processing stage. 18 It's possible. Depends what the input is 19 that is being passed to it and how the input program 20 is written that makes use of the Stuttard system. 21 The possibility certainly is there, as was discussed 22 also in Stuttard by having eight different 23 processing blocks running potentially eight 24 different tasks at the same time. 25 Q. So I think we're talking about two	239 1 system in which the fundamental invention has been 2 to organize the SIMD array of processing elements in 3 a set of independent processing blocks, for these 4 independent SIMD processing blocks to execute 5 multiple tasks where one of the tasks is vertex 6 processing thread execution? 7 The other one is pixel processing thread 8 execution, and my view there is: Yes, it's 9 possible. That was the intention and the 10 fundamental invention of Stuttard. That's what he 11 discloses, and -- and that's the basic difference, I 12 think, between us, my perspective on this and AMD 13 experts' on the other hand. 14 Q. All right. So I understand you want to 15 answer a different question, but respectfully, you 16 did not answer my question. 17 My question is simply: In the work flow 18 of figure 9, the pixel processing that is 19 region-based and the portions you identified of 20 figure 9 cannot occur until the vertex processing 21 for all the vertices has been done because that is 22 the stage that will determine where the primitives 23 will be, which enables region-based pixel 24 processing; correct? 25 A. Subject to everything I said before, all
238 1 different things. You're talking about whether it 2 would be necessary in all cases to wait for vertex 3 processing, but my question is little bit different 4 and it's more at the system-design level. 5 In the work flow of figure 9 and at the 6 system-design level, Stuttard is designed so that 7 the pixel processing is region-based, which can only 8 occur in Stuttard on a given set of pixels after it 9 has been determined by the vertex processing what 10 primitives are in that region; correct? 11 MR. JOHNSON: Objection. Form. 12 A. So it is correct that as was the case with 13 all graphics processing systems at the time of the 14 '053 mention and the prior art that we are 15 discussing, including Stuttard, a graphics 16 processing system would follow a linear graphics 17 processing architecture, meaning that for us or for 18 the system to get to the pixel processing stage, the 19 vertex processing has been done. 20 The question is not this linear graphics 21 processing, which was the conventional design, very 22 well known and practiced at the time of both the 23 '053 invention and the Stuttard reference, but the 24 fact -- the question really is: Is it possible for 25 the parallel processing system, such as the Stuttard	240 1 the caveats and descriptions, the answer is: Yes. 2 Q. What is rastering or rasterization? 3 A. This is one of the final stages of 4 displaying a scene or image from a host space on a 5 two-dimensional screen. The rasterization process 6 allows us to do the sort of display of the 7 individual pixels on the screen with the appropriate 8 brightness and color. 9 Q. Does rasterization come before, during, or 10 after pixel processing? 11 A. It is the last stage; comes after pixel 12 processing typically in the linear graphics pipeline 13 that we're discussing. 14 Q. Let's go to your declaration, 15 paragraph 68, and this is regarding the arbiter 16 element. 17 Do you see that? 18 Sorry. Was that a yes, Doctor? 19 A. Yes. 20 Q. And can you tell me what do you identify 21 as the arbiter in your report. 22 A. I think it's thread scheduler 1025 and 23 thread processors 1026. 24 Q. All right. Which of those performs the 25 required selecting of a command thread?

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

Conducted on January 8, 2024

241 1 A. The thread scheduler 1025 does that. 2 Q. And if we go to figure -- let's turn to 3 figure 4 of Stuttard. 4 Where do the -- withdrawn. 5 From where does the scheduler perform the 6 required selecting of a command thread? 7 A. I'm not sure what you mean "from where." 8 If you can elaborate. 9 Q. Sure. So we talked earlier about what it 10 means to select, and you said something to the 11 effect of "look at the available command threads and 12 you choose one", right? 13 A. Yes. 14 Q. So where is the set of command threads 15 that the scheduler chooses from in figure 4? 16 A. I mean, the -- for example, in the cache 17 memory block 1024, you can see there are eight 18 different threads shown here, BZ0B1 to B7, stored 19 and then the scheduler can decide which one or 20 ones -- two or more, potentially -- it's going to 21 activate and send to the array controller and/or 22 channel controller to be processed. 23 Q. What is the role of the thread processor 24 1026 in that process? 25 A. There's a detailed description in my	243 1 A. Just a correction on your question and my 2 response. 3 You said the "commands." I mean the 4 "command." I assume you mean the command threads. 5 So they do exist in cache 1024, these 6 command threads. 7 And then, your next question is you're 8 saying, well, from figure 4, it looks like the 9 scheduler has no connection to the cache there and 10 therefore, how does it make a selection? 11 There's a description, explicit language 12 about what the scheduler does in the Stuttard system 13 in its specification, and that one says that the 14 scheduler does this selection. Whether the detailed 15 connection from schedule to the cache block is shown 16 or not, yeah, it is not enough details about all the 17 connections between all different elements, 18 especially something like a scheduler. 19 Of course, it has to connect to the 20 threads because we know they say that in Stuttard. 21 That information is made available to it through the 22 status block example; right? You can see the 23 connection to the status block. 24 Q. The -- the eight threads that exist in 25 cache 24 -- 1024. Excuse me. Do those correspond
242 1 report. Let's refer to that rather than I try to do 2 it off memory. 3 Q. That's fine. 4 A. So on page 33 -- top of page 33 of my 5 declaration, third line down from the top, it reads, 6 "Thread processors 1026 control the issuance of 7 instructions from the thread manager to maintain 8 processing of simultaneously active program threads" 9 and then it references Stuttard, page 11, 27 to 33. 10 It goes on to say, "Stuttard thread 11 scheduler working in conjunction with the thread 12 processors in the thread manager arbitrates and 13 schedules a thread based on the relative priorities 14 and ready versus yielding status information from 15 each thread" and then cites Stuttard at page 12, 16 lines 12 to 18, as it has other places, 25, 35 -- 17 row 35 to page 26, row 32. 18 Q. So looking at figure 4 of Stuttard. 19 You are saying the commands from which the 20 scheduler makes its selection exists in the cache 21 1024; is that right? 22 A. Yes. 23 Q. How does the scheduler access those tasks 24 in particular? Because it doesn't look like it's 25 connected to the cache in figure 4.	244 1 to the eight thread processors 1026? 2 MR. DOKHANCHY: Objection. Form. 3 A. Not necessarily. Again, there's a 4 mechanism by which up to eight thread processors 5 could control the instructions streams that are 6 being passed on to eight different processing blocks 7 in the Stuttard system. In such a case, of course, 8 the eight threads running on those processing blocks 9 are being controlled by these thread processors. 10 Controlled in the sense that three threads 11 could have, for example, control-flow operations in 12 them, and so you need to decide what to do on 13 branches and so on and so forth, and the specific 14 sequence of instructions that gets executed are put 15 together by these thread processors, but it doesn't 16 mean that the thread processors, eight of them that 17 you see here, correspond to the eight that are 18 residing at any point in time in the -- in the 19 cache. 20 The timing of all the issue -- because, 21 obviously, the thread processors are already 22 starting to execute these threads, so the thread 23 processor may be finishing execution of one thread 24 and a new thread is coming in to replace it to be 25 sort of executed next.

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 113 of 182

Conducted on January 8, 2024

245 1 So the precise timing is unclear, but in 2 the most general case, there's going to be eight 3 independent threads run existing in the cache and 4 being executed in the parallel processing block. 5 You are halfway through processing them and then the 6 thread processors are working on all those eight 7 different threads independent of each other deciding 8 what instruction to issue next to the parallel 9 processing blocks. 10 Q. So if it is the thread processor that 11 decides what instruction to issue next to the 12 parallel processing blocks, what is the scheduler's 13 role in determining what command threads get sent to 14 the processing blocks? 15 A. Scheduler chooses the command thread. The 16 thread processor decides what instruction sequence 17 is going to be executed as part of that thread. 18 As I said, as clearly described in 19 Stuttard, this system can do all kind of threads: 20 General control threads, 2D activity thread, 3D 21 activity thread, and some or all of these type of 22 thread might have control flow in it. 23 You understand control flow means, for 24 example, a decision point there's a operation in the 25 middle of the thread that says, if X, go left and	247 1 Q. So once the thread processors are loaded 2 with threads, does the scheduler have any further 3 input on or regarding what command threads are 4 executed by the processing blocks? 5 A. Stuttard system enables this also in the 6 sense that if a thread is running but for any reason 7 the thread status changes from ready to yielding, 8 then that means you are going to stop the execution 9 of that thread and, potentially, use a different 10 thread, select another thread to run in place of it. 11 So that preemptive action or preemption is a 12 possibility in the Stuttard system. 13 High-priority task or thread running 14 instead of a background task. So I'm running eight 15 threads. I'm quite happy. Everything is going 16 well. Suddenly, a high-priority thread comes in. A 17 scheduler know it's high priority and replaces it 18 with thread processor, replaces another -- an 19 already executing thread with this one. 20 This is, sort of, rendered obvious in the 21 Stuttard, but then there's also the Williams, and 22 Whitaker, especially, that describe use of 23 priorities in the system. You combine them with 24 Stuttard to do that. 25 Q. And that selection that you're describing
246 1 execute instruction XYZ; if right, branch, then -- 2 sorry -- you go to the right branch and then execute 3 operation instructions A, B, C. 4 So the thread processor decides whether I 5 went left or right on that control flow-up point, on 6 that decision point, and therefore, whether we 7 should execute X, Y, Z instruction subsequence or A, 8 B, C instruction subsequence so it's a fine-level 9 control flow in -- in -- in when the thread has 10 already been selected by the scheduler. 11 This is all described in Stuttard. I 12 mean, it's not something I'm, sort of, guessing 13 what's going on. There's clear language, explicit 14 language in Stuttard that says that. You want me to 15 find it for you, I'll look again. 16 Q. Well, maybe. So I'll get to that if I 17 need you to do that. 18 A. Anyway, just for reference, please, if 19 there's any confusion, read the bottom of page 38, 20 top of page 39 of my report in which I describe that 21 the thread processors allow high-level flow control 22 to be performed for a thread and to assemble 23 instructions to be issued through the array 24 controller and channel controller, for example; 25 right?	248 1 by the scheduler based on priority. Does that 2 happen as to threads that are already loaded in the 3 thread processors 1026? 4 A. It can. I think it -- Stuttard describes 5 the status of a thread to be "active" or "yielding." 6 "Active" means it's running. "Yielding" 7 means it yields, it's not necessarily finished, 8 because if a task is -- if a thread had finished, 9 it's completely outside the radar screen of the 10 scheduler. It's done; right? It's gone. 11 So the fact that a task is either active 12 or yielding, the yielding part means the task is -- 13 was running but now, for some reason, it's yielding, 14 saying somebody else could run and this could happen 15 in maybe some applications because you're trying to 16 access some data item that's not ready for you. 17 There are dependencies in the running applications' 18 tasks, and because the data item you need to 19 continue is not ready, you say, "Well, I can't do 20 anything anyway. I have to wait, so now, therefore, 21 I change my status from ready to yielding." And 22 then, when a status of a task -- a running task or 23 running thread changes from ready to yielding, then 24 it could potentially get sold out by the scheduler 25 that sends another task in to preempt it to increase

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 114 of 182

Conducted on January 8, 2024

249 1 the computational throughput of the system. That's 2 very common in all kinds of panel processing 3 systems. It's very common. 4 Q. And so, what are you pointing to as the 5 priority of the claims? Is it that yielding or 6 ready status or is it something else? 7 A. Let's look at the section of my report 8 that discusses this. 9 So on paragraph 69 of my declaration, 10 there's a full discussion of why and how I think the 11 thread scheduler 1025 of Stuttard meets the 12 requirement of the priority based selection of the 13 threads. 14 So it says, "For example, the thread 15 scheduler 1025 works with the thread processors 1026 16 to schedule and activate the highest priority 17 thread." 18 And then, I called an excerpt from 19 Stuttard that reads like the following: "We think 20 the thread manager 102, the status block 1030 21 receives the status information 1036 from each of 22 the threads, and that is status information by the 23 ways ready versus yielding, but the status 24 information is used by the thread scheduler 1025 to 25 determine which thread should be active at any one	251 1 A. Yeah. I understand the question. I 2 believe I answered it. I'll repeat it one more time 3 quickly. 4 So the scheduler looks at the threads in 5 the cache to decide which one to activate and -- or 6 ones to activate and -- and send to the parallel 7 processing blocks, but if, when running, one of 8 these threads changes its status to yielding for any 9 reason, then that information is still conveyed to 10 scheduler that we'll go back and replace the 11 yield -- potentially replace the yielding thread 12 with a new thread that just came in around. 13 And so, therefore, the scheduler is 14 involved in thread selection both when the threads 15 are first in the cache and even later, when they are 16 in one of the processing blocks but they have 17 changed their status to yielding. 18 Q. When does a thread receive or have a ready 19 or yielding designation? When does that first 20 happen? 21 MR. JOHNSON: Objection. Form. 22 A. Don't know when it happens first, but I 23 don't recall. But I think this part that describes 24 all of this is on Stuttard -- in Stuttard reference. 25 Page 23 starts that discussion. So we go both read
250 1 time. Core instructions 1032 issued by the thread 2 manager 102 are sent to the array controller 104 and 3 the channel controller 108, figure 3." This is on 4 page 12, line 13 to 22. 5 I go on to say, "Stuttard also teaches 6 that the scheduling of the time multiplexing between 7 threads is determined by the dynamic conditions of 8 the program execution based on thread priority," and 9 I reference detailed discussions in Stuttard 10 starting on page 23, line 35 to page 25, line 32. 11 And then, "The thread processor 1026 uses 12 the thread scheduler to assemble instructions for 13 the array controlled on the floor and channel 14 controller 108." Stuttard page 26, 3 to 7, and 15 continues. I don't want to read the whole thing. 16 Q. Yeah. Thank you for reading that. 17 My question is just simply -- you know, we 18 can discuss this in the context of figure 4, if 19 that's helpful, but if not, that's fine. 20 The priority that the scheduler uses in 21 order to select command threads. Is that priority 22 something that is known by the system while the 23 commands are in the cache or is it while the 24 commands are in the thread processor or at some 25 other point?	252 1 it and try to answer the question you asked. 2 Q. Yeah, let's do that. 3 A. So you can see that Stuttard, bottom of 4 page 23, under the section heading "Determining 5 Which Thread Should Be Active," starts describing 6 the process. I'll read just a little bit of it and 7 maybe I can read the rest to myself. 8 But anyway. So "The thread scheduler, 9 when running, recalculates which thread should be 10 active whenever one of the following scheduling 11 triggers occur: A thread higher than the prior 12 activity thread is ready, hence the idea that it's 13 looking at the priority, or the thread is not ready 14 and yielding." 15 So under any of these two trigger 16 conditions, the thread scheduler functionality kicks 17 in and says, let me reschedule. Let me do another 18 selection. 19 So in the first case, it will go and 20 replace a lower priority thread with another -- with 21 a high-priority thread that just came in. 22 In the other case, a thread is running, 23 but it's yielding and it goes and replaces it with 24 some other thread. 25 And so, that's the basis for what I'm

Conducted on January 8, 2024

253 1 saying, and the process continues. You can see 2 there's a lot more discussions there that keeps 3 going in the multiple conditions and so on. 4 So again, do you want to read it? We can 5 read it together. If this is sufficient, we can 6 stop here. 7 Q. So it doesn't quite answer the question. 8 So let's ask it this way: So a command or a set of 9 commands or command threads -- withdrawn. 10 A command thread, according to you, comes 11 in and is stored in cache 1024. 12 Does it yet have a status of yielding or 13 ready at that point? 14 A. So again, therefore, we should read that 15 section because the questions you're asking are 16 answered in that section. 17 So page 24, line 14 of Stuttard says, "In 18 determining the above, a thread is always deemed to 19 be ready unless it is waiting on an instruction 20 cache, waiting on a zero" -- "zero semiform, waiting 21 on a busy execution unit, or waiting on a haul 22 instruction." 23 So that defines when a thread is going to 24 be declared ready, and a -- a thread like this could 25 be in the cache or could be executing, depending on	255 1 Q. Do you see anything -- excuse me -- in 2 Stuttard explaining where the priority information 3 is stored? 4 A. Not that I recall. I mean, there's this 5 statement, which is expressed disclosure that the 6 Stuttard system can clear with higher priority 7 threads, but where that information is stored, I 8 don't recall. 9 I know the status block has the yielding 10 versus ready status of the threads, so that 11 information is there, but the priority in terms of 12 high-priority task versus low-priority task 13 information, where it is stored, I don't recall. I 14 would assume it's a piece of information attached to 15 the thread that tells what the priority of the 16 thread is that's being made available to the thread 17 manager because we know it makes use of it; right? 18 It's clear that the thread manager knows about this 19 information and makes use of it, but exactly where 20 it is stored, I don't remember. 21 Let me do a check here. I don't remember 22 any place that described explicitly as where that 23 information is stored. 24 So on page 25 of Stuttard, line 5, it says 25 that "if the previously active thread is yielding,
254 1 the condition that's discussed here. 2 So for example, waiting on a busy 3 execution unit means you are still in the cache 4 waiting for the execution unit to become available 5 to you. On the other hand, waiting on an 6 instruction cache unit means you are executing, you 7 are in the parallel processing block being executed, 8 but then, you -- you got an instruction cache, so 9 you change. You are not ready now because now you 10 have instruction cache unit; right? 11 I mean, so it answers the question you're 12 asking. It could be in either case. I don't think 13 you can make a determination that the statement that 14 in Stuttard, the scheduler is only active making 15 decisions in one case versus the other, one case 16 being in the cache, threads residing in the cache 17 memory, the other one threads running in the -- in 18 the parallel processing block. 19 And then, it goes on to describe yielding 20 versus not yielding, right, also, so there's another 21 level of discussion. But if you just stay at the 22 top level, it clearly stated that thread with higher 23 priority can replace an active thread and that the 24 threads could be in the yielding state, meaning that 25 they could be replaced.	256 1 the scheduler activates the highest priority ready 2 thread or the lowest priority thread if no thread is 3 ready." 4 So it clearly has the notion of low 5 priority, high priority and this and that. 6 Q. Is it your opinion that the indication of 7 ready or yielding is a type of priority? 8 MR. JOHNSON: Objection. Form. 9 A. I'm sorry. I'm still on the previous 10 question. So I just found claim -- there's a lot of 11 discussion about low-priority, high-priority threads 12 in Stuttard, but claim 31 of Stuttard, on page 65, 13 reads that "a data processing apparatus as claimed 14 in claim 28, 29, or 30, wherein each instruction 15 stream, each thread, is assigned a relative priority 16 level." 17 I mean, okay. So now I'm sorry. Your 18 question was... 19 Q. It was whether you consider the 20 designation ready or yielding to be priority 21 information? 22 MR. JOHNSON: Same objection. 23 A. I'm thinking. 24 So when a task declares itself to be 25 yielding, it implies that it was low-priority task.

Conducted on January 8, 2024

257 1 A high-priority task would normally not -- as far as 2 I know or I understand, would not declare itself to 3 be yielding because it knows it's high priority, and 4 once you release a hardware resource to be nice to 5 others, you have to wait to get back that hardware 6 resource. 7 So if you are high-priority task, for 8 example, a realtime graphics processing task, then 9 you would not assume a yielding status even if you 10 know you have to wait for some period of time for 11 some things that you need to become ready for you to 12 use. 13 So I would say not the ready part because 14 the ready means if I'm ready to run or not or as 15 defined, but certainly the yielding part of that 16 status information conveys the notion of low 17 priority versus high priority to some extent. 18 So yeah. That's all I can say. 19 MR. DOKHANCHY: All right. Let's go off 20 the record. 21 THE VIDEOGRAPHER: We're going off the 22 record. The time on the video monitor is 12:05 p.m. 23 (Recess.) 24 THE VIDEOGRAPHER: We are back on the 25 record. The time on the video monitor is 12:37 p.m.	259 1 Q. So whatever order to instructions come 2 into the FIFO is the same order they'll leave the 3 FIFO; right? 4 A. Yes, or thread instructions or threads or 5 what have you. Yes. 6 Q. How does the arbiter in Stuttard connect 7 to these FIFOs? 8 A. I think this is the same question that you 9 asked previously about the arbiter versus the memory 10 component of Stuttard as part of its thread manager. 11 And I did explain that the arbiter, which 12 is the schedule and thread processors, do have 13 access to information in the cache memory block 14 1024, and therefore, if you go and replace the cache 15 memory block with a series of FIFOs as disclosed by 16 Stuttard's reference on page 23, lines 3 to 11, then 17 you have the same access to these FIFOs as you had 18 to the memory. 19 Q. Does the arbiter in Stuttard have access 20 to memory locations in the middle of the FIFO or 21 only the last part of the FIFO? 22 MR. JOHNSON: Objection. Form. 23 A. Having access is different than how data 24 is pushed in or pulled out of the FIFO, as was also 25 in the background section of the '053. You could
258 1 MR. DOKHANCHY: All right could we get the 2 exhibits back up, please. In particular, Exhibit 1. 3 BY MR. DOKHANCHY: 4 Q. Dr. Pedram, Exhibit -- we're looking at 5 Exhibit 1, starting on page 45, paragraph 73. 6 You address dependant claim 3 of the '053 7 patent, which requires two memory devices; is that 8 correct? 9 A. Yes, correct. 10 Q. And what you point to is Stuttard's 11 disclosure that the cache 1024 could be replaced by 12 a series of FIFOs, one per thread. 13 Let's start with what does "FIFO" stand 14 for? 15 A. "FIFO" is the "first-in, first-out" data 16 structure component. 17 Q. And what does that mean, first-in, 18 first-out? 19 A. So the task -- usually, tasks enter, say, 20 from the front end of the FIFO and exit from the 21 back end. So first-in, first-out means that -- 22 first-in, first-out means that the task that enters 23 first from the front end is also the task that 24 leaves first from the back end of the -- this 25 component.	260 1 have these FIFOs and your system component 2 controller arbiter could have access to -- access 3 means meeting for the meeting, right, doing what's 4 there. 5 So it knows what's there, like, in all of 6 the FIFO. But the way FIFO is designed, you can 7 only take items out in the order they were injected 8 into that FIFO. 9 Q. Can the arbiter select what data it wants 10 to receive from a given FIFO in Stuttard? What -- 11 A. So the -- this is somewhat complicated to 12 respond to, and the reason is: Although Stuttard 13 describes the -- this cache memory could be replaced 14 with the eight FIFOs, one corresponding to each of 15 the processing blocks and the thread processors, at 16 the same time, it also discloses that the higher 17 priority thread could replace a low-priority thread. 18 So if the low-priority thread is in front 19 of the queue about to come out but then you have a 20 high-priority thread behind it, although Stuttard 21 does not in detail explain how it would be doing it, 22 but because we know it does it, because it 23 explicitly discloses that it can replace the 24 yielding low-priority thread with the high priority 25 one, then I, a person skilled in the art, would

Conducted on January 8, 2024

261 1 understand that there is a mechanism in Stuttard 2 that's not expressly discussed that I recall that is 3 that would allow it to actually go and pick a 4 high-priority thread, potentially somewhere in the 5 middle of the FIFO, and bring it to the forefront so 6 it could run next. That mechanism should be there 7 from the other disclosures in Stuttard. 8 Q. In dependant claim 7 in ground 1, you 9 point to Stuttard's disclosure of lighting and 10 shading tasks as evidence of a texture processing 11 engine; is that correct? 12 A. Can you point me to the specific part. 13 I'm sorry -- oh, yeah. I see. In paragraph 84. 14 Yes, I do. 15 Q. I guess, first, what is a texture 16 processing engine? 17 A. It's a processing engine to do the texture 18 rendering as a function of the valuations in the 19 brightness intensity as well as the surface 20 properties that have been input to the system and 21 perhaps also the viewing angle. 22 Q. Is "texture processing engine" a term of 23 art in the graphics processing field? 24 A. It's a commonly used art term, but it 25 might not have a unique definition assignment across	263 1 There might always be a complicated, 2 cumbersome way of doing things some other way, but 3 that's not what a person skilled in the art would 4 understand it, and that's not how he would perform 5 it. 6 Q. I think my question is little bit 7 different, but just let me try again. 8 Is it possible to do something that you 9 would consider a lighting task that is only lighting 10 but not a texture task? 11 A. Yes. If you are just -- so for example, 12 there's a process of maybe deciding what their 13 reflections are of a light source on a polygon, a 14 primitive, in the image. And of course, this 15 involves first deciding what the reflection should 16 be at the corner points of that polygon. So that's 17 the vertex processing thing. 18 But then you have to also extrapolate 19 between the corner points of the polygon to decide 20 along the surface of the object, depending on the 21 curvature of the object and the property of the 22 surface itself, if there is polished surface or not, 23 for example, what the reflections would look like. 24 That second part of lighting definitely involves 25 pixel processing.
262 1 different surfaces and sources. People usually 2 define different tasks -- graphics processing tasks 3 to be part of it. Or not; it depends what reference 4 you are consulting and so on. But it is the term 5 well known that you're trying to do the processing 6 of the texture of a surface of an object. 7 Q. Is there anything called a "texture 8 processing engine" in Stuttard? 9 A. No. The term "texture processing engine" 10 has not been used, although there's obviously 11 discussion of textures and processing these 12 textures. 13 Q. Is it possible to do lighting -- 14 withdrawn. 15 Is it possibly to process lighting, 16 commands, or command threads in a way that does not 17 implicate textures? 18 A. I don't believe so. I understand AMD 19 expert, Dr. Hart, opined on this saying that texture 20 processing can also be done as sort of a vertex 21 processing task. 22 I disagree with that statement. The 23 standard way of doing texture processing is to do 24 pixel processing, and that's what the state of the 25 art and state of the practice has been for decades.	264 1 So again, when you look at lighting and 2 ask the question about lighting, yeah, lighting 3 verticis is about vertex processing lighting, like 4 in surfaces is pixel processing. When we talk about 5 texture processing, typically, it is not about 6 vertexes -- verticis on a polygon but about the 7 surface of the polygon. 8 Q. Again, I think we're talking past a little 9 bit -- talking past each other a little bit. I'm 10 not asking about pixel processing. I'm just asking 11 about lighting and textures. 12 So my question is: Are all lighting tasks 13 necessarily also texture tasks or are there some 14 lighting tasks that can be performed without texture 15 considerations? 16 A. I think the appropriate question to ask is 17 that when a POSITA look at the Stuttard reference 18 and finds this description of lighting and shading 19 performed to produce data, depending upon the chosen 20 point of view, would understand this lighting and 21 shading to be a task that involves pixel processing 22 and texture mapping or not. 23 And my answer there is: Yes, definitely a 24 POSITA looking at this disclosure in the context of 25 Stuttard's reference would understand this lighting

Conducted on January 8, 2024

265 1 and shading to be examples of the texture processing 2 engine. 3 Now, in some hypothetical case, maybe 4 there is a very limited definition of what lighting 5 means in that application and we are only looking at 6 lighting a single vertex but nothing else. Yes, in 7 the context of that single vertex being lighted, 8 perhaps you are just doing vertex processing there, 9 but that's not how a POSITA would understand this 10 disclosure in Stuttard. 11 Q. To be honest, I'm still very confused 12 about your reference to vertex and pixel processing. 13 I'm just asking about lighting and shading versus 14 texture. So -- and I understand you said that my 15 question is improper and you wanted to answer a 16 different question, but respectfully, I am entitled 17 to an answer to my question. 18 And my question is: Does every lighting 19 and shading task necessarily involve texture 20 processing or are there some potential lighting and 21 shading tasks that do not require processing of 22 textures? 23 A. My response to you, sir, is you haven't 24 given me enough information. You're asking me to 25 look at the universe of all possible lighting and	267 1 Q. Grounds 1 is your Stuttard-based 2 reference -- ground, rather, and ground 2 is 3 Stuttard plus Williams. 4 Does that seem correct? 5 A. Yes, that's correct. 6 Q. Okay. And in ground 2, the combination of 7 Stuttard and Williams, you point to the same arbiter 8 in paragraph 107 as you do in ground 1; correct? 9 A. Correct, yes. 10 Q. Okay. And you've not tried any opinion 11 that modifies the arbiter of Stuttard for the 12 purposes of ground 2; correct? 13 A. Not here. I mean, in paragraph 107, as 14 you read -- you can read as I do, I state that 15 Stuttard discloses element 1B, which is the arbiter, 16 and all the other required performance on the 17 arbiter for the same reasons I discussed in ground 18 1, which is Stuttard. 19 So yeah, that sentence is accurate with 20 respect to this paragraph. 21 Q. Okay. But just to be clear, my question 22 isn't limited to this paragraph. My question is 23 about your opinions with respect to ground 2. 24 So in ground 2, you have not provided any 25 opinion that modifies the alleged arbiter in
266 1 shading tasks in all kinds of applications in all 2 kinds of systems, and saying "Can you think of any 3 such occurrence in which the lighting and shading 4 does not involve texture processing?" And I'm 5 saying I don't think so, but more importantly, I 6 don't know. It's not something I could speculate. 7 If you give me the specifics of a system, 8 the specifics of an application, a graphics 9 processing task, I could respond to you. And I gave 10 you examples on which lighting and shading could 11 involve vertex processing or pixel processing. You 12 don't think I'm being responsive to your question. 13 So your specific question, I can not answer as 14 asked. 15 Q. Okay. Let's go down paragraph 107 of your 16 report. 17 And this is in your ground 2, which 18 combines Stuttard with Williams; is that correct? 19 A. It can be. Let me check, please. 20 Yes, it is. 21 Q. Okay. And in ground 2, you allege the 22 same arbiter as you do in ground 1; correct? 23 A. I need to get a little bit of context 24 because ground 1 -- can you remind me, please. 25 Ground 1 refers to --	268 1 Stuttard from ground 1; correct? 2 A. Right. But when you combine elements of 3 Williams into the Stuttard system, you integrate 4 them. For example, if you add GBS, this scheduler, 5 the unit of Williams into the system of Stuttard, of 6 course, that GBS itself participates is a component 7 of the combined system and plays a role. It doesn't 8 replace the arbiter, as identified here, but 9 provides additional functionality that a person 10 skilled in the art would understand what it is in 11 view of the combined reference, Stuttard and 12 Williams. 13 Q. I'm not sure I understood any of that. 14 But my point is, sir, when it came time to 15 identify the arbiter in ground 2, you didn't point 16 to anything other than Stuttard as you disclosed it 17 in ground 1; correct? 18 A. With respect to this paragraph, 107, I 19 agreed with you, sir. I did say that already. 20 Q. And paragraph 107 is the only place in 21 your report in ground 2 where you say what the 22 arbiter is; correct? 23 A. Correct. I'm not changing -- 24 MR. JOHNSON: Objection. Form. 25 Q. Okay.

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 119 of 182

Conducted on January 8, 2024

269 1 THE WITNESS: I didn't hear -- Jeffrey, 2 was -- was something said? 3 MR. JOHNSON: I just objected to form. 4 THE WITNESS: I see. 5 A. And so, please ask the question again. 6 Maybe the part of this is that -- 7 Q. Okay. 8 A. -- I'm not sure what your question relates 9 to. If it's generally relating to this, the 10 statement that Stuttard, in my paragraph 107, only 11 identifies the arbiter of the ground 2 to be the 12 arbiter of the ground 1, I'm not disagreeing with 13 that. 14 If the statement is, "Does any element of 15 the William system getting incorporated into 16 Stuttard system would help Stuttard's arbiter in 17 some ways?" the answer is: Yes, it does. 18 But I didn't explain it here. It is in 19 the body of my report, but I didn't explain it here. 20 Q. Okay. Let's introduce, as Exhibit 6, 21 document 26. 22 (Exhibit 6 marked.) 23 BY MR. DOKHANCHY: 24 Q. Okay. Exhibit 6, Dr. Pedram, is the 25 Williams reference you rely on in your report; is	271 1 And the destination of these two buses are 2 different. 116A goes to unit 402, then can, for 3 example, do vertex processing, and bottom one goes 4 to 404, and that can do pixel processing. 5 Q. Okay. All right. Let's take them, sort 6 of, one by one. 7 So the input buffer of Williams in the 8 combination of ground 2. Are you proposing that the 9 input buffer would be an additional component to 10 Stuttard's architecture or would it replace any of 11 the components, such as the cache in Stuttard? 12 MR. JOHNSON: Objection. Form. 13 A. It could be in either configuration. We 14 could add an input buffer, for example, just before 15 cache of Stuttard or we could treat the cache of 16 Stuttard as the input buffer. So I -- I would not 17 pick the one versus the other. I think both 18 possibilities exist and are consistent with both 19 Stuttard and by Williams. 20 Q. Why would a POSITA add the input buffer 21 and keep the cache in Stuttard? 22 A. For a number of reasons. 23 The input buffer could be placed very 24 close to the system bridge, physically close so it 25 would be easy to write into it and make use of it in
270 1 that correct? 2 A. Yes. 3 Q. Okay. And just looking at figure 4, what 4 parts of Williams do you propose including in this 5 ground 2 combination of Stuttard claims? 6 MR. JOHNSON: Objection. Form. 7 A. So the way -- I guess Stuttard in 8 combination -- I'm sorry -- Williams in combination 9 with Stuttard is to show that although a POSITA 10 would have found it obvious that the Stuttard system 11 includes -- stores both vertex and pixel threads in 12 its memory system, to the extent that would be 13 helpful, one can look at the William reference in 14 which there's an input buffer connected to the 15 system bridge 106, and Williams very explicit 16 discloses that the -- this input buffer and/or the 17 system memory 108 sitting on top of the system 18 bridge would contain both pixel and vertex command 19 threads, even if at the same time sitting there in 20 the input buffer, because then the global bus 21 scheduler of Williams, GBS204, takes the command 22 threads in the input buffer and dispatches the 23 vertex command threads on the top of the bus 116A 24 and the pixel command threads on one or more of the 25 bottom buses; 116B, for example.	272 1 that way. 2 It could also -- I'm sorry. Let me 3 finish. And so, if you have -- if you are a 4 minimalist trying to minimize any changes to the 5 Stuttard system, as you combine it with Williams, a 6 POSITA would understand that the input buffer is 7 just another buffer, and it's easy to just add a 8 buffer before the cache. 9 And then, you can think of the combination 10 of the input buffer and the cache as serving as the 11 memory element. But it's just a buffer and you can 12 add it into the system. And these are pipeline 13 systems, so meaning that it -- I mean, one more 14 input buffer there doesn't change the throughput -- 15 processing throughput of the system. It could 16 provide benefits but doesn't reduce the processing 17 performance because you added something on the path. 18 Usually, pipeline architecture doesn't matter. 19 So I think both possibilities exist, and 20 I -- I -- 21 Q. Okay. So, you know, if you're a POSITA, 22 such as yourself, looking at Stuttard, and you also 23 know about Williams, what motivates you to add the 24 input buffer to Stuttard? 25 MR. JOHNSON: Objection. Form.

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 120 of 182

Conducted on January 8, 2024

273 1 A. So again, I can either add it or integrate 2 it. 3 Q. Let's start with add. 4 A. Yes. I agree with you. So let's start 5 with the add. 6 Again, as I said, the addition of the 7 input buffer would allow the -- the CPU system 8 bridge to put the command threads in two different 9 portions of the input buffer because, one, we know 10 that the designation for those two threads are going 11 to be different. One is going to go through one bus 12 to one execution unit. The other type of thread 13 will go through a different set of bus or buses to a 14 different execution unit. 15 Sequential access of the information has 16 its own benefits, well known to the person skilled 17 in the art at the time of the invention. You having 18 all of the vertex threads in one portion, all of the 19 command threads in another portion, in the input 20 buffer and from then, going to the cache and caching 21 them, putting there, and dispatching them to the 22 rest of Stuttard system, for example, it would make 23 sense to a POSITA who doesn't have any issues other 24 than providing additional benefits in terms of 25 organizing the command threads of one type in a	275 1 them. 2 And you know a pointer is four bytes. So 3 this is the first thread at address X. The next one 4 is found at address X plus 4. The next one is found 5 at address X plus 8 and so on and so forth. So 6 that's much easier than to say the first one is at 7 X, the second one is at Y, the third one is Z, and 8 you have to always change all of these things from X 9 to Y to Z before you can access it. 10 It's well known. I mean, everybody 11 acknowledges that, and there's no -- no novelty 12 here. It's just conventional state of the knowledge 13 about sequential access benefits to random access 14 benefits. 15 Q. So let's talk about what -- okay. So that 16 was the scenario where the input buffer is added to 17 Stuttard. 18 Are the benefits, in your view, any 19 different if one were to replace Stuttard's cache 20 with the input buffer of Williams'? 21 A. No. They are the same, as far as I can 22 see, in this context. 23 Q. Okay. The -- is the graphics bus 24 controller, the GBS, of Williams part of what you 25 would include in the combination in ground 2?
274 1 separate portion from the command threads of the 2 other type. 3 Q. So in the scenario in which the input 4 buffer is added, your testimony is it would be 5 beneficial to add the input buffer because it could 6 segregate command types in different parts of memory 7 and then those could go to the cache. 8 And that's beneficial why? 9 A. So because if I'm going to access pixel 10 command threads to send all of them to execution 11 unit 404 on a set of buses that I identify, if 12 though command threads are consecutively stored in a 13 portion of the memory, such as the input buffer, it 14 would make it more efficient from an access-latency 15 point of view to do this. 16 Let me elaborate. Generally, if you want 17 to access memory in a random manner, you have to 18 jump from one address to the next address, to next 19 address, and this jumping involves these branching 20 operations that are expensive and break the -- sort 21 of the flow of the program -- sequential flow of the 22 program. On the other hand, if they are all 23 consecutively stored -- for example, let's assume 24 they have pointers to beginning address of the 25 threads -- then you're just sequentially accessing	276 1 A. If I'm trying to incorporate the -- the 2 efficient data communication between the system 3 bridge and execution units in a graphics processing 4 system, such as taught by Williams, then the 5 graphics bus scheduler would also be a useful entity 6 to have because the graphics plus scheduler then 7 would allow a Stuttard system to take out the 8 entries from its input buffer or combine input 9 buffer and cache and put it on different buses that 10 carry this information to various processing blocks 11 of Stuttard. 12 So imagine there are eight processing 13 blocks, eight different buses going to these 14 processing blocks, and having a graphics bus 15 scheduler that will decide which command thread goes 16 to -- through which bus goes to which processing 17 block would be, again, a useful addition. 18 Q. So in ground 2, you are including the GBS 19 of Williams in the system of Stuttard; true? 20 MR. JOHNSON: Objection. Form. 21 A. So in my declaration, I did not feel the 22 need to show exactly how the different components of 23 Williams can be combined or incorporated within the 24 Stuttard system. The main use I was making of 25 Williams reference is to show that Williams' system,

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 121 of 182

Conducted on January 8, 2024

277 1 which is quite compatible with the Stuttard 2 system— and there are multiple reasons for a 3 person skilled in the art to look at their 4 combination, obviously — teaches that the memory 5 unit could store both types of command threads, 6 pixel command threads and vertex command threads, at 7 the same time. 8 And if somebody were to reap the 9 additional benefits of Williams' system, such as 10 using multiple buses — dedicated buses for multiple 11 types of command threads for the data transmission 12 to the processing blocks of Stuttard, it can also do 13 this through this incorporation of the graphics bus 14 scheduler. 15 But that's not required. I mean, you 16 don't have to include the graphics bus scheduler of 17 Williams into Stuttard for ground 2 because ground 2 18 is really — relates — the only part it's trying to 19 fill in with respect to ground 1 is the memory 20 element, not anything having to do with bus 21 scheduling or data different buses and so on. 22 Those benefits are there, and in my view, 23 they are very compatible, very much so. And I think 24 PTAB also acknowledges that because PTAB comments on 25 this and explains why they are compatible in spite	279 1 be that you don't make such a status assignment. 2 Everything is dynamically decided by the scheduler 3 itself, so although buses are dynamic, or on a 4 one-to-one basis assigned to parallel processing 5 blocks, what flows on those buses could be decided 6 by the scheduler itself dynamically at one time and 7 any hybrid combination of the two. 8 So many possibilities exist, and I don't 9 want to exclude one at the expense of the other. 10 Q. In Williams, the bus lines 116 are 11 dedicated to certain command types and those 12 mappings don't change over time; is that accurate? 13 A. In figure 4, that's -- 14 MR. JOHNSON: Objection. Form. 15 A. So in the embodiment of figure 4, it is 16 described that 116A is, I believe, a vertex command 17 bus and going to retiring the threads to be executed 18 to computational unit 402. But I think in figure 6 19 embodiment, the order is switched around in terms of 20 which one is vertex, which one is pixel. And then, 21 again, in figure 7, we go back to the same 22 assignment. 23 So it seems, if I remember exactly, and we 24 can check this, this assignment in Williams, it's 25 not a fixed assignment. They go back and forth on
278 1 of what the AMD declaration is. 2 So I think -- I agree with PTAB here that 3 Williams' system is completely compatible with 4 Stuttard. But again, the only use I'm making for 5 ground 2 of Williams is the memory element stored in 6 both types of pixel and vertex command threads as 7 taught with respect to input buffer 202. That's it. 8 Q. If a POSITA were to include the GBS from 9 Williams in conjunction with the input buffer and 10 use that in the Stuttard reference, would one also 11 include the type-specific bus lines 116 in that 12 modification? 13 MR. JOHNSON: Objection. Form. 14 A. I -- it's difficult to predict when you 15 try to incorporate -- "you" meaning -- sorry -- a 16 POSITA -- when tries to incorporate not only the 17 memory elements of Williams but also its graphics 18 bus scheduler and different buses into Stuttard, 19 what combination exactly they would come up with. 20 There are many possibilities. I don't want to 21 exclude any of them. 22 One possibility, certainly, is that you 23 preallocate some buses to some processing blocks for 24 pixel processing and some for graphics processing. 25 Statistically, another possibility would	280 1 it, but I think more important than any fixed 2 assignment is the notion that you could have 3 dedicated buses for different types of command 4 threads. 5 Now, exactly which one gets assigned to 6 which -- which command threads to which command bus 7 is probably less relevant than the fact that there 8 is an assignment from -- at any point in time during 9 the execution of processes and command threads in 10 Williams' system, there is an assignment of which 11 bus carries which type of data. Whether this could 12 be changed from one time instance to next time 13 instance, it's possible to do it. Yeah, I would 14 think. 15 Q. Are you aware of any disclosure in 16 Williams that describes the assignments of bus lines 17 116A to N with regard to what type of command they 18 carry changing over time? 19 A. Not that I recall, but again, if you -- I 20 want to check this again. So maybe we should check 21 it together. 22 The embodiment -- so remember: 116A is a 23 vertex command thread carrying bus in figure 4. In 24 figure 6 embodiment, if you could look at figure 6 25 embodiment -- or I should look it. I think it's a

Conducted on January 8, 2024

281 1 different one. Maybe I'm mistaken, but let me 2 check. 3 Q. You're welcome to check, but my question 4 isn't about different embodiments. It's about over 5 time in the same embodiment. So I just don't want 6 you to waste your time looking up something that is 7 nonresponsive to my question. So if your question 8 is about different embodiments having different 9 assignments, that's not what I'm asking about. 10 I'm asking about in one embodiment, the 11 assignment changing over time. 12 Are you aware of anything to that nature? 13 A. I acknowledge your question. Just 14 finishing what I was doing is that, in fact, I was 15 correct. In figure 6 embodiment, bus 116A is now 16 used to transfer texture-related, pixel-related 17 graphics commands, not vertex ones. So from one 18 embodiment to next embodiment, the assignment 19 changes. We both agree on that now. 20 The -- with respect to at any point in 21 time the assignment changing, I don't recall any 22 such disclosure in Williams. 23 Q. Why would it be beneficial to have 24 type-specific bus lines in Stuttard? 25 MR. JOHNSON: Objection. Form.	283 1 memory and the parallel processing blocks may not be 2 a good idea. I don't know. 3 But I've never opined in that context that 4 would be a good idea. It may or may not. It seems 5 little bit challenging to -- to -- on that front. 6 Q. Can you describe what does the GBS in 7 Williams do. 8 A. So different embodiments. It does 9 different things. But essentially, in the -- this 10 embodiment of figure 4, which I need to remind 11 myself. Let's look at it. 12 So the embodiment of figure 4. It's -- 13 the function is described on page 19 of Williams. 14 It says, "GBS 202 preferably assigns and/or source 15 the graphics command according to the type of 16 command. For example, geometry-type graphics 17 commands are assigned to a first bus and pixel-type 18 graphical commands are assigned to a second bus. 19 The plurality of buses 116A through 116N transfer 20 the multiple graphic commands between GBS 204 and 21 graphics subsystem 102 such that each of the 22 multiple graphics commands is transferred across its 23 assigned bus to a specific functional component of 24 graphics subsystem 102." And then, goes on to 25 provide those source code for an application showing
282 1 A. So please remember that my combination of 2 Stuttard and Williams, I'm not suggesting that we 3 necessarily adopt different buses carrying different 4 types of command into Stuttard. The only make -- 5 use I have of Williams in that combination is to 6 read on the memory element restoring both types of 7 pixel and vertex command threads. 8 Now, having said that, if the question is: 9 As a person skilled in the art looking at Stuttard 10 and Williams combination, would I also find benefits 11 to do dedicated bus assignment to different pixel 12 and vertex command threads? I would say as long as 13 this assignment is not statically done, it is 14 dynamically reconfigurable by scheduler or the host 15 system, then there are benefits because you go to 16 the appropriate portion of the memory and all the 17 appropriate threads are there. You pick them up and 18 you send them to the right subset of buses that you 19 know are the right ones at the time. 20 But because the nature of the applications 21 in Stuttard is more general -- it include 2D 22 graphics, 3D graphics, it includes general control 23 in addition within the 3D has pixel processing and 24 vertex processing and all that -- a fixed 25 high-priority assignment of buses between the cache	284 1 which of the open GL commands in -- in that 2 application are sent over which bus, depending on 3 their type. 4 Q. So when the GBS receives an instruction 5 from the input buffer, it's first determined what 6 type of command it relates to; is that fair? 7 A. Yes. 8 Q. The input -- withdrawn. 9 When the commands in Williams are placed 10 into the input buffer, they're not placed into any 11 specific memory or portion of memory according to 12 command type; right? 13 A. I disagree with this statement because it 14 implicitly assumes that the proportions of the 15 memory have to be continuous in a fixed location 16 inside memory itself. There's no such requirement. 17 The only thing that's required is there are two 18 different portions that may not be continuous and 19 that command threads go into one portion and pixel 20 threads into the other portion, which would make 21 sense to have it that way. So it -- so yeah. 22 Q. Dr. Pedram, I'm asking you about the 23 disclosure in Williams. In Williams, there's no 24 disclosure of anything putting commands into the 25 input buffer in any particular memory or portion of

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 123 of 182

Conducted on January 8, 2024

285 1 memory on the basis of the type of command that the 2 command is; correct or incorrect? 3 MR. JOHNSON: Objection. Form. 4 A. I disagree with this sentence -- 5 statement. 6 Q. Okay. Where in Williams is there any 7 disclosure of putting a command into the input 8 buffer into a particular location on the basis of 9 what type of command it is? 10 A. There is disclosure in Williams that 11 the -- through system bridge, these command threads 12 are written into the input buffer. These command 13 threads could be either pixel or vertex command 14 thread. They get stored there, obviously, in 15 different locations, and that's all we need to show; 16 that they get stored at the same time in the input 17 buffer of the Williams system. 18 If you're looking for the literal language 19 that there's two different portions in the input 20 buffer of Williams and one type of command thread 21 goes into one input buffer portion and the other one 22 goes into the other type, that language, literal 23 phrase, does not exist in Williams. 24 But a POSITA looking at Williams would 25 understand that there are two different portions of	287 1 saying, oh, this bit field says it's a vertex 2 command thread or pixel command thread. 3 Both possibilities exist. Williams is 4 silent on it. A POSITA would understand that, at 5 least, there are these two ways to determine what 6 the type of the command thread is. 7 Q. Your opinion is that Williams is silent on 8 how the GBS distinguishes the different types of 9 graphic commands. 10 Is that your testimony? 11 A. GBS knows the type of the command thread 12 that it sees. Exactly how that information is 13 encoded and relayed to the graphics bus scheduler, 14 at least, at this point, sitting here, I don't 15 remember. Williams discuss the exact mechanism by 16 which this type -- this information type is conveyed 17 to GBS. 18 Q. Let's go look at -- this is page 23 of 19 Williams. 20 Starting at line 13, Williams says, "GBS 21 204 can distinguish between the different types of 22 graphics command; e.g., graphics command pertaining 23 to geometry, graphics commands pertaining to texture 24 itself. Alternatively, a subsystem other than GBS 25 204 can distinguish between the different type of
286 1 the input buffer. One would be storing pixel 2 command threads. One would be storing the vertex 3 command threads. 4 Q. Is there any disclosure in Williams -- 5 well, withdrawn. 6 As the information is coming out of the 7 input buffer in Williams, does the GBS know what 8 type of information or command that is before it 9 analyzes it? 10 A. I don't understand the sentence "before it 11 analyzes it." 12 GBS knows the type of information 13 associated with the command thread, and that's how 14 it assigns it to the appropriate bus that it should 15 go on and get transmitted to the execution units. 16 So certainly, the graphics bus scheduler 17 knows about this. Now, how does it know about it? 18 There's information associated with the thread that 19 it is coming in or it could be implicit in how and 20 where it's stored on the input buffer. 21 On that question, Williams is silent, as 22 far as I can see, whether it can detect the type of 23 thread by the fact that where in the input buffer it 24 was stored at or that it is actually looking at the 25 particular bit field associated with the thread and	288 1 graphics commands and associate a descriptor with 2 each graphics command. GBS 204 can then distinguish 3 the type of graphics commands based on the commands 4 associated descriptor; that is GBS can use the 5 descriptor to make decisions." 6 You can agree that Williams discloses two 7 methods, at least, for the GBS to distinguish types 8 of graphics commands; one: Distinguish based on its 9 own analysis of the command, and two: It can 10 determine it based on a descriptor based with each 11 graphics command; correct? 12 A. That's correct, yes. Thank you for 13 refreshing my mind. But that's correct, yes. 14 Q. Okay. There's no disclosure in Williams 15 of GBS determining the type of graphics command 16 based on where it was located in the input buffer; 17 fair? 18 A. Not that I can see here. 19 Q. If a POSITA was looking at the Williams 20 system, why would he or she think to make the 21 modification such that the commands going into the 22 input buffer are placed into dedicated portions of 23 memory corresponding to what type of command they 24 are? 25 MR. JOHNSON: Objection. Form.

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 124 of 182

Conducted on January 8, 2024

289 1 A. Again, this would provide access 2 efficiency into the input buffer memory itself. 3 Now, this information may not be known or used by 4 the GBS, but still the efficiency factor is there. 5 It doesn't matter that the GBS knows about it or it 6 doesn't. 7 So if I've stored the pixel command 8 threads in consecutive spaces in the input buffer, 9 then I start accessing them, I would access them 10 sequentially, and that would make it more efficient 11 for me to bring those to GBS to do the -- its 12 analysis and send them on different paths, meaning 13 buses. 14 So the benefit is always there. Doesn't 15 matter if GBS is aware or unaware. 16 Q. Are you -- well, withdrawn. 17 Does Williams anywhere describe its input 18 buffer as anything other than transmitting to the 19 GBS in a FIFO manner? 20 A. I don't recall. 21 Please show me the portion of Williams 22 that you think provide support for this statement. 23 Q. I don't think there's any portion, so I'm 24 asking if you're aware of it. 25 A. The question makes a huge assumption that	291 1 Pixel command threads are considered to be 2 related because they involve the same kind of 3 processing. They get routed through the same set of 4 buses eventually -- we know that -- to the same 5 execution unit or units, and therefore, independent 6 of whether GBS is aware of this optimization or 7 makes use of it or not, the fact is when you store 8 the pixel command threads in one portion of the 9 memory to provide access efficiency for going 10 through the pixel command threads and you do the 11 same on the vertex command threads to provide access 12 latency benefits for accessing those vertex command 13 threads that's there, that's a benefit well known, 14 well practiced, well used in the state of the art. 15 A POSITA would know it. 16 It has nothing to go to do with the 17 function of the GBS, and I'm not sure why you are 18 trying to connect the two, implying that somehow 19 there's some kind of duplicative function or 20 redundancy in the system because in the -- in the 21 input buffer, there's this knowledge implicit in 22 where things sit, and in the GBS, there's the same 23 knowledge explicit because it's analyzed it. 24 I know that's okay. I don't see any 25 problem, any consistency, or any compatibility. The
290 1 the only way that the input buffer transmits 2 something to GBS is in a FIFO manner. I don't know 3 that. 4 Q. So my question is: If you have a unit, 5 such as the GBS, which is capable of determining 6 command time based on the command itself or a 7 descriptor associated with the command type, why 8 would anyone at the overhead required to segregate 9 memory in an attempt to use that information to know 10 what type of memory command is stored in that 11 portion? 12 Does that question make sense? I can 13 rephrase it. 14 A. No. I understand the question. I think 15 you're just missing the order of the component. 16 So the input buffer comes before the GBS. 17 The GBS has its own function to perform in terms of 18 doing analysis, of using this description 19 information to decide what to do with the command 20 threads it's received, where to send them off to. 21 But before GBS comes in, and consistent with the 22 functions performed by the GBS, is that this input 23 buffer -- and it's well known that there's major 24 source of efficiency in storing related command 25 threads in the same portion of an input buffer.	292 1 main reason for input buffer having two portions to 2 store this is access efficiencies associated with 3 that has nothing do with downstream what happens on 4 the buses or anything else, and that's why the two 5 are decoupled and the two optimizations could be 6 combined together to provide further efficiency both 7 on the access latency to the input buffer and on 8 complication latency and dedication of buses to 9 different types of command threads. 10 Q. So when you say the benefit is access 11 efficiency, access by whom or by what entity are you 12 talking about? 13 A. Both when you write into the portion and 14 when you read from the portion. So imagine on the 15 host interface trying to convey a bunch of pixel 16 commands or pixel command threads to input buffer. 17 I know they're all pixel command threads. I'm going 18 to write them one after the other with -- with a 19 stripe of four bytes. So starting address X. Write 20 the next hundred pixel command threads starting 21 address X up to X plus 400, or X plus 396 to be more 22 precise. 23 So I wrote a hundred of these pixel 24 command threads in the -- in the input buffer in 25 that way, and that's very easy. It happens because

Conducted on January 8, 2024

293 1 there's just a stripe. It's the four. I've always 2 known. No jump is needed. 3 On the other hand, when I'm on the GBS 4 side trying to look at the information, I go to the 5 portion of the memory, and I can start from there 6 and just read it. Maybe I don't know which portion 7 is which type of vertex or pixel command threads, 8 but I know there are two dedicated portions, and I 9 can go into one portion and read all of the pixel 10 command threads from there or vertex command threads 11 and do my own analysis and decide. 12 So there are access efficiencies on both 13 side. Certainly, there's access efficiency on the 14 host side. There can also be access efficiency on 15 the GBS side as reading it. 16 Q. Let's talk about on the host side. 17 You're saying it would be more efficient 18 for a host to be required to write only to specific 19 portions of memory as opposed to allowing to host to 20 write anywhere. 21 Is that what you're saying? 22 MR. JOHNSON: Object to form. 23 A. The term "to be required" is not what I'm 24 saying. What I'm saying is that a host, using the 25 host interface to write different types of command	295 1 evict it, but if it's not a cache, you have to be 2 careful. 3 So yes, subject to some second-order 4 events, which are important, you're right. 5 Q. Why would it be beneficial to add the 6 restriction that takes away that flexibility from 7 the host in the writing process? 8 A. I think I've answered the question. So 9 you are asking me to explain again and again, so 10 I'll -- 11 Q. I'm sorry. I don't -- 12 A. I really believe I have answered the 13 question. But anyway, quickly answering your 14 question. 15 Generally, it's well understood by a 16 person skilled in the art that flexibility and 17 efficiency are, sort of, two sides of the same coin, 18 but different sides of the same coin. 19 So there is a trade-off between the two. 20 The more flexible the system is, the less efficient 21 it becomes and vice versa, typically. And so, you 22 have to be careful to find the right balance between 23 flexibility and efficiency so the two don't go hand 24 to hand, and FPGA -- field-programmable gate 25 array -- gives you flexibility. You can program
294 1 threads, would find it beneficial to write command 2 threads of one type to one portion of the memory and 3 the command threads of the other type to a different 4 portion of the memory with the understanding that 5 command threads of the same type get executed, 6 typically, together one after the other and so on. 7 So having this kind of sequential access 8 to memory will have benefits when you try to execute 9 them. There are always exceptions to all of these; 10 right? So there could be exceptions, but we are -- 11 systems are architected to optimize for the common 12 case, and the common case would be that there is 13 benefit to write pixel command threads together in 14 one portion and vertex command threads together in 15 another portion; when I write them; when they are 16 accessed; when they are analyzed; when they are sent 17 over to the buses; when they are executed. I mean, 18 there are all these things that are the common-case 19 scenario that a system has optimized for. 20 Q. Without the condition that certain 21 portions of memory are dedicated to certain command 22 types, the host could write any commands anywhere in 23 memory; right? 24 A. As long as there is no valid data there 25 that you are overwriting. If it's a cache, you can	296 1 anything you want into it, but it's not efficient as 2 an application specification IC design because the 3 components are not going to be as fast or as densely 4 packed or as low power as the level (phonetic). 5 So this fundamental trade-off between 6 programmability and flexibility on one hand and 7 efficiency/performance on the other hand is a 8 fundamental trade-off that is present in all kinds 9 of system optimizations. 10 Q. Got it. 11 A. So your question presumes that they are 12 the same. They are different. They are different 13 and they are, in fact, working in different ways. 14 Q. I think we're talking about different 15 things. I'm talking about specifically in the 16 writing context. 17 In scenario 1, the host can write to any 18 memory allocation that's open. In scenario 2, the 19 host can only write to a memory allocation that is 20 open and is within a certain range. 21 Scenario 1 will be faster and more 22 efficient. Don't you agree? 23 A. No, I don't agree. 24 Q. Okay. So adding the constraint of only 25 being able to write to a particular memory

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 126 of 182

Conducted on January 8, 2024

297 1 allocation is increasing speed and efficiency, in 2 your view? 3 A. It can. It can. And if that means you're 4 writing all of the pixel command threads into one 5 portion, absolutely, it is higher efficiency. 6 Randomly writing these in different locations gives 7 you more flexibility, I agree, but it's not more 8 efficient. No, it's not. 9 Q. I didn't say "random." My question did 10 not say anything about randomness. 11 A. "Anywhere." You said "anywhere." 12 Q. My question -- my question was: In 13 scenario 1, the host can write anywhere it wants to, 14 provided that it's open, and scenario 2, it cannot 15 write anywhere it wants to. 16 You're saying scenario 2 is more efficient 17 and faster; correct? 18 A. Yes, I'm saying scenario 2 is more 19 efficient. Yes. 20 Q. Okay. All right. Let's take a ten-minute 21 break. 22 THE VIDEOGRAPHER: We are going off the 23 record. The time is 1:55 p.m. 24 (Recess.) 25 THE VIDEOGRAPHER: We are back on the	299 1 buffer, which is part of the memory device here in 2 deciding where to send the corresponding command 3 threads through different buses. 4 Q. Is there any discussion in Williams of an 5 arithmetic logic unit? 6 A. Is this with respect to a particular -- 7 Q. Yes. So claim 6 requires the command 8 processing engine have an ALU; right? Of the '053 9 patent. 10 A. I want to see what I've said with respect 11 to claim 6 in -- in -- on ground 2. 12 Q. So I'm not restricting my question to 13 ground 2. So let's at least look at ground 3, and 14 we can scroll there in your report. 15 A. Yeah. I'm looking at ground 2. If you 16 want to get to ground 3, we'll get there, but for 17 now, let's focus on ground 2. 18 And in ground 2, dependant claim 5, I 19 mentioned that the Stuttard discloses claim 6, not 20 Williams. So I've not made a statement about 21 Williams. 22 Q. Okay. So up on the screen is 23 paragraph 148 of your report, which is ground 3 of 24 claim 6. 25 And my question is whether Williams
298 1 record. The time on the video monitor is 2:07 p.m. 2 MR. DOKHANCHY: All right. Can we get 3 Exhibit 5 up, please. Sorry. Exhibit 6, rather. 4 BY MR. DOKHANCHY: 5 Q. I want to go to your opinions about figure 6 6, and in particular, you point to the output 7 buffer, Dr. Pedram, as being one of the memories 8 that can satisfy the '053 patent claims; is that 9 right? 10 A. I'm sorry. Are you waiting for me? 11 Q. Yes. Did you say something? 12 A. Yeah. I said yes. I agree with you. 13 Q. Okay. Does the GBS in Williams select 14 command threads from the output buffer? 15 A. No, it doesn't. 16 Q. Okay. And your view is that the GBS does 17 select command threads from the input buffer; is 18 that right? 19 A. No. I never said that GBS selects command 20 buffers from the input buffer. The function of the 21 selection is still met by the arbiter of Stuttard. 22 This is a combination. And what I identified as the 23 arbiter of Stuttard does the job. Graphics plus 24 scheduler provides some additional functionality 25 with respect to analyzing information in the input	300 1 discloses or discusses in any way an arithmetic 2 logic unit. 3 A. What I said in paragraph 148 of my report 4 is accurate any time you do, certainly, at least 5 geometry processing operations, transformations, for 6 example; right? 7 So a transformation is typically done by 8 taking a vector of coordinates -- X, Y, Z, for 9 example -- and multiplying it with the 10 transformation vector three by three that tells you 11 what's the new coordinate system that you're going 12 to get when you go from the user space to the screen 13 space -- the display space. And this kind of matrix 14 vector operation requires multiplication and 15 accumulation, which are things that can only be done 16 by ALUs. I mean, arithmetic logic units do this 17 kind of thing; right? 18 So you may not call it -- you may call it 19 anything else you like, but the fact is that you 20 will be performing arithmetic logic unit operations 21 when you do geometric calculations, such as 22 coordinate transformations. 23 It's not -- I mean, a POSITA would know 24 that. It's obvious, and it's necessarily, actually. 25 So Williams must necessarily include an arithmetic

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

Conducted on January 8, 2024

301 1 logic unit to do these; otherwise, system cannot 2 function. 3 Q. Okay. So is it true that a GPU cannot 4 perform geometry nor vertex operations without an 5 ALU? 6 A. Embedded inside its processing blocks, 7 whatever they're called, must be ALUs to do this. 8 There's no other option. 9 Q. Okay. Let's go -- so we touched on this 10 before, but I think maybe you had a different 11 question in mind. So I want to put this up on the 12 screen. So paragraph 133. 13 This is ground 3, which is the combination 14 of Williams and Whittaker in your report; correct? 15 A. Yes, of course. Stuttard also; right? 16 Q. Not in ground 3. Feel free to take a look 17 at your report if you want, but Stuttard is not a 18 part of ground 3. 19 Does that appear correct to you? 20 A. Yes, it does. 21 Q. Okay. So with regard to ground 3 and what 22 the arbiter is and what the selecting is, you point 23 to the fact that GBS can sort and assign graphics 24 commands to a plurality of buses; correct? 25 A. So we are talking about element 1B --	303 1 A. Just one correction to what you said in 2 debating the formula there. 3 So we said a command comes in -- there's a 4 lot of background noise. So -- so the statement you 5 made was that a command comes in and this happens 6 with respect to the command, which may be the case, 7 may be true, but it could also be that the way the 8 system is described and implemented, multiple 9 commands are there and then you look at them and 10 decide which one goes where. So it doesn't have to 11 be necessarily one command at that time on which 12 this action is being performed on GBS. 13 Q. Where does Williams describe more than one 14 command being operated on at the same time by the 15 GBS, as you've just described? 16 A. Yeah. It doesn't describe it one way or 17 the other. It doesn't say if the determination of 18 the command type is done as -- when one command 19 comes in or when there's a bunch of commands ready 20 and then it makes a determination. I think leaves 21 it to the reader or the implementer to do it 22 different ways. I mean, the fact remains that the 23 system -- the global bus scheduler will be looking 24 at a command and deciding what the type of command 25 is.
302 1 Q. Yes. 2 A. -- in paragraph 133? 3 Q. Correct. 4 Sir, do you need me to repeat question? 5 A. Yes. Please repeat the question. Thank 6 you. 7 Q. Okay. As to why Williams discloses 8 element 1B of the '053 patent, you state that the 9 selecting is done because the GBS can sort and 10 assign graphics commands to a plurality of buses; is 11 that right? 12 A. Yes. 13 Q. Okay. Is there anything else that you 14 contend in ground 3 that meets the selecting 15 requirement of 1B? 16 A. There is -- as you know, there is a 17 discussion about the priorities and so on in 18 Whittaker. But setting that aside with respect to 19 the selection action only, you are correct, yes. 20 Q. And the way it works is that a command 21 comes into the GBS, the GBS determines what type of 22 command it is, and then the GBS puts that command 23 onto the appropriate bus 116; is that right? 24 A. Correct. 25 Q. So we talked about --	304 1 But the question is: Do you have multiple 2 such commands waiting to be assigned? What buses 3 they to go on or no, there's never a queue of such 4 commands, and you just immediately decide as soon as 5 something comes in? That's the question, not that 6 somehow the -- from my perspective, somehow I'm 7 implying that the GBS is simultaneously, in same 8 instance in time, is looking at two different 9 command threads and making a determination. 10 Q. So in the possibility where the GBS is -- 11 I don't remember exactly how you phrased it, but 12 feel free to correct me. 13 In the situation where the GBS is not 14 simply assigning commands one by one as they come in 15 but rather from a group, what process does the GBS 16 use to select a particular command? 17 A. There are thread priorities; right? And 18 depending on what the priorities are, there might be 19 four different command threads. All of them -- say 20 of vertex command type that the GBS knows and which 21 one it releases to the bus to get executed next is 22 the selection that it makes. It's making a 23 selection based on the priority of the thread. So 24 that's what the selection is, on that basis, and 25 this is taught in Whittaker.

Conducted on January 8, 2024

305 1 Q. So we'll get to Whittaker. I'm just 2 asking about Williams. 3 A. Okay. So -- 4 Q. So in Williams, you say that there's a 5 possibility that the GBS chooses from one of several 6 command threads and then assigns that command thread 7 to the bus; correct? 8 A. Yes. 9 Q. So my question is: In Williams, in that 10 possibility, what decision protocol or procedure 11 does the GBS use to determine what command thread to 12 select in order to categorize input on the proper 13 bus? 14 MR. JOHNSON: Objection. Form. 15 A. So in a case in which a lot of command 16 threads are available of different types, but you 17 have a limited number of buses of two types 18 available to it, for example, there could be one 19 vertex command thread carrying bus available, but 20 there are three different command threads that are 21 available to be dispatched, at that point, 22 obviously, Williams' system have to choose one out 23 of three out of these command threads to dispatch, 24 and this could be based on a number of different 25 criteria. For example, first come, first serve, if	307 1 it could choose from, but how it decides which one 2 to select and assign to a given bus. 3 Is that what you're talking about? 4 Sorry. That's a yes? 5 A. Yes, that's correct. 6 Q. And is that the selecting that you say 7 meets the limitation of the claims of the '053 8 patent? 9 A. Yeah. When the selecting is the only 10 requirement, yes. When the selecting is based on 11 relative priorities, then I use Whittaker to 12 describe that part. 13 Q. Right. I'm just talking about selecting. 14 A. Yes. 15 Q. So in this scenario -- well, withdrawn. 16 I'll get to that a second. 17 Of those many possibilities that you 18 listed, are any of those possible selection 19 methodologies described in Williams? 20 A. Not that I recall off the top of my head. 21 Maybe there's something, but I don't recall anything 22 to that effect. 23 But again, this is really knowledge of 24 person skilled in the art at the time of the 25 invention. Whenever you have contention on a shared
306 1 there's anything like that, or if not, round robin, 2 or if not, based on affinity of a incoming thread to 3 already executing thread on the vertex processing 4 unit. Threads do have affinity to each other. So 5 think of it as similarity between the work that they 6 do and the data that they access and anything else. 7 I mean, so there's a wide number of 8 scheduling algorithms that are known to the person 9 skilled in the art at the time of invention that 10 would allow him or her to look at Williams' system, 11 realize because the resource is limited, maybe just 12 one vertex command carrying bus, the number of 13 available command threads are a lot more than that, 14 a selection has to be made, and therefore, Williams' 15 scheduler is doing that selection. 16 It's sort of obvious to a person skilled 17 in the art in view of Williams and the knowledge of 18 what's going on at the time in graphics processing 19 systems and, generally, in scheduling 20 multi-processing systems that these options exist, 21 and you could have used any one of them to decide -- 22 select. 23 Q. And just to be clear, you're talking about 24 the process by which the GBS would decide which 25 command thread out of the plurality of options that	308 1 resource in a multiprocessing system, that 2 contention will have to be resolved by a bus arbiter 3 or a selector or a scheduler that can use any of a 4 number of well-known selection mechanisms to decide 5 which one of the ready threads to assign to that 6 shared resource, which is under contention. That is 7 the bus. It's well known. It's well practiced. It 8 would have been easily recalled by any person 9 skilled in the art looking at Williams. 10 Q. And then, there's another scenario that 11 you said is possible, and that's that the command 12 threads come into to the GBS one by one, and as they 13 come in, the GBS simply sorts them on to the proper 14 bus; right? 15 A. It is possible for that scenario also, 16 although there is no disclosure that I recall in 17 Williams to say that. 18 On the other hand, if you know that 19 there's going to be contention on the shared 20 resource, the bus, the idea that as first come, 21 first serve, although it is one of the scheduling 22 algorithms, it's possible. It's one of the 23 scheduling algorithms. But although that exists, a 24 person skilled in the art would be discouraged from 25 employing it in a high-throughput system, such as

Conducted on January 8, 2024

309 1 Williams, whose goal is to be able to provide the 2 maximum efficiency in terms of the communication and 3 the execution of the tasks and threads in the 4 system, because a person skilled in the art would 5 know sometimes being a little bit more deliberate, 6 taking your time to look at things before you make a 7 decision, is going to give you better results. 8 So normally, systems like this, at least, 9 what they do is called "one of N" sort of type 10 decision-making, which means you wait for at least N 11 threads to become available and then you pick one. 12 Always wait for at least N threads. N could be two. 13 N could be four. N could be five. So at least wait 14 for that many threads before you make a decision 15 because that tends to avoid you making rash 16 decisions that we live to regret. That's much more 17 common strategy for scheduling. And I've used it in 18 many of my publications, this one-of-N policy. 19 Q. And in the scenario that we just 20 described, where the commands get categorized and 21 put onto buses one by one by the GBS, would there be 22 a set of commands to choose from such that the GBS 23 makes this selection of command threads or is it 24 only in the first method that we talked about where 25 there was a selection?	311 1 make a decision. Maybe there are five command 2 threads and you pick one from among five. Maybe 3 there are 20 command threads and you look at 20 and 4 decide, well, which one, or which ones, even. It 5 doesn't have to be just one. It depends on how many 6 resources you have. 7 So in that situation, even if I'm doing it 8 on a regular periodic basis, as I describe, a 9 polling base, there is still -- there is very much 10 the possibility and, in fact, a high probability 11 that multiple command threads are available to you 12 as the GBS unit, and you make a decision which one 13 to dispatch to the shared bus. 14 Q. Okay. I think I understand what you're 15 saying, but I think the last thing you said makes it 16 clear we're talking about two different scenarios. 17 You just said multiple command threads are available 18 to you. I'm talking about the scenario where only 19 one command thread is available to the GBS. 20 You agree, in that instance, there's no 21 selection for the GBS to do; right? 22 A. So this is an apparatus claim. So an 23 apparatus claim is important that the capability 24 exists for the system to look at multiple command 25 threads and select one from among -- multiple of
310 1 A. I mean, think about it in this way: 2 Again, such details are not provided in Williams for 3 obvious reasons because these are well known to a 4 person skilled in the art, but how does a GBS even 5 know that a thread has come in that it needs to look 6 at; right? 7 It can do it in one of two ways; right? 8 Polling-based mechanism, in which at regular 9 intervals in time, you look and see what's there and 10 you make a decision, or it could be done on an 11 interrupt basis, which means as soon as a thread 12 comes, it interrupts you and tells you, I'm here. 13 Make a decision; right? You may not be able to 14 service it immediately because you have other things 15 to do, but it told you on there. The polling-based 16 mechanism is typically done in these kind of 17 throughput-intensive applications. 18 Interrupt rev-1 (phonetic) is for reactive 19 systems, the control systems, typically. So for a 20 -- a polling-based mechanism, then at regular 21 intervals in time, maybe every -- I don't know -- 22 every hundred nanoseconds, every hundred clock 23 cycles, or every 20 clock cycles, or every five 24 clock cycles, you -- you look and see, oh, what's 25 there? Maybe there's one command thread and you	312 1 them or maybe more than one or select the substance 2 from the given set, a larger set, that I think 3 there's no doubt Williams has that capability and is 4 configured to do so. Now, if, in practice, the 5 situation is there's only one thread that comes in 6 once in a blue moon, there's no contention. 7 That thread that comes in, there's no -- 8 nothing to do; just okay, fine. I still assign you. 9 I assign you to go to run on this bus. But this is 10 a nominal assignment; that is there's no contention. 11 There is no need to really look inside. You just 12 say, Fine. You're here. Nobody else is asking me 13 for the resource you are asking for. Here, I give 14 it to you. 15 So still scheduler acts like I'm the king. 16 So I'm giving you the bus, but actually the bus is 17 there. Nobody is using it, so, of course, it's 18 going to give the bus to the requesting thread. So 19 in that case, yeah, there is only one thread 20 available. I just assign it to the bus that it 21 needs. That's it. Whether you call it "scheduling" 22 or not, I leave it to you. You decide. Or your 23 case can make -- your side can make a case one way 24 or the other. But from my perspective, the 25 selection is always happening.

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 130 of 182

Conducted on January 8, 2024

1 Sometimes it's straightforward selection 2 in the sense that nobody else is contending for the 3 same resource. So you give it to them, to whoever 4 is requesting it, that resource. Sometimes there's 5 contention and you need to make a selection. But 6 certainly, Williams' system is configured to make 7 that selection. 8 Q. I want to come back to a point that we 9 talked about earlier, and that is that your -- or, 10 at least, part of your reason for why you say one 11 would look at the input buffer in Williams and 12 modify it so that there are dedicated proportions 13 for each type of command thread. 14 Part of that motivation is access 15 efficiency on the readout side; correct? 16 A. On the write, but also readout side. Yes, 17 on both sides. 18 Q. So I just want to focus on the readout 19 side. 20 So when you're saying that it would make 21 for more efficient reads out of the input buffer, 22 what sorts of types of access -- withdrawn. Let me 23 rephrase the question clearly. 24 You know, what entities or what components 25 within Stuttard would benefit from such an	313 1 example, and in that case, if the four threads that 2 I'm waiting for come consecutively, of course, they 3 arrive faster, so I'm able to make that decision 4 faster, and that improves the throughput of the 5 system. 6 Again -- and I can go on and give you 7 other examples in which things could be very good 8 for Stuttard's system when it actually imports this 9 idea of storing pixel and vertex command threads 10 in -- in its cache memory and different portions; 11 right? Separate portions. 12 Q. So I want to go back to Stuttard. Let's 13 take a look at figure 4. 14 So earlier in the day, we talked about the 15 scheduler, and I asked you, you know, how does it 16 access the information in the cache? 17 Do you recall that? 18 A. Yes, sir. 19 Q. And you said, "Although it's not shown, a 20 POSITA would know that it has access to the cache"; 21 is that accurate? 22 A. First of all, certainly, scheduler has 23 information about the threads in the cache through 24 this status block, for example. You see white 25 directional arrows there, right, between status
314 1 efficiency? 2 MR. JOHNSON: Objection. Form. 3 A. Let's consider for simplicity, at least, 4 the case in which the input buffer and the cache, 5 the -- in Stuttard are the same and combined in a 6 way that implements also the functionality described 7 in Williams. 8 In that case, the fact that the threads 9 that need to be looked at by the thread manager, the 10 thread scheduler, and thread processors of Stuttard 11 are in one portion of the memory, it means that when 12 you try to access them to send them to the parallel 13 processing blocks of Stuttard, then there's access 14 efficiency. They're in approximate locations in the 15 memory. You can access them without using so-called 16 "random-access mechanism." You are doing access 17 through sequential means that improves access 18 efficiency. That's one level to response to your 19 question. 20 The other way I could respond to you, 21 which would -- may resonate a little bit better with 22 you -- I'm not sure -- is that if you look at the, 23 say, one of four policies for scheduling that I just 24 described -- so I wait for four threads to be there 25 before I can decide which one to dispatch, for	315 1 block and -- and the information that gets to the 2 thread processors and the scheduler and so on. 3 So -- so there is information there and there's 4 express disclosure that the scheduler makes use of 5 that information. So but in -- yeah, in just 6 trying, there's no direct arrow from scheduler going 7 to the cache. Fine. 8 So now you're question is... 9 Q. Well, so let's talk about that. 10 So the status block information. That is 11 two things; right? It is ready or yielding; 12 correct? 13 And what that tells -- sorry. I need a 14 verbal answer. 15 A. Yes. Yes, sir. But I did say yes also, 16 but for some reason it doesn't come through 17 sometimes. You're right. But yes. 18 Q. And so, that will tell the scheduler for 19 each of the -- eight, in this example -- eight 20 threads in the cache what are the eight statuses. 21 Is that accurate or is it something else? 22 A. No. If there are eight -- remember: This 23 shows a maximum of eight being there, ready, but 24 could be only two ready threads. So it doesn't mean 25 that because there are eight slots, there are eight 316

Conducted on January 8, 2024

317 1 threads ready to run at any point in time; right? 2 Q. I wasn't implying that. What my question 3 was is that this status block, 1030, informs the 4 schedule -- scheduler 1025 about the status of the, 5 in this example, eight threads in the cache, and 6 that status is ready or yielding; correct? 7 A. Right. But we also remember, 8 Mr. Dokhanchy, that in Stuttard, we did find clear 9 reference to high-priority versus low-priority 10 thread. So there is information about the priority 11 of the threads in addition to this ready versus 12 yielding. We couldn't track it to exactly which 13 field where stores that information, but we know 14 that information is being used by scheduler 1025 to 15 make decision -- to make decisions. There is 16 three-page discussion of that that we went through, 17 at least in part, earlier today. 18 But let's go with your hypothetical 19 situation, which there are apparently eight threads 20 ready and running -- ready to run in the cache of 21 1024, and their status information -- at least that 22 information, ready versus yielding, has been 23 provided to the scheduler. Go ahead then, please. 24 Q. And so, your contention is that the 25 scheduler, even though, as you said it's not clear	319 1 Stuttard that expressly states that. I can check 2 again, but my recollection is that I don't remember 3 something in Stuttard about the selection being 4 based on whether the command thread is pixel or 5 vertex command thread. It just looks at the 6 priority of the thread, looks at the ready versus 7 yielding status. It has its own policy, looks at 8 the status of the execution unit. It looks at 9 whether there's a halt instruction or not, whether 10 there's a 074 or note, and based on considerations 11 on this, it decides. 12 So again, if you go back to the bottom of 13 page 23 to middle of page 25 that we were looking at 14 in Stuttard earlier today, I don't see anything in 15 that part which describes determining which thread 16 should be active that relates to the type of thread, 17 hence my reasoning. 18 Q. When, if ever, does the scheduler 1025 19 learn what type of command thread it is operating 20 on? 21 MR. JOHNSON: Objection. Form. 22 A. I don't know the answer. I don't think 23 it's relevant to the claims or the prior art 24 analysis performed, but I don't know the answer, 25 when the scheduler will know the type of thread
318 1 how, knows the priority of thread in the cache; 2 correct? 3 A. That's what Stuttard says; right? 4 Q. Okay. 5 A. We saw it page 24, page 25. All of these 6 discussions about a thread with higher priority than 7 the current active thread means there's information 8 about priority, which is other than ready or 9 yielding. It's not the same as ready or yielding 10 information. 11 Q. Okay. Does the scheduler know the type of 12 command, vertex or pixel, for the threads that are 13 in the cache? 14 MR. JOHNSON: Objection. Form. 15 A. I don't recall. I -- I -- I would assume 16 it does, but its selection is not on the basis of -- 17 as far as I remember -- the selection it makes, it's 18 not on the basis of vertex versus pixel command 19 threads, but it does -- it should know or might 20 know. Let's put it this way: I just don't recall 21 way or the other. 22 Q. And why do you say that the selection by 23 the scheduler is not on the basis of vertex versus 24 pixel command threads? 25 A. Because I haven't seen anything in	320 1 being run. 2 I know Stuttard describes the type of 3 thread because it does clearly say there's a least 4 general control. There's 2D activity. There's 3D 5 activity and -- but -- but whether it knows about 6 vertex versus pixel and when it knows it, I'm not 7 sure. 8 MR. DOKHANCHY: All right. That's it for 9 me, pending any redirect and necessary follow-up. 10 So, Jeff, I don't know you're ready to go 11 now or you need time but... 12 MR. JOHNSON: Sorry. I didn't realize my 13 camera had been turned off. 14 Sure. Let me just ask a few things. 15 EXAMINATION 16 BY MR. JOHNSON: 17 Q. Dr. Pedram, where do you currently work? 18 A. University of Southern California. 19 Q. And what's your title? 20 A. Professor of electrical and computer 21 engineering. I'm also the Charles Lee Powell chair 22 in electrical engineering and computer sciences. 23 Q. And do you teach any classes? 24 A. Yes, I do. 25 Q. Can you -- do any of the sources that you

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

ATI Ex. 2007

IPR2023-00687

Page 132 of 182

Conducted on January 8, 2024

321 1 teach relate to the subject matter of the asserted 2 patents -- I'm sorry -- of the '053 patent? 3 A. Yes. I teach, and over the years, I've 4 taught many different courses. I regularly taught 5 courses as well as special topic classes. These 6 courses range from basic VLSI design that describes 7 how you put together a very large-scale integration 8 circuit or system at different levels for 9 undergraduates, for graduates, for PhD students. 10 There are three different courses there with a lot 11 of different topics covered, including parallel 12 processing, including graphics processing, including 13 all kinds of access to memory, DDR controllers, and 14 so on. 15 I also teach courses on switching theory, 16 on the computer ADD design, algorithms ranging from 17 system synthesis that includes architecture, design, 18 and optimizations all the way down to physical 19 design, placement routing, and so on. 20 I've taught special topic classes on 21 advanced processing architectures with a focus on 22 parallel processing and low-power design, low-power 23 methodologies as applied to such systems. 24 Q. Okay. Do you have a research group? 25 A. Yes, I do.	323 1 A. Yes. I mean, parallel processing, 2 graphics processing, scheduling, arbitration, 3 different types of memory being used, banking in 4 memories, accesses to buses; all kind of papers we 5 have written down on these things. 6 In fact, any one of the elements you pick 7 in claim 1 or claim 5 of -- I can give you papers 8 that we have published on that one word. I mean, 9 memory, arbiter, processing block, parallel 10 processing, tasks, threads. I mean, in everything, 11 we have been quite active. 12 Q. Have you ever been named as a committee or 13 program chair in any areas related to the technology 14 of the '053 patent? 15 A. I've served -- we don't even keep track of 16 these things, right, but I've served as the founding 17 technical chair and, later, general chair of a 18 symposium on low-power electronics design back in 19 1996. That's still going on. I'm the chair of the 20 executive committee. This group focuses on 21 low-power electronics, but in all types of systems, 22 including parallel processing systems and graphics 23 processing system, I have been the general co-chair 24 of the major conference in the circuits and systems 25 society of the --
322 1 Q. How large is it? 2 A. So it's been different sizes at different 3 times. I think at the largest, it was about 16 PhD 4 students. At the present time, I have 12 PhD 5 students and two postdoctoral associates and a 6 university research staff person assigned to work 7 with my group. So at this point, 15 people work in 8 my group. 9 My funding over the years has been robust 10 because of the contracts I've had and, typically, 11 you know, at universities, we pay for everything we 12 do; right? So I pay for these 12 graduate students 13 100K each a year; right? 14 Q. Do you and your group publish any 15 technical articles? 16 A. Yeah. We have published more than 800, I 17 think, papers in different conferences and journals, 18 including computer architecture classes -- 19 conferences, including design automation 20 conferences, machine learning conferences, operating 21 system-type classes -- conferences, and so on. 22 Yeah. 23 Q. Do any of those articles that you 24 published relate to the technology of the '053 25 patent?	324 1 (Reporter clarification.) 2 THE WITNESS: Circuits and system 3 symposium. It's called International Symposium of 4 Circuits and Systems. And I've served on many 5 technical program committee capacities, many 6 different conferences. Again, really don't keep 7 track of them for someone with my seniority. So 8 lots of different involvements. 9 BY MR. JOHNSON: 10 Q. Are you the named inventor on any patents? 11 A. Yes, I believe 11 or 12 assigned patents. 12 Maybe -- I think two are pending decision and some 13 of them do relate to the work that we did at the 14 time on backlight scaling and image enhancement, to 15 compensate for it and so on. So some of those 16 patents are directly related to the subject matter. 17 Q. And I think earlier today, you -- you 18 testified about a project that you did for the 19 United States military to develop a graphics 20 processing system. 21 Do you recall that? 22 A. Yes, I do. 23 Q. And how much money did the United States 24 award you for that project? 25 A. Dollars that are more valuable than today,

PLANET DEPOS

888.433.3767 | WWW.PLANETDEPOS.COM

325	327
1 but -- and I don't remember the exact amount, but 2 typically, when we get contracts like this, it's in 3 the order of 5, \$6 million. I believe that was the 4 number. My largest contract so far from a US 5 government agency has been about \$16 million, and 6 I've had three or four of them at that scale, large 7 scale. My group is very well funded, fortunately, 8 so far. 9 Q. Give me just a minute. 10 If you could look back at Exhibit 1 and go 11 to paragraph 41. 12 A. Exhibit 1 is my declaration? 13 Q. I believe it is. 14 DEPOSITION TECHNICIAN: Counsel, do you 15 want that on the screen? 16 MR. JOHNSON: No. It's unnecessary. 17 A. Okay. Where do I go? 18 Q. Paragraph 41. 19 A. Yes. 20 Q. In paragraph 41, you identify a 21 description of the "person of ordinary skill in the 22 art"; is that correct? 23 A. Definitely. 24 Q. And is it your opinion that you were, at 25 least, a person of ordinary skill in the art,	1 ERRATA SHEET FOR THE DEPOSITION OF: 2 3 CASE NAME: REALTEK SEMICONDUCTOR CORP. V ATI TECHNOLOGIES ULC 4 5 CASE NUMBER: IPR2023-00564 6 7 U.S. PATENT NO. 7,742,053 8 9 DEPO DATE: MONDAY, JANUARY 8, 2024 10 11 DEPONENT: MASSOUD PEDRAM, PH.D. 12 13 CORRECTIONS 14 PG. LN. NOW READS SHOULD READ REASONS 15 _____ 16 _____ 17 _____ 18 _____ 19 _____ 20 _____ 21 _____ 22 _____ 23 _____ 24 25 SIGNATURE _____ DATE _____
326	328
1 according to that description in paragraph 41, as of 2 the time of the invention in the '053 patent? 3 A. Yes, sir, I am. 4 Q. Okay. I don't have any other questions. 5 MR. DOKHANCHY: Nothing further from me. 6 Thank you, Dr. Pedram. 7 MR. JOHNSON: All right. 8 THE WITNESS: Thank you. 9 THE VIDEOGRAPHER: This marks the end of 10 the deposition of Dr. Massoud Pedram. We're going 11 off the record. The time is 3:02 -- 3:03 p.m. 12 Pacific Standard. 13 (Deposition concluded at 3:03 P.M.) 14 -oOo- 15 16 17 18 19 20 21 22 23 24 25	1 STATE OF CALIFORNIA) 2) SS. 3 COUNTY OF LOS ANGELES) 4 5 I, MASSOUD PEDRAM, PH.D, HEREBY CERTIFY 6 UNDER PENALTY OF PERJURY UNDER THE LAWS OF THE 7 STATE OF CALIFORNIA THAT THE FOREGOING IS TRUE AND 8 CORRECT. 9 EXECUTED THIS ____ DAY OF _____, 10 2024, AT _____, CALIFORNIA. 11 12 13 14 15 _____ 16 MASSOUD PEDRAM, PH.D 17 18 19 20 21 22 23 24 25

Transcript of Massoud Pedram, Ph.D., Volume II
Conducted on January 8, 2024

40 (329 to 332)

329 1 CERTIFIED STENOGRAPHER'S CERTIFICATE 2 STATE OF CALIFORNIA)) SS. 3 COUNTY OF LOS ANGELES) 4 5 I, NATALIE PARVIZI-AZAD, HERBY CERTIFY: 6 I AM A DULY QUALIFIED CERTIFIED SHORTHAND 7 REPORTER IN THE STATE OF CALIFORNIA, HOLDER OF 8 CERTIFICATE NUMBER CSR 14125 ISSUED BY THE COURT 9 REPORTERS BOARD OF CALIFORNIA AND WHICH IS IN FULL 10 FORCE AND EFFECT. (BUS. & PROF. § 8016) 11 I AM NOT FINANCIALLY INTERESTED IN THIS 12 ACTION AND NOT A RELATIVE OR EMPLOYEE OF ANY 13 ATTORNEY OF THE PARTIES, OR OF ANY OF THE PARTIES. 14 (CIV. PROC. § 2025.320(A)) 15 I AM AUTHORIZED TO ADMINISTER OATHS OR 16 AFFIRMATIONS PURSUANT TO CALIFORNIA CODE OF CIVIL 17 PROCEDURE, SECTION 2093 (B) AND PRIOR TO BEING 18 EXAMINED, THE DEPONENT WAS FIRST PLACED UNDER OATH 19 OR AFFIRMATION BY ME. (CIV. PROC. §§ 2025.320, 20 2025.540(A)) 21 I AM THE CERTIFIED OFFICER THAT 22 STENOGRAPHICALLY RECORDED THE TESTIMONY IN THE 23 FOREGOING PROCEEDING AND THE FOREGOING TRANSCRIPT 24 IS A TRUE RECORD OF THE TESTIMONY GIVEN. (CIV. 25 PROC. § 2025.540(A))	
330 1 I HAVE NOT, AND SHALL NOT, OFFER OR PROVIDE 2 ANY SERVICES OR PRODUCTS TO ANY PARTY'S ATTORNEY 3 OR THIRD PARTY WHO IS FINANCING ALL OR PART OF THE 4 ACTION WITHOUT FIRST OFFERING SAME TO ALL PARTIES 5 OR THEIR ATTORNEYS ATTENDING THE PROCEEDING AND 6 MAKING SAME AVAILABLE AT THE SAME TIME TO ALL 7 PARTIES OR THEIR ATTORNEYS. (CIV. PROC. § 8 2025.320(B)) 9 I SHALL NOT PROVIDE ANY SERVICE OR PRODUCT 10 CONSISTING OF THE CERTIFIED STENOGRAPHER'S 11 NOTATIONS OR COMMENTS REGARDING THE DEMEANOR OF 12 ANY WITNESS, ATTORNEY, OR PARTY PRESENT AT THE 13 PROCEEDING TO ANY PARTY OR ANY PARTY'S ATTORNEY OR 14 THIRD PARTY WHO IS FINANCING ALL OR PART OF THE 15 ACTION, NOR SHALL I COLLECT ANY PERSONAL 16 IDENTIFYING INFORMATION ABOUT THE WITNESS AS A 17 SERVICE OR PRODUCT TO BE PROVIDED TO ANY PARTY OR 18 THIRD PARTY WHO IS FINANCING ALL OR PART OF THE 19 ACTION. (CIV. PROC. § 2025.320(C)) 20 1/11/2024 21 DATED: _____ 22 23  24 _____ NATALIE PARVIZI-AZAD, CSR NO.14125 25	

A			
a)	291:12	312:16, 315:8	232:21, 235:22,
329:15, 329:21,	according	add	236:6, 238:8,
329:26	253:10, 283:15,	184:18, 268:4,	240:10, 240:11,
ability	284:11, 326:1	271:14, 271:20,	292:18, 294:6
214:16, 215:6	accumulation	272:7, 272:12,	again
able	300:15	272:23, 273:1,	182:9, 184:1,
207:16, 214:25,	accurate	273:3, 273:5,	185:5, 186:5,
217:23, 221:10,	179:12, 267:19,	274:5, 295:5,	187:6, 195:6,
221:17, 229:25,	279:12, 300:4,	321:16	197:10, 198:22,
230:1, 235:18,	315:21, 316:21	added	199:8, 199:13,
296:25, 309:1,	acknowledge	199:10, 272:17,	199:20, 202:3,
310:13, 315:3	204:14, 281:13	274:4, 275:16	206:25, 209:18,
above	acknowledges	adding	216:5, 244:3,
190:2, 190:5,	275:11, 277:24	296:24	246:15, 253:4,
253:18	across	addition	253:14, 263:7,
absolutely	261:25, 283:22	216:22, 273:6,	264:1, 264:8,
187:19, 297:5	action	276:17, 282:23,	269:5, 273:1,
abstract	180:2, 196:17,	317:11	273:6, 276:17,
210:11, 210:12,	247:11, 302:19,	additional	278:4, 279:21,
212:2	303:12, 329:13,	268:9, 271:9,	280:19, 280:20,
access	330:4, 330:15,	273:24, 277:9,	289:1, 295:9,
221:21, 242:23,	330:19	298:24	307:23, 310:2,
248:16, 259:13,	activate	address	315:6, 319:2,
259:17, 259:19,	241:21, 249:16,	258:6, 274:18,	319:12, 324:6
259:23, 260:2,	251:5, 251:6	274:19, 274:24,	agency
273:15, 274:9,	activates	275:3, 275:4,	184:21, 185:6,
274:17, 275:9,	256:1	275:5, 292:19,	325:5
275:13, 289:1,	active	292:21	agree
289:9, 291:9,	242:8, 248:5,	addressing	190:14, 194:10,
291:11, 292:2,	248:6, 248:11,	189:7	195:16, 210:5,
292:7, 292:10,	249:25, 252:5,	adjustment	211:21, 231:8,
292:11, 293:12,	252:10, 254:14,	180:18, 180:20	273:4, 278:2,
293:13, 293:14,	254:23, 255:25,	administer	281:19, 288:6,
294:7, 306:6,	318:7, 319:16,	329:16	296:22, 296:23,
313:14, 313:22,	323:11	administrative	297:7, 298:12,
314:12, 314:13,	activity	176:22	311:20
314:15, 314:16,	245:20, 245:21,	adopt	agreed
314:17, 315:16,	252:12, 320:4,	282:3	194:23, 268:19
315:20, 321:13	320:5	advanced	agreeing
access-latency	acts	184:21, 184:23,	216:18
274:14	312:15	185:5, 321:21	ahead
accessed	actual	affinity	187:1, 202:4,
294:16	182:3	306:2, 306:4	317:23
accesses	actually	affirmation	ai
323:4	182:10, 186:21,	329:20	186:25
accessing	190:17, 195:2,	affirmations	algorithms
274:25, 289:9,	235:4, 261:3,	329:17	306:8, 308:22,
	286:24, 300:24,	after	308:23, 321:16
		232:13, 232:16,	

alj 189:7, 189:15, 192:15, 193:19, 195:22, 205:3 alj's 194:10 all 177:13, 180:5, 184:1, 186:17, 187:1, 187:7, 187:21, 188:16, 189:2, 193:11, 195:15, 198:2, 206:9, 210:11, 211:9, 216:19, 216:22, 220:18, 220:25, 221:11, 223:3, 223:16, 226:15, 226:16, 229:22, 230:1, 230:13, 231:8, 234:1, 234:2, 236:7, 236:15, 238:2, 238:13, 239:14, 239:21, 239:25, 240:24, 243:16, 243:17, 244:20, 245:6, 245:19, 245:21, 246:11, 249:2, 251:24, 257:18, 257:19, 258:1, 260:5, 264:12, 265:25, 266:1, 267:16, 271:5, 273:18, 274:10, 274:22, 275:8, 282:16, 282:24, 285:15, 292:17, 293:9, 294:9, 294:18, 296:8, 297:4, 297:20, 298:2, 304:19, 315:22, 318:5, 320:8, 321:13, 321:18, 323:4, 323:21, 326:7, 330:3, 330:4,	330:6, 330:14, 330:18 allege 266:21 alleged 267:25 allocation 231:19, 296:18, 296:19, 297:1 allow 180:12, 182:6, 235:10, 246:21, 261:3, 273:7, 276:7, 306:10 allowing 293:19 allows 240:6 alluded 194:25 almost 183:15 along 263:20 already 188:10, 244:21, 246:10, 247:19, 248:2, 268:19, 306:3 also 175:24, 177:25, 180:17, 181:2, 184:3, 185:4, 186:24, 187:7, 191:24, 197:4, 199:3, 211:6, 224:23, 224:24, 234:8, 235:24, 237:22, 247:5, 247:21, 250:5, 254:20, 258:23, 259:24, 260:16, 261:21, 262:20, 263:18, 264:13, 272:2, 272:22, 276:5, 277:12, 277:24, 278:10, 278:17, 282:10,	293:14, 301:15, 303:7, 308:15, 313:16, 314:6, 316:15, 317:7, 320:21, 321:15 alternative 194:22, 195:9 alternatively 287:24 although 184:14, 194:14, 198:15, 260:12, 260:20, 262:10, 270:9, 279:3, 308:16, 308:21, 308:23, 315:19 alu 299:8, 301:5 alus 300:16, 301:7 always 194:13, 195:14, 196:3, 218:21, 218:22, 218:24, 253:18, 263:1, 275:8, 289:14, 293:1, 294:9, 309:12, 312:25 amd 187:14, 187:17, 188:23, 207:22, 208:2, 239:12, 262:18, 278:1 amd's 178:15 among 190:9, 190:19, 191:25, 197:14, 311:2, 311:25 amount 325:1 analysis 209:23, 288:9, 289:12, 290:18, 293:11, 319:24 analyzed 208:16, 291:23, 294:16	analyzes 286:9, 286:11 analyzing 298:25 angeles 173:22, 174:23, 177:1, 328:3, 329:4 angle 184:7, 234:9, 235:2, 235:6, 261:21 another 197:19, 199:4, 220:12, 232:18, 237:3, 247:10, 247:18, 248:25, 252:17, 252:20, 254:20, 272:7, 273:19, 278:25, 294:15, 308:10 answer 190:13, 191:10, 199:1, 201:23, 202:2, 202:4, 202:5, 204:5, 204:19, 212:8, 239:15, 239:16, 240:1, 252:1, 253:7, 264:23, 265:15, 265:17, 266:13, 269:17, 316:14, 319:22, 319:24 answered 251:2, 253:16, 295:8, 295:12 answering 227:18, 295:13 answers 254:11 antenna 184:25 any 178:19, 179:11, 179:16, 183:18, 185:25, 186:3, 187:8, 194:5,
--	---	---	---

194:12, 195:19, 198:15, 198:24, 200:2, 203:11, 207:18, 207:21, 208:15, 209:22, 212:14, 213:23, 219:24, 230:21, 236:8, 236:9, 244:18, 246:19, 247:2, 247:6, 249:25, 251:8, 252:15, 255:22, 266:2, 267:10, 267:24, 268:13, 269:14, 271:10, 272:4, 273:23, 275:18, 278:21, 279:7, 280:1, 280:8, 280:15, 281:20, 281:21, 284:10, 284:25, 285:6, 286:4, 289:23, 291:24, 291:25, 294:22, 296:17, 299:4, 300:1, 300:4, 306:21, 307:18, 308:3, 308:8, 317:1, 320:9, 320:23, 320:25, 322:14, 322:23, 323:6, 323:13, 324:10, 326:4, 329:13, 329:14, 330:2, 330:9, 330:12, 330:13, 330:15, 330:17 anyone 290:8 anything 178:20, 199:16, 200:6, 200:13, 202:14, 203:11, 203:21, 204:6, 209:22, 248:20, 255:1, 262:7, 268:16, 277:20, 281:12, 284:24,	289:18, 292:4, 296:1, 297:10, 300:19, 302:13, 306:1, 306:6, 307:21, 318:25, 319:14 anyway 246:18, 248:20, 252:8, 295:13 anywhere 289:17, 293:20, 294:22, 297:11, 297:13, 297:15 apparatus 256:13, 311:22, 311:23 apparently 198:17, 317:19 appeal 173:2, 174:2 appear 301:19 appears 231:14 application 215:10, 215:11, 233:17, 233:25, 236:23, 265:5, 266:8, 283:25, 284:2, 296:2 applications 248:15, 248:17, 266:1, 282:20, 310:17 applied 188:12, 190:6, 192:10, 193:4, 205:7, 321:23 applies 218:4 apply 189:24, 190:12, 190:22 appreciate 191:10 appropriate 193:6, 193:7, 202:20, 219:9,	240:7, 264:16, 282:16, 282:17, 286:14, 302:23 appropriately 234:6 approximate 314:14 aratus 209:4 arbiter 196:6, 196:10, 196:18, 196:24, 197:3, 197:13, 197:16, 197:20, 197:25, 198:4, 199:22, 200:5, 200:14, 201:10, 201:12, 201:20, 201:24, 202:7, 202:10, 202:15, 202:21, 202:24, 203:16, 203:17, 203:23, 203:24, 203:25, 204:2, 204:3, 204:7, 204:10, 204:15, 204:18, 204:21, 205:8, 205:12, 205:13, 205:17, 206:10, 206:19, 207:7, 207:11, 208:22, 209:7, 209:12, 211:3, 211:7, 211:17, 212:13, 212:14, 213:15, 213:22, 214:16, 214:20, 214:25, 215:3, 215:7, 215:14, 215:16, 215:20, 215:22, 216:1, 216:14, 216:23, 217:13, 217:20, 218:13, 218:19, 219:6, 219:20, 240:15, 240:21, 259:6, 259:9, 259:11, 259:19,	260:2, 260:9, 266:22, 267:7, 267:11, 267:15, 267:17, 267:25, 268:8, 268:15, 268:22, 269:11, 269:12, 269:16, 298:21, 298:23, 301:22, 308:2, 323:9 arbitrates 242:12 arbitration 323:2 architected 294:11 architecture 181:4, 184:12, 184:23, 185:2, 185:19, 238:17, 271:10, 272:18, 321:17, 322:18 architectures 181:8, 321:21 areas 323:13 argued 195:2, 195:21 argument 208:2 arguments 194:22, 195:9 arises 191:1 arithmetic 299:5, 300:1, 300:16, 300:20, 300:25 army 179:21, 183:12 army-related 181:21 around 251:12, 279:19 array 239:2, 241:21, 246:23, 250:2, 250:13, 295:25
---	--	---	--

arrive 205:18, 315:3 arrived 204:3, 204:4, 205:9, 205:14 arrives 198:7, 199:15, 200:20, 202:9 arrow 316:6 arrows 315:25 art 191:2, 191:8, 197:2, 200:2, 204:22, 205:10, 208:14, 208:15, 208:23, 238:14, 260:25, 261:23, 261:24, 262:25, 263:3, 268:10, 273:17, 277:3, 282:9, 291:14, 295:16, 306:9, 306:17, 307:24, 308:9, 308:24, 309:4, 310:4, 319:23, 325:25 art" 325:22 art's 205:4 articles 322:15, 322:23 aside 302:18 asked 184:14, 191:9, 202:5, 252:1, 259:9, 266:14, 315:15 asking 192:2, 201:21, 202:23, 209:20, 253:15, 254:12, 264:10, 265:13, 265:24, 281:9, 281:10, 284:22,	289:24, 295:9, 305:2, 312:12, 312:13 aspects 186:3 assemble 246:22, 250:12 assembly 180:6 asserted 176:16, 208:12, 217:7, 321:1 assertive 191:3 assign 219:6, 219:8, 301:23, 302:10, 307:2, 308:5, 312:8, 312:9, 312:20 assigned 230:12, 231:24, 232:4, 256:15, 279:4, 280:5, 283:17, 283:18, 283:23, 304:2, 322:6, 324:11 assigning 197:21, 304:14 assignment 232:11, 261:25, 279:1, 279:22, 279:24, 279:25, 280:2, 280:8, 280:10, 281:11, 281:18, 281:21, 282:11, 282:13, 282:25, 312:10 assignments 280:16, 281:9 assigns 283:14, 286:14, 305:6 associate 288:1 associated 205:2, 286:13, 286:18, 286:25,	288:4, 290:7, 292:2 associates 322:5 assume 243:4, 255:14, 257:9, 274:23, 318:15 assumes 284:14 assumption 203:9, 203:10, 289:25 assumptions 203:1, 204:1 ati 173:11, 174:11, 175:13, 177:7, 177:19, 327:3 attached 255:14 attempt 290:9 attending 177:14, 330:5 attorney 177:17, 329:14, 330:2, 330:12, 330:13 attorneys 330:5, 330:7 audio 227:17 augmented 182:21 authorized 329:16 automatic 180:17 automation 322:19 availability 197:15, 198:4 available 188:13, 197:1, 213:23, 232:9, 236:4, 241:11, 243:21, 254:4,	255:16, 305:16, 305:18, 305:19, 305:21, 306:13, 309:11, 311:11, 311:17, 311:19, 312:20, 330:6 avoid 236:17, 309:15 award 324:24 aware 192:3, 193:5, 280:15, 281:12, 289:15, 289:24, 291:6 away 295:6 B b 329:18 b) 330:8 b2 176:26 b7 241:18 back 179:20, 179:23, 195:25, 199:4, 208:19, 212:9, 220:23, 223:3, 225:23, 228:25, 251:10, 257:5, 257:24, 258:2, 258:21, 258:24, 279:21, 279:25, 297:25, 313:8, 315:12, 319:12, 323:18, 325:10 background 179:15, 197:24, 247:14, 259:25, 303:4 backlight 180:19, 180:23, 186:10, 186:13, 324:14
---	--	---	--

backlighted 183:4 backlights 180:18 balance 295:22 banking 323:3 base 311:9 based 189:20, 194:21, 197:14, 198:3, 207:9, 214:23, 235:22, 236:5, 237:8, 242:13, 248:1, 249:12, 250:8, 288:3, 288:8, 288:10, 288:16, 290:6, 304:23, 305:24, 306:2, 307:10, 319:4, 319:10 basic 239:11, 321:6 basis 252:25, 279:4, 285:1, 285:8, 304:24, 310:11, 311:8, 318:16, 318:18, 318:23 beam-forming 185:1 bear 227:20 beat 215:3 because 198:13, 198:18, 199:21, 201:15, 204:12, 204:21, 208:9, 208:25, 209:21, 211:16, 218:1, 229:20, 230:3, 230:20, 235:1, 235:6, 235:18, 235:25, 236:2, 239:21,	242:24, 243:20, 244:20, 248:8, 248:15, 248:18, 253:15, 254:9, 255:17, 257:3, 257:13, 260:22, 266:24, 270:20, 272:17, 273:9, 274:5, 274:9, 276:6, 277:17, 277:24, 282:15, 282:20, 284:13, 291:2, 291:20, 291:23, 292:25, 296:2, 302:9, 306:11, 309:4, 309:15, 310:3, 310:14, 316:25, 318:25, 320:3, 322:10 become 254:4, 257:11, 309:11 becomes 236:4, 295:21 been 186:2, 186:16, 188:10, 194:11, 195:14, 196:3, 203:21, 208:7, 229:8, 238:9, 238:19, 239:1, 239:21, 246:10, 261:20, 262:10, 262:25, 308:8, 317:22, 320:13, 322:2, 322:9, 323:11, 323:12, 323:23, 325:5 before 173:2, 174:2, 174:25, 178:9, 188:20, 203:6, 206:18, 217:18, 230:25, 231:11, 231:15, 233:10, 235:16, 235:24, 236:8, 236:9,	239:25, 240:9, 271:14, 272:8, 275:9, 286:8, 286:10, 290:16, 290:21, 301:10, 309:6, 309:14, 314:25 beginning 174:23, 274:24 begins 177:4 behalf 174:21, 177:19, 177:21 behind 235:3, 260:20 being 189:8, 191:5, 195:19, 208:12, 218:3, 225:9, 226:13, 227:17, 228:10, 228:18, 237:4, 237:5, 237:8, 237:19, 244:6, 244:9, 245:4, 254:7, 254:16, 255:16, 265:7, 266:12, 296:25, 298:7, 303:12, 303:14, 309:5, 316:23, 317:14, 319:3, 320:1, 323:3, 329:18 believe 181:22, 188:25, 191:2, 191:7, 195:5, 210:3, 212:24, 219:15, 231:22, 251:2, 262:18, 279:16, 295:12, 324:11, 325:3, 325:13 below 225:20 beneficial 274:5, 274:8, 281:23, 294:1,	295:5 benefit 289:14, 291:13, 292:10, 294:13, 313:25 benefits 272:16, 273:16, 273:24, 275:13, 275:14, 275:18, 277:9, 277:22, 282:10, 282:15, 291:12, 294:8 best 180:1, 193:9 better 179:3, 231:7, 309:7, 314:21 between 188:23, 199:6, 206:19, 215:7, 224:24, 239:12, 243:17, 250:6, 263:19, 276:2, 282:25, 283:20, 287:21, 287:25, 295:19, 295:22, 296:5, 306:5, 315:25 beyond 229:24 bidding 235:9 big 179:21, 228:22 binatization 225:4, 225:5, 225:20, 231:11, 231:15, 232:13, 232:17, 235:21, 236:4, 236:9, 236:16, 236:20 binning 225:5, 227:22, 229:18, 230:4, 230:25, 231:3, 235:14 bins 225:9
---	--	---	--

bintization 231:16	251:16, 260:15, 276:10, 276:13, 276:14, 277:12, 278:23, 279:5, 283:1, 301:6, 314:13	271:24, 273:8, 276:3, 285:11	bus 197:18, 197:19, 270:20, 270:23, 273:11, 273:13, 275:23, 276:5, 276:14, 276:16, 277:13, 277:16, 277:20, 278:11, 278:18, 279:10, 279:17, 280:6, 280:11, 280:16, 280:23, 281:15, 281:24, 282:11, 283:17, 283:18, 283:23, 284:2, 286:14, 286:16, 287:13, 302:23, 303:23, 304:21, 305:7, 305:13, 305:19, 306:12, 307:2, 308:2, 308:7, 308:14, 308:20, 311:13, 312:9, 312:16, 312:18, 312:20, 329:11
bit 179:3, 179:4, 179:15, 184:18, 192:6, 196:15, 214:15, 238:3, 252:6, 263:6, 264:9, 266:23, 283:5, 286:25, 287:1, 309:5, 314:21	blue 312:6	brightness 240:8, 261:19	
block 198:8, 199:16, 199:17, 199:24, 199:25, 200:12, 200:15, 200:21, 200:22, 201:2, 201:8, 201:9, 201:12, 202:20, 226:15, 226:19, 226:23, 230:12, 231:24, 232:3, 232:5, 241:17, 243:15, 243:22, 243:23, 245:4, 249:20, 254:7, 254:18, 255:9, 259:13, 259:15, 276:17, 315:24, 316:1, 316:10, 317:3, 323:9	board 173:2, 174:2, 188:10, 329:10	bring 261:5, 289:11	
blocks 199:23, 202:19, 223:11, 224:3, 224:19, 224:23, 225:10, 226:16, 226:24, 227:3, 228:6, 228:16, 230:13, 230:14, 230:17, 230:22, 231:20, 232:7, 232:9, 232:12, 237:23, 239:3, 239:4, 244:6, 244:8, 245:9, 245:12, 245:14, 247:4, 251:7,	bodement 206:23	broadest 209:1	
	body 269:19	buffer 270:14, 270:16, 270:20, 270:22, 271:7, 271:9, 271:14, 271:16, 271:20, 271:23, 272:6, 272:7, 272:8, 272:10, 272:11, 272:14, 272:24, 273:7, 273:9, 273:20, 274:4, 274:5, 274:13, 275:16, 275:20, 276:8, 276:9, 278:7, 278:9, 284:5, 284:10, 284:25, 285:8, 285:12, 285:17, 285:20, 285:21, 286:1, 286:7, 286:20, 286:23, 288:16, 288:22, 289:2, 289:8, 289:18, 290:1, 290:16, 290:23, 290:25, 291:21, 292:1, 292:7, 292:16, 292:24, 298:7, 298:14, 298:17, 298:20, 299:1, 313:11, 313:21, 314:4	buses 197:21, 270:25, 271:1, 273:13, 274:11, 276:9, 276:13, 277:10, 277:21, 278:18, 278:23, 279:3, 279:5, 280:3, 282:3, 282:18, 282:25, 283:19, 289:13, 291:4, 292:4, 292:8, 294:17, 299:3, 301:24, 302:10, 304:2, 305:17, 309:21, 323:4
	both 201:11, 210:6, 213:10, 213:11, 213:24, 214:17, 216:2, 216:10, 223:13, 224:7, 225:15, 238:22, 251:14, 251:25, 270:11, 270:18, 271:17, 271:18, 272:19, 277:5, 278:6, 281:19, 282:6, 287:3, 292:6, 292:13, 293:12, 313:17		busy 253:21, 254:2
	bottom 189:15, 191:13, 192:15, 246:19, 252:3, 270:25, 271:3, 319:12	build 184:22	butler 175:25, 177:13
	branch 246:1, 246:2	built 182:23, 185:8	bytes 275:2, 292:19
	branches 244:13	bunch 292:15, 303:19	
	branching 274:19		
	break 220:19, 274:20, 297:21		
	bridge 270:15, 270:18,		

bz0b1 241:18	233:16, 300:18, 312:21	305:19, 306:12	certificate 329:1, 329:9
C	called 180:14, 204:18, 225:6, 249:18, 262:7, 301:7, 309:9, 324:3	case 173:5, 174:5, 177:8, 178:18, 179:7, 180:9, 191:1, 194:20, 198:18, 204:14, 204:18, 206:7, 207:22, 208:2, 209:11, 219:14, 226:17, 226:21, 238:12, 244:7, 245:2, 252:19, 252:22, 254:12, 254:15, 265:3, 294:12, 303:6, 305:15, 312:19, 312:23, 314:4, 314:8, 315:1, 327:3, 327:6	certified 174:26, 178:2, 329:1, 329:7, 329:22, 330:10
c) 330:19	came 198:7, 198:17, 199:15, 200:20, 208:16, 251:12, 252:21, 268:14	cases 238:2	certify 328:6, 329:6
cache 223:12, 241:16, 242:20, 242:25, 243:5, 243:9, 243:15, 243:25, 244:19, 245:3, 250:23, 251:5, 251:15, 253:11, 253:20, 253:25, 254:3, 254:6, 254:8, 254:10, 254:16, 258:11, 259:13, 259:14, 260:13, 271:11, 271:15, 271:21, 272:8, 272:10, 273:20, 274:7, 275:19, 276:9, 282:25, 294:25, 295:1, 314:4, 315:10, 315:16, 315:23, 316:7, 316:20, 317:5, 317:20, 318:1, 318:13	camera 182:5, 320:13	cetera 233:9	chair 320:21, 323:13, 323:17, 323:19
cache" 315:20	cameron 176:23	categorize 305:12	challenging 283:5
caching 273:20	can't 201:22, 201:23, 202:1, 248:19	categorized 309:20	change 248:21, 254:9, 272:14, 275:8, 279:12
calculated 234:24	cannot 179:11, 192:19, 202:5, 219:11, 219:16, 219:17, 230:21, 239:20, 297:14, 301:1, 301:3	caught 227:17	changed 200:25, 251:17, 280:12
calculations 300:21	capability 211:25, 311:23, 312:3	causes 202:19	changes 247:7, 248:23, 251:8, 272:4, 281:19
california 173:22, 174:23, 175:19, 177:1, 320:18, 328:1, 328:8, 328:11, 329:2, 329:8, 329:10, 329:17	capable 210:6, 210:15, 210:23, 211:10, 211:15, 212:5, 212:25, 290:5	causing 203:4, 203:11	changing 268:23, 280:18, 281:11, 281:21
call 201:9, 219:25,	capacities 324:5	caveat 231:23	channel 202:25, 241:22, 246:24, 250:3, 250:13
	careful 295:2, 295:22	caveats 240:1	characteristics 234:10
	carefully 207:15	century 180:14	charles 320:21
	carmel 175:17	certain 209:7, 209:8, 279:11, 294:20, 294:21, 296:20	check 195:15, 255:21, 266:19, 279:24, 280:20, 281:2, 281:3, 319:1
	carries 280:11	certainly 184:16, 236:25, 237:21, 257:15, 278:22, 286:16, 293:13, 300:4, 313:6, 315:22	choice 196:9, 196:13, 196:18, 196:22, 205:19, 217:2, 217:3, 217:4
	carry 276:10, 280:18		choose 213:25, 215:8,
	carrying 280:23, 282:3,		

215:17, 218:2, 241:12, 305:22, 307:1, 309:22 chooses 197:1, 197:14, 197:17, 197:18, 215:15, 217:1, 241:15, 245:15, 305:5 choosing 197:6, 218:9 chosen 190:18, 218:10, 218:13, 264:19 circuit 321:8 circuits 323:24, 324:2, 324:4 citation 224:1, 224:6, 224:8, 225:12, 225:14 citations 223:14 cites 242:15 civ 329:15, 329:20, 329:25, 330:7, 330:19 civil 329:17 claim 188:6, 188:9, 189:3, 189:16, 189:25, 190:16, 190:23, 191:12, 191:14, 192:7, 192:9, 192:15, 192:22, 193:19, 194:13, 194:16, 196:5, 201:17, 205:6, 205:25, 206:9, 207:3, 209:1, 209:2, 209:11, 209:16, 209:17, 209:24,	210:21, 210:22, 211:2, 211:6, 211:14, 212:4, 212:15, 213:4, 213:5, 213:20, 214:2, 214:13, 215:3, 215:18, 216:19, 216:22, 217:7, 217:20, 218:11, 218:16, 219:23, 220:2, 220:4, 220:5, 220:6, 220:15, 256:10, 256:12, 256:14, 258:6, 261:8, 299:7, 299:11, 299:18, 299:19, 299:24, 311:22, 311:23, 323:7 claimed 205:23, 256:13 claims 176:16, 191:4, 195:3, 195:19, 208:12, 208:13, 208:24, 208:25, 209:4, 212:3, 215:23, 216:16, 249:5, 270:5, 298:8, 307:7, 319:23 clarification 181:10, 184:19, 324:1 clarify 181:14, 200:11 classes 320:23, 321:5, 321:20, 322:18, 322:21 classification 186:17 clean 187:21 clear 182:22, 211:24, 213:5, 228:13,	246:13, 255:6, 255:18, 267:21, 306:23, 311:16, 317:8, 317:25 clearly 245:18, 254:22, 256:4, 313:23, 320:3 clock 203:13, 203:14, 310:22, 310:23, 310:24 close 271:24 co-chair 323:23 code 283:25, 329:17 cohn 175:14 coin 295:17, 295:18 cold 179:5 collaboration 185:7 collect 204:16, 330:15 color 240:8 column 229:1 com 175:10, 175:11, 175:21 combination 182:2, 267:6, 270:5, 270:8, 271:8, 272:9, 275:25, 277:4, 278:19, 279:7, 282:1, 282:5, 282:10, 298:22, 301:13 combine 247:23, 268:2, 272:5, 276:8 combined 268:7, 268:11,	276:23, 292:6, 314:5 combines 266:18 come 185:17, 240:9, 259:1, 260:19, 278:19, 304:14, 305:25, 308:12, 308:13, 308:20, 310:5, 313:8, 315:2, 316:16 comes 197:23, 198:6, 198:13, 199:3, 199:14, 200:19, 202:8, 203:10, 203:15, 240:11, 247:16, 253:10, 290:16, 290:21, 302:21, 303:3, 303:5, 303:19, 304:5, 310:12, 312:5, 312:7 coming 204:11, 244:24, 286:6, 286:19 commands 186:4, 190:10, 205:21, 208:22, 214:22, 215:8, 215:21, 217:4, 218:22, 242:19, 243:3, 250:23, 250:24, 253:9, 262:16, 281:17, 283:17, 283:18, 283:20, 283:22, 284:1, 284:9, 284:24, 287:9, 287:23, 288:1, 288:3, 288:8, 288:21, 292:16, 294:22, 301:24, 302:10, 303:9, 303:19, 304:2, 304:4, 304:14, 309:20, 309:22
---	---	---	---

comment 206:11	comprising 185:9, 194:1	conjunction 242:11, 278:9	193:4, 193:20, 195:22
comments 277:24, 330:11	computation 185:12, 185:13	connect 243:19, 259:6, 291:18	constructions 192:9, 192:22, 205:3, 205:7
committee 323:12, 323:20, 324:5	computational 249:1, 279:18	connected 242:25, 270:14	constructions" 188:13
common 195:6, 249:2, 249:3, 294:11, 294:12, 309:17	computations 185:10, 233:7	connection 183:7, 243:9, 243:15, 243:23	construed 188:10, 188:21, 190:3
common-case 294:18	computer 233:19, 320:20, 320:22, 321:16, 322:18	connections 243:17	construing 176:15
commonly 261:24	concept 209:25	consecutive 289:8	consulting 262:4
communication 276:2, 309:2	concepts 185:15	consecutively 274:12, 274:23, 315:2	contain 270:18
compatibility 291:25	concerned 194:23	consider 192:4, 207:1, 208:10, 256:19, 263:9, 314:3	containing 223:12
compatible 277:1, 277:23, 277:25, 278:3	concerning 226:1	considerations 185:20, 189:20, 264:15, 319:10	contains 194:15, 221:3, 226:10
compensate 324:15	concluded 185:4, 326:13	considered 291:1	contend 302:14
complete 194:5, 194:12, 194:15, 195:18	condition 254:1, 294:20	consistency 291:25	contending 313:2
completely 200:3, 248:9, 278:3	conditions 182:4, 235:2, 250:7, 252:16, 253:3	consistent 191:24, 196:3, 197:4, 199:21, 216:19, 271:18, 290:21	content 221:5, 221:18
complicated 198:25, 260:11, 263:1	conference 323:24	consisting 330:10	contention 307:25, 308:2, 308:6, 308:19, 312:6, 312:10, 313:5, 317:24
complication 292:8	conferences 322:17, 322:19, 322:20, 322:21, 324:6	constraint 296:24	context 196:15, 197:3, 200:3, 201:16, 204:23, 205:1, 205:6, 250:18, 264:24, 265:7, 266:23, 275:22, 283:3, 296:16
component 181:23, 258:16, 258:25, 259:10, 260:1, 268:6, 271:9, 283:23, 290:15	confidential 178:14, 178:15, 178:20, 178:21, 178:22	constriction 188:6	continue 248:19
components 180:6, 183:1, 183:22, 271:11, 276:22, 296:3, 313:24	configuration 271:13	construction 189:4, 190:1, 190:2, 190:5, 190:6, 190:12, 190:21, 191:5, 192:7, 192:11, 192:13, 192:16, 192:24, 192:25, 193:2, 193:3,	continues 250:15, 253:1
comprise 213:5	configured 312:4, 313:6		continuous 284:15, 284:18
	confused 265:11		contract 179:21, 181:21,
	confusing 236:18		
	confusion 217:12, 246:19		

325:4 contracts 322:10, 325:2 contrast 180:22 control 242:6, 244:5, 245:20, 245:22, 245:23, 246:5, 246:9, 246:21, 282:22, 310:19, 320:4 control-flow 244:11 controlled 244:9, 244:10, 250:13 controller 241:21, 241:22, 246:24, 250:2, 250:3, 250:14, 260:2, 275:24 controllers 321:13 conventional 238:21, 275:12 convey 292:15 conveyed 251:9, 287:16 conveys 257:16 coordinate 233:5, 300:11, 300:22 coordinates 233:24, 300:8 copies 178:19, 187:22 core 250:1 corner 263:16, 263:19 corp 173:5, 174:5, 175:3, 327:3 corporation 177:7	correct 179:9, 187:19, 187:22, 188:6, 188:14, 190:23, 193:16, 193:20, 195:3, 195:22, 204:16, 205:15, 206:7, 213:14, 216:3, 218:13, 218:14, 218:15, 222:21, 223:1, 223:8, 223:20, 223:24, 224:21, 226:6, 226:25, 227:13, 227:24, 228:7, 228:12, 229:4, 229:9, 229:10, 229:12, 229:22, 229:23, 231:1, 231:3, 231:5, 231:14, 231:21, 232:14, 238:10, 238:12, 239:24, 258:8, 258:9, 261:11, 266:18, 266:22, 267:4, 267:5, 267:8, 267:9, 267:12, 268:1, 268:17, 268:22, 268:23, 270:1, 281:15, 285:2, 288:11, 288:12, 288:13, 297:17, 301:14, 301:19, 301:24, 302:3, 302:19, 302:24, 304:12, 305:7, 307:5, 313:15, 316:12, 317:6, 318:2, 325:22, 328:9 correction 243:1, 303:1 corrections 327:14 correspond 224:14, 224:16,	225:19, 243:25, 244:17 corresponding 225:8, 260:14, 288:23, 299:2 coughing 179:5 could 179:25, 184:14, 193:8, 198:20, 198:24, 199:6, 206:22, 212:15, 212:18, 213:19, 215:3, 215:22, 217:7, 226:23, 234:8, 235:24, 236:12, 244:5, 244:11, 248:14, 248:24, 253:24, 253:25, 254:12, 254:24, 254:25, 258:1, 258:11, 259:25, 260:2, 260:13, 260:17, 261:6, 266:6, 266:9, 266:10, 271:13, 271:14, 271:15, 271:23, 272:2, 272:15, 274:5, 274:7, 277:5, 279:5, 280:2, 280:11, 280:24, 285:13, 286:19, 292:5, 294:10, 294:22, 303:7, 305:18, 305:24, 306:21, 307:1, 309:12, 309:13, 310:10, 314:20, 315:7, 316:24, 325:10 couldn't 317:12 counsel 177:15, 221:2, 222:21, 325:14 county 328:3, 329:4	coupled 207:7, 209:9, 216:24 course 180:20, 183:20, 186:5, 186:22, 205:19, 208:6, 211:10, 219:1, 231:16, 243:19, 244:7, 263:14, 268:6, 301:15, 312:17, 315:2 courses 321:4, 321:5, 321:6, 321:10, 321:15 court 177:24, 329:9 covered 321:11 covers 220:5 cpu 273:7 criteria 198:3, 305:25 csr 173:30, 173:31, 329:9, 330:25 cumbersome 263:2 current 318:7 currently 184:20, 187:14, 320:17 curvature 263:21 cycles 203:14, 310:23, 310:24
			D darpa 184:21, 185:5 data 185:14, 197:5, 197:7, 198:6,

198:7, 198:9, 198:10, 199:14, 199:18, 200:1, 200:18, 200:19, 200:23, 202:8, 202:11, 202:21, 202:22, 202:25, 203:10, 203:13, 203:15, 203:19, 204:16, 205:15, 206:23, 223:13, 223:23, 224:4, 224:17, 224:20, 224:24, 224:25, 225:9, 225:21, 225:22, 226:1, 226:8, 226:10, 226:18, 226:22, 227:10, 227:23, 228:11, 229:12, 229:18, 229:21, 230:5, 230:10, 231:1, 231:4, 231:7, 232:23, 233:10, 235:10, 235:15, 248:16, 248:18, 256:13, 258:15, 259:23, 260:9, 264:19, 276:2, 277:11, 277:21, 280:11, 294:24, 306:6 date 177:9, 327:10, 327:30 dated 330:21 day 315:14, 328:10 daytime 182:4 ddr 321:13 debating 303:2 decades 262:25 decide 241:19, 244:12,	251:5, 263:19, 276:15, 290:19, 293:11, 303:10, 304:4, 306:21, 306:24, 308:4, 311:4, 312:22, 314:25 decided 279:2, 279:5 decides 215:16, 219:20, 245:11, 245:16, 246:4, 307:1, 319:11 deciding 184:6, 245:7, 263:12, 263:15, 299:2, 303:24 decision 187:15, 245:24, 246:6, 305:10, 309:7, 309:14, 310:10, 310:13, 311:1, 311:12, 315:3, 317:15, 324:12 decision-making 309:10 decisions 188:11, 254:15, 288:5, 309:16, 317:15 declaration 176:10, 187:15, 188:1, 188:12, 188:21, 189:25, 192:5, 207:20, 207:23, 208:4, 223:4, 223:5, 240:14, 242:5, 249:9, 276:21, 278:1, 325:12 declare 257:2 declared 178:6, 253:24 declares 256:24	decoupled 292:5 dedicated 277:10, 279:11, 280:3, 282:11, 288:22, 293:8, 294:21, 313:12 dedication 292:8 deemed 253:18 defense 184:20, 185:5 define 205:8, 236:20, 262:2 defined 257:15 defines 253:23 defining 195:18 definitely 186:7, 224:23, 263:24, 264:23, 325:23 definition 261:25, 265:4 degradation 180:21 delay 198:19, 203:11 delayed 203:3 delete 178:19 deliberate 309:5 demeanor 330:11 densely 296:3 dependant 258:6, 261:8, 299:18 dependencies 248:17 dependent 234:9	depending 184:15, 232:2, 253:25, 263:20, 264:19, 284:2, 304:18 depends 231:24, 236:23, 237:18, 262:3, 311:5 depicts 223:22 depo 327:10 deponent 327:12, 329:19 depos 177:13, 177:25 deposition 173:20, 174:20, 175:26, 177:5, 177:14, 178:11, 178:16, 187:5, 195:15, 196:2, 221:2, 221:7, 221:12, 221:22, 221:23, 221:25, 222:1, 222:4, 222:5, 222:9, 222:15, 222:20, 325:14, 326:10, 326:13, 327:1 depth 203:14 describe 225:14, 246:20, 247:22, 254:19, 283:6, 289:17, 303:13, 303:16, 307:12, 311:8 described 181:16, 206:15, 228:10, 233:5, 234:14, 245:18, 246:11, 255:22, 279:16, 283:13, 303:8, 303:15, 307:19, 309:20, 314:6, 314:24
--	---	--	---

describes 210:13, 231:2, 248:4, 251:23, 260:13, 280:16, 319:15, 320:2, 321:6 describing 224:6, 229:2, 233:2, 233:17, 247:25, 252:5 description 176:9, 223:19, 233:20, 234:4, 241:25, 243:11, 264:18, 290:18, 325:21, 326:1 descriptions 240:1 descriptor 288:1, 288:4, 288:5, 288:10, 290:7 design 180:5, 181:7, 181:9, 183:25, 185:22, 186:3, 238:21, 296:2, 321:6, 321:16, 321:17, 321:19, 321:22, 322:19, 323:18 designation 251:19, 256:20, 273:10 designed 179:22, 183:21, 184:10, 184:11, 184:13, 184:16, 185:23, 185:25, 238:6, 260:6 designing 179:16, 179:23, 183:18 designs 180:16 desktop 182:16, 187:7 destination 271:1	detail 196:15, 199:10, 260:21 detailed 241:25, 243:14, 250:9 details 198:9, 198:23, 199:12, 229:25, 230:2, 243:16, 310:2 detect 286:22 determination 176:18, 176:21, 178:12, 180:1, 193:15, 209:23, 217:25, 254:13, 303:17, 303:20, 304:9 determine 208:21, 234:16, 239:22, 249:25, 287:5, 288:10, 305:11 determined 219:5, 234:20, 238:9, 250:7, 284:5 determines 302:21 determining 191:12, 245:13, 252:4, 253:18, 288:15, 290:5, 319:15 develop 324:19 developed 183:16 developing 185:2 device 194:1, 206:15, 207:4, 207:13, 209:6, 209:10, 209:14, 209:15, 212:17, 212:18, 212:19, 212:20, 212:24, 214:3, 214:6, 216:24, 223:6, 299:1 devices 234:1, 258:7 devoted 228:7, 228:17 dictated 219:5 diego 175:19 difference 239:11 different 179:18, 182:3, 182:23, 183:22, 185:9, 185:16, 185:23, 192:6, 194:15, 197:21, 198:3, 201:8, 207:5, 207:6, 213:19, 215:11, 221:9, 221:20, 222:17, 225:8, 225:9, 232:18, 234:6, 234:7, 237:14, 237:22, 237:24, 238:1, 238:3, 239:15, 241:18, 243:17, 244:6, 245:7, 247:9, 259:23, 262:1, 262:2, 263:7, 265:16, 271:2, 273:8, 273:11, 273:13, 273:14, 274:6, 275:19, 276:9, 276:13, 276:22, 277:21, 278:18, 280:3, 281:1, 281:4, 281:8, 282:3, 282:11, 283:8, 283:9, 284:18, 285:15, 285:19, 285:25, 287:8, 287:21, 287:25, 289:12, 292:9, 293:25, 294:3, 295:18, 296:12, 296:13, 296:14, 297:6, 299:3, 301:10, 303:22, 304:8, 304:19, 305:16, 305:20, 305:24, 311:16, 315:10, 321:4, 321:8, 321:10, 321:11, 322:2, 322:17, 323:3, 324:6, 324:8 differently 214:15 difficult 278:14 direct 316:6 directional 315:25 directly 185:14, 324:16 disagree 190:15, 190:19, 191:1, 194:19, 195:23, 236:2, 262:22, 284:13, 285:4 disagreed 195:25, 196:1, 196:2, 196:3 disagreeing 269:12 disagreement 231:8, 235:22 disclosed 223:7, 259:15, 268:16 discloses 239:11, 260:16, 260:23, 267:15, 270:16, 288:6, 299:19, 300:1, 302:7 disclosure 207:10, 255:5,
--	---

258:11, 261:9, 264:24, 265:10, 280:15, 281:22, 284:23, 284:24, 285:7, 285:10, 286:4, 288:14, 308:16, 316:4 disclosures 208:23, 261:7 discouraged 308:24 discrepancy 220:16 discriminative 229:21 discuss 178:20, 190:25, 193:7, 250:18, 287:15 discussed 191:5, 193:6, 220:6, 226:13, 230:4, 237:21, 254:1, 261:2, 267:17 discusses 220:6, 224:10, 249:8, 300:1 discussing 233:16, 238:15, 240:13 discussion 219:24, 249:10, 251:25, 254:21, 256:11, 262:11, 299:4, 302:17, 317:16 discussions 250:9, 253:2, 318:6 dispatch 305:23, 311:13, 314:25 dispatched 305:21 dispatches 270:22 dispatching 273:21	display 179:24, 180:7, 180:19, 181:23, 181:24, 182:2, 182:13, 182:15, 184:1, 184:2, 227:13, 231:21, 233:21, 234:1, 234:8, 235:6, 240:6, 300:13 displayed 226:3 displaying 240:4 distinction 208:11 distinguish 287:21, 287:25, 288:2, 288:7, 288:8 distinguishes 287:8 division 227:3 doctor 240:18 document 187:2, 187:10, 188:17, 193:12, 193:14, 206:3, 206:6, 221:1, 221:3, 221:7, 221:8, 222:6, 269:21 documented 182:8 documents 187:7, 187:12 doing 180:17, 184:20, 185:10, 186:17, 197:20, 219:14, 219:15, 230:20, 233:4, 236:13, 236:14, 236:19, 236:21, 236:24, 237:1, 237:2, 237:9, 237:10,	237:12, 260:3, 260:21, 262:23, 263:2, 265:8, 281:14, 290:18, 306:15, 311:7, 314:16 dokhanchy 175:16, 176:5, 177:18, 178:9, 179:1, 181:13, 187:1, 187:6, 187:11, 188:19, 193:11, 193:13, 206:2, 206:5, 220:18, 220:25, 221:6, 221:8, 221:20, 221:24, 222:3, 222:8, 222:12, 222:18, 222:22, 222:23, 244:2, 257:19, 258:1, 258:3, 269:23, 298:2, 298:4, 317:8, 320:8, 326:5 dollars 324:25 done 179:18, 180:11, 183:16, 186:7, 198:2, 203:22, 205:3, 209:23, 226:18, 236:5, 238:19, 239:21, 248:10, 262:20, 282:13, 300:7, 300:15, 302:9, 303:18, 310:10, 310:16 doubt 312:3 down 188:4, 189:6, 190:7, 193:24, 206:9, 209:16, 225:17, 242:5, 266:15, 321:18, 323:5	downloaded 221:15 downstream 203:7, 206:23, 292:3 dr 176:10, 177:5, 178:17, 179:2, 188:1, 193:16, 201:19, 204:19, 206:7, 219:12, 222:24, 258:4, 262:19, 269:24, 284:22, 298:7, 320:17, 326:6, 326:10 dropbox 221:19 due 186:13, 192:18 duly 329:7 duplicative 291:19 during 178:11, 178:21, 182:7, 232:8, 240:9, 280:8 dynamic 186:10, 250:7, 279:3 dynamically 279:2, 279:6, 282:14
E			
each 190:19, 191:7, 192:1, 218:11, 220:7, 226:10, 242:15, 245:7, 249:21, 256:14, 256:15, 260:14, 264:9, 283:21, 288:2, 288:10, 306:4, 313:13, 316:19, 322:13 earlier 217:3, 228:4,			

241:9, 313:9, 315:14, 317:17, 319:14, 324:17 early 179:20 easier 275:6 easily 308:8 easy 271:25, 272:7, 292:25 effect 241:11, 307:22, 329:11 efficiencies 292:2, 293:12 efficiency 228:19, 289:2, 289:4, 290:24, 291:9, 292:6, 292:11, 293:13, 293:14, 295:17, 295:23, 296:7, 297:1, 297:5, 309:2, 313:15, 314:1, 314:14, 314:18 efficient 274:14, 276:2, 289:10, 293:17, 295:20, 296:1, 296:22, 297:8, 297:16, 297:19, 313:21 eight 230:16, 237:22, 237:23, 241:17, 243:24, 244:1, 244:4, 244:6, 244:8, 244:16, 244:17, 245:2, 245:6, 247:14, 260:14, 276:12, 276:13, 316:19, 316:20, 316:22, 316:23, 316:25, 317:5, 317:19	either 189:9, 191:7, 206:12, 211:4, 211:6, 213:2, 214:21, 216:7, 216:11, 216:13, 216:25, 217:21, 218:4, 226:17, 226:20, 248:11, 254:12, 271:13, 273:1, 285:13 elaborate 241:8, 274:16 electrical 320:20, 320:22 electronics 323:18, 323:21 element 195:19, 201:12, 201:13, 206:10, 206:16, 206:24, 213:5, 214:3, 218:15, 223:6, 240:16, 267:15, 269:14, 272:11, 277:20, 278:5, 282:6, 301:25, 302:8 elements 201:11, 201:15, 225:18, 226:9, 226:14, 226:16, 226:22, 226:24, 227:4, 229:3, 232:3, 232:21, 239:2, 243:17, 268:2, 278:17, 323:6 elliot 176:23, 189:1, 190:3, 194:20 else 182:14, 200:13, 201:14, 209:16, 214:24, 235:3, 248:14, 249:6, 265:6, 292:4, 300:19, 302:13,	306:6, 312:12, 313:2, 316:21 embedded 301:6 embodiment 279:15, 279:19, 280:22, 280:24, 280:25, 281:5, 281:10, 281:15, 281:18, 283:10, 283:12 embodiments 281:4, 281:8, 283:8 employee 329:13 employing 308:25 enable 185:11 enables 239:23, 247:5 encoded 287:13 encrypted 185:14 end 189:17, 203:10, 203:15, 258:20, 258:21, 258:23, 258:24, 326:9 ending 174:24, 226:5 engine 185:9, 210:2, 210:3, 210:5, 210:15, 210:23, 211:10, 211:15, 212:5, 217:6, 218:21, 218:23, 218:25, 219:5, 219:7, 219:10, 219:11, 219:14, 219:15, 219:16, 219:17, 219:19, 219:21, 220:12, 234:3, 261:11, 261:16, 261:17,	261:22, 262:8, 262:9, 265:2, 299:8 engineering 320:21, 320:22 engines 209:25, 210:4, 218:11, 218:17, 218:18, 219:1, 219:18, 220:1, 220:7, 220:10 enhancement 180:22, 324:14 enough 198:12, 228:22, 236:12, 243:16, 265:24 enter 258:19 entered 203:14 enters 258:22 entirely 181:7 entities 177:22, 201:8, 313:24 entitled 265:16 entity 204:18, 276:5, 292:11 entries 198:20, 276:8 errata 327:1 escaping 186:13 especially 195:10, 243:18, 247:22 esq 175:5, 175:16 essentially 204:2, 233:3, 283:9 et 233:8
--	--	---	--

even 190:9, 195:25, 251:15, 257:9, 270:19, 310:4, 311:4, 311:7, 317:25, 323:15 events 295:4 eventually 291:4 ever 184:10, 212:12, 212:19, 214:25, 319:18, 323:12 every 191:21, 227:17, 265:18, 310:21, 310:22, 310:23 everybody 275:10 everything 191:20, 195:16, 225:20, 239:25, 247:15, 279:2, 322:11, 323:10 evict 295:1 evidence 193:25, 224:3, 261:10 exact 287:15, 325:1 exactly 214:12, 228:13, 229:15, 235:13, 255:19, 276:22, 278:19, 279:23, 280:5, 287:12, 304:11, 317:12 examination 176:5, 176:6, 178:25, 320:15 examined 178:7, 329:19 example 184:5, 197:12, 198:14, 204:25, 207:12, 211:8, 	230:6, 230:7, 230:17, 232:6, 233:18, 241:16, 243:22, 244:11, 245:24, 246:24, 249:14, 254:2, 257:8, 263:11, 263:23, 268:4, 270:25, 271:3, 271:14, 273:22, 274:23, 283:16, 300:6, 300:9, 305:18, 305:25, 315:1, 315:24, 316:19, 317:5 examples 265:1, 266:10, 315:7 exceptions 294:9, 294:10 excerpt 224:8, 249:18 exclude 278:21, 279:9 excuse 227:22, 243:25, 255:1 execute 239:4, 244:22, 246:1, 246:2, 246:7, 294:8 executed 244:14, 244:25, 245:4, 245:17, 247:4, 254:7, 279:17, 294:5, 294:17, 304:21, 328:10 executes 211:11 executing 247:19, 253:25, 254:6, 306:3 execution 239:6, 239:8, 244:23, 247:8, 250:8, 253:21, 254:3, 254:4, 	273:12, 273:14, 274:10, 276:3, 280:9, 286:15, 291:5, 309:3, 319:8 executive 323:20 exhaust 230:13 exhibit 176:9, 176:10, 176:15, 176:18, 176:25, 176:28, 176:30, 178:12, 178:16, 178:21, 178:23, 187:2, 187:4, 187:25, 188:16, 188:18, 189:2, 189:3, 193:10, 193:12, 206:3, 206:4, 221:1, 221:6, 221:11, 221:21, 221:25, 222:4, 222:7, 222:11, 222:13, 222:19, 222:24, 258:2, 258:4, 258:5, 269:20, 269:22, 269:24, 298:3, 325:10, 325:12 exhibits 258:2 exist 243:5, 243:24, 271:18, 272:19, 279:8, 285:23, 287:3, 306:20 existing 245:3 exists 228:21, 233:12, 235:4, 236:25, 242:20, 308:23, 311:24 exit 258:20 expense 279:9 	expensive 274:20 experience 179:16, 195:8 experimentally 182:24 expert 262:19 experts 207:22, 208:2, 239:13 explain 259:11, 260:21, 269:18, 269:19, 295:9 explaining 223:6, 255:2 explains 277:25 explicit 207:10, 208:12, 243:11, 246:13, 270:15, 291:23 explicitly 255:22, 260:23 express 208:13, 316:4 expressed 255:5 expressly 261:2, 319:1 extensively 181:8 extent 257:17, 270:12 extrapolate 263:18 F fact 180:8, 203:1, 209:12, 215:8, 215:13, 234:25, 238:24, 248:11, 280:7, 281:14, 286:23, 291:7, 296:13, 300:19, 301:23, 303:22,
---	---	---	---

311:10, 314:8, 323:6 factor 289:4 faculty 180:4, 183:12, 185:7 fair 195:24, 213:12, 284:6, 288:17 familiar 184:12 far 194:22, 209:22, 257:1, 275:21, 286:22, 318:17, 325:4, 325:8 fast 296:3 faster 296:21, 297:17, 315:3, 315:4 feel 179:5, 276:21, 301:16, 304:12 feeling 179:2 ferris 175:14 few 320:14 field 182:1, 182:11, 182:25, 183:5, 261:23, 286:25, 287:1, 317:13 field-programmab- le 295:24 field-tested 180:10, 182:9 fifo 198:6, 198:7, 198:14, 198:20, 199:4, 199:5, 199:14, 199:15, 200:19, 201:14, 202:9, 202:24,	202:25, 203:2, 203:5, 203:11, 203:12, 203:14, 203:16, 203:18, 203:20, 258:13, 258:15, 258:20, 259:2, 259:3, 259:20, 259:21, 259:24, 260:6, 260:8, 260:10, 261:5, 289:19, 290:2 fifos 199:7, 258:12, 259:7, 259:15, 259:17, 260:1, 260:14 figure 223:19, 223:22, 224:11, 224:13, 224:19, 225:17, 225:18, 225:24, 238:5, 239:18, 239:20, 241:2, 241:3, 241:15, 242:18, 242:25, 243:8, 250:3, 250:18, 270:3, 279:13, 279:15, 279:18, 279:21, 280:23, 280:24, 281:15, 283:10, 283:12, 298:5, 315:13 file 221:19 fill 277:19 filled 214:12 final 240:3 finally 225:22 financially 329:12 financing 330:3, 330:14,	330:18 find 208:17, 209:22, 246:15, 282:10, 294:1, 295:22, 317:8 finding 193:25, 194:19 finds 264:18 fine 187:24, 200:7, 204:12, 204:13, 206:2, 206:18, 242:3, 250:19, 312:8, 312:12, 316:7 fine-level 246:8 finish 236:7, 272:3 finished 219:12, 248:7, 248:8 finishing 244:23, 281:14 firing 184:7 first 194:1, 206:21, 209:21, 214:8, 223:17, 224:1, 224:5, 234:15, 234:19, 234:23, 251:15, 251:19, 251:22, 252:19, 258:23, 258:24, 261:15, 263:15, 275:3, 275:6, 283:17, 284:5, 305:25, 308:20, 308:21, 309:24, 315:22, 329:19, 330:4 first-in 258:15, 258:17, 258:21, 258:22 first-out 258:15, 258:18,	258:21, 258:22 five 309:13, 310:23, 311:1, 311:2 fixed 199:1, 279:25, 280:1, 282:24, 284:15 flexibility 295:6, 295:16, 295:23, 295:25, 296:6, 297:7 flexible 295:20 flight 206:24, 209:15 floor 175:7, 250:13 flow 203:18, 225:24, 238:5, 239:17, 245:22, 245:23, 246:9, 246:21, 274:21 flow-up 246:5 flowchart 223:23 flowing 199:7 flows 279:5 focus 181:7, 206:1, 299:17, 313:18, 321:21 focuses 323:20 follow 238:16 follow-up 320:9 followed 213:18 following 249:19, 252:10 follows 178:7
--	--	---	--

force 329:11 forefront 261:5 foregoing 328:8, 329:24 form 190:24, 191:18, 192:17, 195:4, 196:11, 196:21, 197:9, 198:11, 199:19, 200:24, 202:12, 204:20, 205:16, 205:24, 210:8, 211:1, 212:21, 212:22, 213:13, 214:1, 214:19, 215:5, 215:25, 216:4, 216:17, 217:16, 224:15, 224:22, 227:14, 231:13, 234:21, 235:12, 238:11, 244:2, 251:21, 256:8, 259:22, 268:24, 269:3, 270:6, 271:12, 272:25, 276:20, 278:13, 279:14, 281:25, 285:3, 288:25, 293:22, 305:14, 314:2, 318:14, 319:21 formula 303:2 forth 180:25, 192:11, 228:23, 244:13, 275:5, 279:25 fortunately 325:7 forward 198:1 found 195:22, 200:2, 208:10, 256:10, 270:10, 275:4	founding 323:16 four 183:15, 189:18, 190:19, 191:6, 191:15, 191:25, 203:14, 230:15, 275:2, 292:19, 293:1, 304:19, 309:13, 314:23, 314:24, 315:1, 325:6 fpga 295:24 frame 181:1, 182:11, 186:16, 237:6 free 301:16, 304:12 frequently 183:14 friday's 178:11, 178:16, 221:21, 222:4 front 202:24, 258:20, 258:23, 260:18, 283:5 full 230:7, 249:10, 329:10 full-blown 184:17 fully 185:11 function 197:3, 197:20, 199:22, 204:7, 205:11, 231:2, 261:18, 283:13, 290:17, 291:17, 291:19, 298:20, 301:2 functional 283:23 functionality 180:13, 252:16, 268:9, 298:24, 314:6	functions 198:5, 290:22 fundamental 239:1, 239:10, 296:5, 296:8 funded 325:7 funding 184:16, 322:9 further 190:4, 193:5, 193:19, 199:12, 199:16, 200:21, 247:2, 292:6, 326:5 furthermore 186:9, 186:15 G gate 295:24 gave 266:9 gbs 268:4, 268:6, 270:21, 275:24, 276:18, 278:8, 283:6, 283:14, 283:20, 284:4, 286:7, 286:12, 287:8, 287:11, 287:17, 287:20, 287:24, 288:2, 288:4, 288:7, 288:15, 289:4, 289:5, 289:11, 289:15, 289:19, 290:2, 290:5, 290:16, 290:17, 290:21, 290:22, 291:6, 291:17, 291:22, 293:3, 293:15, 298:13, 298:16, 298:19, 301:23, 302:9, 302:21, 302:22, 303:12, 303:15, 304:7, 304:10, 304:13, 304:15, 304:20, 305:5, 305:11, 306:24, 308:12, 308:13, 309:21, 309:22, 310:4, 311:12, 311:19, 311:21 general 186:24, 192:2, 245:2, 245:20, 282:21, 282:22, 320:4, 323:17, 323:23 generally 202:15, 224:16, 227:23, 228:11, 269:9, 274:16, 295:15, 306:19 generals 180:9 generic 200:12 generically 234:1 geometric 300:21 geometry 224:17, 224:20, 233:4, 287:23, 300:5, 301:4 geometry-type 283:16 getting 236:22, 269:15 give 179:11, 188:5, 197:12, 221:14, 266:7, 309:7, 312:13, 312:18, 313:3, 315:6, 323:7, 325:9 given 194:5, 194:12, 198:10, 232:3, 232:17, 238:8, 260:10, 265:24, 307:2, 312:2,
--	---	--

329:25 gives 193:19, 295:25, 297:6 giving 221:16, 312:16 gl 284:1 glad 193:6 global 270:20, 303:23 glovsky 175:14 go 187:1, 189:6, 190:2, 192:22, 193:18, 197:18, 197:19, 202:4, 210:11, 217:25, 223:16, 224:11, 225:12, 225:17, 228:25, 240:14, 241:2, 245:25, 246:2, 250:5, 251:10, 251:25, 252:19, 257:19, 259:14, 261:3, 266:15, 273:11, 273:13, 274:7, 279:21, 279:25, 282:15, 284:19, 286:15, 287:18, 291:16, 293:4, 293:9, 295:23, 298:5, 300:12, 301:9, 304:3, 312:9, 315:6, 315:12, 317:18, 317:23, 319:12, 320:10, 325:10, 325:17 goal 309:1 goes 198:16, 199:4, 202:25, 203:12, 203:16, 242:10,	252:23, 254:19, 271:2, 271:3, 276:15, 276:16, 283:24, 285:21, 285:22, 303:10 going 180:20, 187:9, 203:9, 217:17, 220:20, 222:16, 225:23, 226:18, 230:10, 236:5, 237:9, 241:20, 245:2, 245:17, 246:13, 247:8, 247:15, 253:3, 253:23, 257:21, 273:10, 273:11, 273:20, 274:9, 276:13, 279:17, 288:21, 291:9, 292:17, 296:3, 297:22, 300:11, 306:18, 308:19, 309:7, 312:18, 316:6, 323:19, 326:10 gone 248:10 good 186:22, 283:2, 283:4, 315:7 government 325:5 gpgpus 186:23 gpu 181:7, 181:8, 181:9, 181:17, 184:10, 184:11, 184:13, 301:3 gpus 184:12, 186:17, 186:22 graduate 322:12 graduates 321:9 granting 187:15	graphic 186:23, 234:4, 237:4, 283:20, 287:9 graphical 226:2, 283:18 graphics 179:17, 179:25, 180:12, 180:16, 181:11, 181:25, 184:17, 184:24, 185:14, 185:16, 185:20, 186:4, 186:6, 186:12, 209:5, 213:15, 216:21, 224:20, 226:3, 226:5, 226:9, 232:16, 233:18, 233:20, 234:3, 234:12, 237:8, 238:13, 238:15, 238:16, 238:20, 240:12, 257:8, 261:23, 262:2, 266:8, 275:23, 276:3, 276:5, 276:6, 276:14, 277:13, 277:16, 278:17, 278:24, 281:17, 282:22, 283:15, 283:16, 283:21, 283:22, 283:24, 286:16, 287:13, 287:22, 287:23, 288:1, 288:2, 288:3, 288:8, 288:11, 288:15, 298:23, 301:23, 302:10, 306:18, 321:12, 323:2, 323:22, 324:19 ground 185:3, 261:8, 266:17, 266:21, 266:22, 266:24, 266:25, 267:2, 267:6, 267:8,	267:12, 267:17, 267:23, 267:24, 268:1, 268:15, 268:17, 268:21, 269:11, 269:12, 270:5, 271:8, 275:25, 276:18, 277:17, 277:19, 278:5, 299:11, 299:13, 299:15, 299:16, 299:17, 299:18, 299:23, 301:13, 301:16, 301:18, 301:21, 302:14 grounds 267:1 group 180:2, 181:6, 304:15, 321:24, 322:7, 322:8, 322:14, 323:20, 325:7 guess 180:15, 190:4, 224:9, 261:15, 270:7 guessing 246:12 H half 181:6 halfway 245:5 halt 319:9 hand 178:3, 239:13, 254:5, 274:22, 293:3, 295:23, 295:24, 296:6, 296:7, 308:18 handed 190:6, 203:7 handle 230:15 happen 206:22, 235:24,
--	--	---	---

236:24, 248:2, 248:14, 251:20 happened 195:11, 200:21, 231:15 happening 203:4, 312:25 happens 231:10, 232:12, 232:16, 235:21, 251:22, 292:3, 292:25, 303:5 happy 180:11, 247:15 hardware 257:4, 257:5 hart 262:19 haul 253:21 head 182:17, 307:20 head-mounted 179:24, 180:6, 182:2, 182:15, 182:17, 184:2 heading 252:4 hear 269:1 heard 182:10 heavily 186:16 held 189:15, 198:14 help 269:16 helpful 250:19, 270:13 hence 252:12, 319:17 herby 329:6 here 177:4, 179:6, 184:19, 189:15, 190:15, 191:5,	191:13, 192:4, 192:21, 193:5, 195:23, 201:8, 212:2, 214:7, 216:9, 217:12, 221:13, 224:6, 226:13, 230:5, 230:18, 230:24, 233:16, 236:18, 241:18, 244:17, 253:6, 254:1, 255:21, 267:13, 268:8, 269:18, 269:19, 275:12, 278:2, 287:14, 288:18, 299:1, 310:12, 312:12, 312:13 hereby 328:6 herrington 175:4, 177:21 hidden 235:1 high 185:18, 247:17, 256:5, 257:3, 257:17, 260:24, 311:10 high-level 246:21 high-priority 247:13, 247:16, 252:21, 255:12, 256:11, 257:1, 257:7, 260:20, 261:4, 282:25, 317:9 high-throughput 308:25 higher 197:24, 252:11, 254:22, 255:6, 260:16, 297:5, 318:6 highest 249:16, 256:1 highlighted 187:8	highly 185:8 hint 190:15, 192:4 history 190:17 holder 329:8 homomorphic 185:11 honest 265:11 hope 179:2 host 226:1, 227:24, 228:5, 228:12, 229:2, 229:20, 231:5, 232:23, 233:24, 240:4, 282:14, 292:15, 293:14, 293:16, 293:18, 293:19, 293:24, 293:25, 294:22, 295:7, 296:17, 296:19, 297:13 host-system 233:6, 233:12, 233:15, 233:19, 234:5, 235:5 houston 175:8 huge 289:25 hundred 292:20, 292:23, 310:22 hybrid 279:7 hypothetical 204:8, 265:3, 317:18 I ic 296:2 id 193:20	idea 252:12, 283:2, 283:4, 308:20, 315:9 identified 239:19, 268:8, 298:22 identifies 269:11 identify 177:15, 240:20, 268:15, 274:11, 325:20 identifying 184:6, 208:23, 330:16 idle 230:22 ii 173:21, 174:20 illustrating 223:23 image 180:21, 182:1, 186:14, 186:17, 225:7, 228:22, 230:7, 236:14, 236:19, 237:2, 237:3, 240:4, 263:14, 324:14 images 186:18, 226:2 imagine 233:25, 237:4, 276:12, 292:14 immediately 304:4, 310:14 implemented 303:8 implementer 303:21 implements 314:6 implicate 262:17 implicit 286:19, 291:21 implicitly 284:14
---	---	--	--

implied 210:3 implies 256:25 implying 291:18, 304:7, 317:2 important 280:1, 295:4, 311:23 importantly 266:5 imports 315:8 improper 265:15 improved 228:19 improves 314:17, 315:4 include 191:13, 192:14, 192:25, 275:25, 277:16, 278:8, 278:11, 282:21, 300:25 included 179:24, 181:23, 186:21, 214:14, 224:24 includes 270:11, 282:22, 321:17 including 183:22, 188:23, 198:3, 225:21, 238:15, 270:4, 276:18, 321:11, 321:12, 322:18, 322:19, 323:22 incoming 306:2 incompatible 204:24 inconsistent 204:24 incorporate 276:1, 278:15,	278:16 incorporated 269:15, 276:23 incorporation 277:13 incorrect 205:1, 205:6, 285:2 increase 248:25 increasing 297:1 independent 239:3, 239:4, 245:3, 245:7, 291:5 indication 256:6 individual 240:7 information 178:14, 178:16, 184:3, 184:4, 198:12, 203:2, 226:4, 228:5, 228:15, 229:8, 229:16, 229:21, 232:22, 234:20, 234:24, 235:8, 235:14, 235:17, 235:18, 235:25, 236:4, 236:5, 236:16, 242:14, 243:21, 249:21, 249:22, 249:24, 251:9, 255:2, 255:7, 255:11, 255:13, 255:14, 255:19, 255:23, 256:21, 257:16, 259:13, 265:24, 273:15, 276:10, 286:6, 286:8, 286:12, 286:18, 287:12, 287:16, 289:3, 290:9, 290:19, 293:4, 298:25, 315:16,	315:23, 316:1, 316:3, 316:5, 316:10, 317:10, 317:13, 317:14, 317:21, 317:22, 318:7, 318:10, 330:16 informs 317:3 infrared 182:5 infringe 212:15, 212:18 initial 176:18, 178:12, 180:15, 193:14, 199:5 initially 231:5 injected 260:7 input 205:18, 237:18, 237:19, 247:3, 261:20, 270:14, 270:16, 270:20, 270:22, 271:7, 271:9, 271:14, 271:16, 271:20, 271:23, 272:6, 272:10, 272:14, 272:24, 273:7, 273:9, 273:19, 274:3, 274:5, 274:13, 275:16, 275:20, 276:8, 278:7, 278:9, 284:5, 284:8, 284:10, 284:25, 285:7, 285:12, 285:16, 285:19, 285:21, 286:1, 286:7, 286:20, 286:23, 288:16, 288:22, 289:2, 289:8, 289:17, 290:1, 290:16, 290:22, 290:25,	291:21, 292:1, 292:7, 292:16, 292:24, 298:17, 298:20, 298:25, 305:12, 313:11, 313:21, 314:4 inputs 204:3, 204:4, 204:11, 205:9, 205:14, 205:18 inserted 199:4 inside 233:19, 284:16, 301:6, 312:11 insist 204:25 instance 214:12, 280:12, 280:13, 304:8, 311:20 instead 247:14 institution 187:15 instruction 245:8, 245:11, 245:16, 246:1, 246:7, 246:8, 253:19, 253:22, 254:6, 254:8, 254:10, 256:14, 284:4, 319:9 instructions 223:11, 242:7, 244:5, 244:14, 246:3, 246:23, 250:1, 250:12, 259:1, 259:4 integrate 268:3, 273:1 integration 321:7 intensity 180:23, 261:19 intention 239:9 intentional 198:19
---	---	---	---

inter 176:12	186:16, 251:14	279:3, 279:6, 284:16, 287:24, 289:2, 290:6	314:2, 318:14, 319:21, 320:12, 320:16, 324:9, 325:16, 326:7
interactions 195:7, 199:6	180:15, 203:22	J	journals 322:17
interested 329:12	324:8	january 173:23, 174:25, 177:2, 177:9, 327:10	judge 176:23, 188:25, 190:3, 194:20
interface 292:15, 293:25	involves 233:7, 234:13, 263:15, 263:24, 264:21, 274:19	jeff 320:10	judge's 190:15
international 324:3	involving 180:4	jeffrey 175:5, 177:20, 269:1	july 176:24
interpretation 195:2	ipr 173:7, 174:7, 177:9, 179:7, 327:6	jerry 175:25, 177:12	jump 223:3, 274:18, 293:2
interpreted 208:25	iraq 182:11	jersey 183:14	jumping 274:19
interrupt 310:11, 310:18	issuance 242:6	job 173:32, 233:20, 234:3, 298:23	justification 190:4, 193:1
interrupts 310:12	issue 228:18, 244:20, 245:8, 245:11	johnson 175:5, 176:6, 177:20, 190:24, 191:18, 192:17, 195:4, 196:11, 196:21, 197:9, 198:11, 199:19, 200:24, 202:12, 204:20, 205:16, 205:24, 210:8, 211:1, 212:21, 213:13, 214:1, 214:19, 215:5, 215:25, 216:4, 216:17, 217:16, 224:15, 224:22, 227:14, 231:13, 234:21, 235:12, 238:11, 251:21, 256:8, 256:22, 259:22, 268:24, 269:3, 270:6, 271:12, 272:25, 276:20, 278:13, 279:14, 281:25, 285:3, 288:25, 293:22, 305:14,	K
intervals 310:9, 310:21	issued 206:7, 246:23, 250:1, 329:9		karen 175:26
intrinsic 193:25	issues 220:17, 273:23		keep 271:21, 323:15, 324:6
introduce 188:16, 206:3, 221:1, 269:20	itc 178:12, 188:10, 188:22, 189:1, 189:4, 192:11, 193:15, 195:1, 207:22, 208:2		keeps 253:2
introduced 187:2	item 248:16, 248:18		kept 203:6
invalidate 191:3, 191:8, 208:24	items 225:9, 260:7		kicks 252:16
invalidity 195:11, 208:8	itself 178:23, 186:14, 189:16, 191:14, 193:2, 200:1, 200:9, 202:21, 203:18, 204:5, 206:24, 207:3, 207:6, 209:2, 211:14, 234:8, 256:24, 257:2, 263:22, 268:6,		kind 180:13, 183:17, 228:19, 245:19, 291:2, 291:19, 294:7, 300:13, 300:17, 310:16, 323:4
invention 210:13, 238:23, 239:1, 239:10, 273:17, 306:9, 307:25, 326:2			kinds 180:18, 181:2, 186:17, 198:2, 234:1, 249:2, 266:1, 266:2, 296:8, 321:13
inventor 210:12, 324:10			king 312:15
investigation 188:22, 193:15			
investigator 180:3, 183:9, 183:10, 183:21			
involve 265:19, 266:4, 266:11, 291:2			
involved 180:5, 183:11,			

know 178:17, 194:20, 195:1, 198:23, 207:14, 207:17, 227:19, 235:19, 243:20, 247:17, 250:17, 251:22, 255:9, 255:17, 257:2, 257:10, 260:22, 266:6, 272:21, 272:23, 273:9, 275:2, 282:19, 283:2, 286:7, 286:17, 290:2, 290:9, 291:4, 291:15, 291:24, 292:17, 293:6, 293:8, 300:23, 302:16, 308:18, 309:5, 310:5, 310:21, 313:24, 315:15, 315:20, 317:13, 318:11, 318:19, 318:20, 319:22, 319:24, 319:25, 320:2, 320:10, 322:11 knowing 199:16 knowledge 203:22, 275:12, 291:21, 291:23, 306:17, 307:23 known 233:3, 238:22, 250:22, 262:5, 273:16, 275:10, 289:3, 290:23, 291:13, 293:2, 306:8, 308:7, 310:3 knows 255:18, 257:3, 260:5, 286:12, 286:17, 287:11, 289:5, 304:20, 318:1, 320:5,	320:6 L lab 182:24, 182:25, 183:1, 183:4 labs 183:12 land 179:22, 179:23, 186:6 language 190:16, 194:13, 199:21, 201:17, 205:25, 207:3, 208:13, 209:2, 211:2, 211:14, 213:4, 213:21, 216:19, 243:11, 246:13, 246:14, 285:18, 285:22 large 230:7, 234:7, 236:12, 322:1, 325:6 large-scale 321:7 larger 312:2 largest 322:3, 325:4 last 178:11, 181:14, 210:13, 218:15, 240:11, 259:21, 311:15 lasted 185:6 latency 291:12, 292:7, 292:8 later 181:16, 182:10, 183:2, 186:9, 203:15, 215:10, 227:9, 251:15, 323:17 law 176:23	laws 328:7 lawyers 195:8 lcd 182:20, 183:2, 183:4 learn 319:19 learning 186:25, 322:20 least 186:15, 189:18, 189:20, 191:15, 194:1, 200:4, 208:14, 235:13, 287:5, 287:14, 288:7, 299:13, 300:4, 309:8, 309:10, 309:12, 309:13, 313:10, 314:3, 317:17, 317:21, 320:3, 325:25 leave 259:2, 312:22 leaves 198:7, 199:14, 200:19, 202:9, 258:24, 303:20 lee 225:22, 320:21 left 245:25, 246:5 lengths 185:10 less 280:7, 295:20 let's 187:1, 188:16, 189:2, 193:11, 198:6, 202:6, 205:25, 206:2, 206:9, 210:11, 220:18, 220:25, 221:20, 223:3, 225:12, 228:25, 240:14, 241:2,	242:1, 249:7, 252:2, 253:8, 257:19, 258:13, 266:15, 269:20, 271:5, 273:3, 273:4, 274:23, 275:15, 283:11, 287:18, 293:16, 297:20, 299:13, 299:17, 301:9, 314:3, 315:12, 316:9, 317:18, 318:20 level 230:3, 238:4, 238:6, 254:21, 254:22, 256:16, 296:4, 314:18 levels 321:8 levin 175:14, 175:28, 177:19 light 210:20, 234:9, 263:13 lighted 265:7 lighting 182:4, 233:8, 235:2, 235:7, 261:9, 262:13, 262:15, 263:9, 263:24, 264:1, 264:2, 264:3, 264:11, 264:12, 264:14, 264:18, 264:20, 264:25, 265:4, 265:6, 265:13, 265:18, 265:20, 265:25, 266:3, 266:10 limitation 307:7 limitations 215:4, 217:15 limited 265:4, 267:22,
--	---	---	---

305:17, 306:11 line 223:18, 224:2, 224:9, 225:7, 225:25, 226:6, 226:8, 227:8, 227:22, 229:2, 242:5, 250:4, 250:10, 253:17, 255:24, 287:20 linear 232:15, 238:16, 238:20, 240:12 lines 225:13, 228:3, 229:1, 232:20, 242:16, 259:16, 278:11, 279:10, 280:16, 281:24 listed 307:18 literal 285:18, 285:22 litigation 195:8 little 179:3, 179:4, 179:15, 184:18, 192:6, 196:14, 214:15, 238:3, 252:6, 263:6, 264:8, 264:9, 266:23, 283:5, 309:5, 314:21 live 309:16 llp 175:4 ln 327:15 loaded 226:9, 226:22, 247:1, 248:2 local 227:10 located 234:17, 235:20, 288:16	location 223:17, 233:11, 235:8, 235:25, 284:15, 285:8 locations 234:16, 259:20, 285:15, 297:6, 314:14 logic 200:15, 200:20, 200:22, 201:2, 201:9, 299:5, 300:2, 300:16, 300:20, 301:1 long 203:20, 282:12, 294:24 look 182:6, 205:25, 207:14, 220:3, 222:21, 241:11, 242:24, 246:15, 249:7, 263:23, 264:1, 264:17, 265:25, 270:13, 277:3, 280:24, 280:25, 283:11, 287:18, 293:4, 299:13, 301:16, 303:9, 306:10, 309:6, 310:5, 310:9, 310:24, 311:3, 311:24, 312:11, 313:11, 314:22, 315:13, 325:10 looked 186:9, 228:4, 314:9 looking 202:22, 205:11, 232:20, 233:22, 242:18, 252:13, 258:4, 264:24, 265:5, 270:3, 272:22, 281:6, 282:9, 285:18, 285:24, 286:24,	288:19, 299:15, 303:23, 304:8, 308:9, 319:13 looks 196:25, 202:15, 243:8, 251:4, 319:5, 319:6, 319:7, 319:8 los 173:22, 174:23, 177:1, 328:3, 329:4 lost 180:23, 186:13 lot 184:8, 185:18, 186:6, 186:8, 186:12, 199:10, 253:2, 256:10, 303:4, 305:15, 306:13, 321:10 lots 324:8 low 256:4, 257:16, 296:4 low-power 321:22, 323:18, 323:21 low-priority 255:12, 256:11, 256:25, 260:17, 260:18, 260:24, 317:9 lower 252:20 lowest 256:2 M machine 186:25, 322:20 made 191:23, 196:10, 196:13, 197:15, 204:2, 206:14, 243:21, 255:16, 299:20, 303:5,	306:14 main 175:6, 183:10, 276:24, 292:1 maintain 242:7 major 182:12, 290:23, 323:24 make 178:10, 180:1, 195:9, 196:18, 203:1, 203:8, 203:9, 203:24, 207:11, 209:23, 217:25, 231:16, 243:10, 254:13, 271:25, 273:22, 274:14, 279:1, 282:4, 284:20, 288:5, 288:20, 289:10, 290:12, 309:6, 309:14, 310:10, 310:13, 311:1, 311:12, 312:23, 313:5, 313:6, 313:20, 315:3, 317:15 makes 202:17, 237:20, 242:20, 255:17, 255:19, 289:25, 291:7, 303:20, 304:22, 309:23, 311:15, 316:4, 318:17 making 206:19, 215:13, 215:19, 233:7, 236:15, 254:14, 276:24, 278:4, 304:9, 304:22, 309:15, 330:6 manager 223:12, 242:7, 242:12, 249:20, 250:2, 255:17, 255:18, 259:10,
---	--	---	--

314:9 manner 274:17, 289:19, 290:2 many 179:18, 185:9, 185:23, 278:20, 279:8, 307:17, 309:14, 309:18, 311:5, 321:4, 324:4, 324:5 mapping 264:22 mappings 279:12 mark 178:22, 193:11, 221:24, 222:13 marked 187:4, 187:8, 188:18, 193:10, 206:4, 222:7, 222:19, 269:22 marks 326:9 massoud 173:20, 174:20, 176:4, 176:10, 177:5, 178:5, 326:10, 327:12, 328:6, 328:17 material 208:17 matrix 300:13 matter 177:6, 189:4, 228:18, 228:20, 272:18, 289:5, 289:15, 321:1, 324:16 maximize 181:4 maximum 309:2, 316:23 maybe 183:10, 196:14, 202:16, 206:1,	209:14, 230:14, 230:15, 235:16, 246:16, 248:15, 252:7, 263:12, 265:3, 269:6, 280:20, 281:1, 293:6, 301:10, 306:11, 307:21, 310:21, 310:25, 311:1, 311:2, 312:1, 324:12 mean 184:15, 186:7, 192:9, 196:13, 196:23, 202:22, 208:5, 222:7, 224:16, 228:18, 231:2, 231:19, 231:22, 236:11, 241:7, 241:16, 243:3, 243:4, 244:16, 246:12, 254:11, 255:4, 256:17, 258:17, 267:13, 272:13, 275:10, 277:15, 294:17, 300:16, 300:23, 303:22, 306:7, 310:1, 316:24, 323:1, 323:8, 323:10 meaning 189:24, 190:22, 191:12, 194:23, 197:2, 204:23, 204:24, 205:2, 238:17, 254:24, 272:13, 278:15, 289:12 means 194:4, 196:25, 205:5, 211:6, 218:6, 229:14, 241:10, 245:23, 247:8, 248:6, 248:7, 248:12, 254:3, 254:6, 257:14, 258:21,	258:22, 260:3, 265:5, 297:3, 309:10, 310:11, 314:11, 314:17, 318:7 meant 190:7, 200:15 mechanism 244:4, 261:1, 261:6, 287:15, 310:8, 310:16, 310:20, 314:16 mechanisms 308:4 media 177:4 medium 202:16 mee 225:21 meet 215:22, 216:16, 217:15 meeting 180:8, 260:3 meets 249:11, 302:14, 307:7 memories 298:7, 323:4 memory 194:1, 194:15, 194:18, 202:16, 205:23, 206:15, 206:21, 206:24, 207:4, 207:8, 207:13, 209:6, 209:10, 209:14, 209:15, 212:17, 212:18, 212:20, 212:24, 214:3, 214:5, 214:8, 216:24, 223:6, 224:25, 227:11, 233:19, 234:5, 241:17, 242:2, 254:17, 258:7, 259:9, 259:13,	259:15, 259:18, 259:20, 260:13, 270:12, 270:17, 272:11, 274:6, 274:13, 274:17, 277:4, 277:19, 278:5, 278:17, 282:6, 282:16, 283:1, 284:11, 284:15, 284:16, 284:25, 285:1, 288:23, 289:2, 290:9, 290:10, 291:9, 293:5, 293:19, 294:2, 294:4, 294:8, 294:21, 294:23, 296:18, 296:19, 296:25, 299:1, 314:11, 314:15, 315:10, 321:13, 323:3, 323:9 mention 238:14 mentioned 186:6, 299:19 merely 189:21, 197:5 met 199:22, 298:21 method 309:24 methodologies 307:19, 321:23 methods 288:7 middle 245:25, 259:20, 261:5, 319:13 might 209:18, 230:8, 245:22, 261:25, 263:1, 304:18, 318:19 military 324:19 million 325:3, 325:5
---	---	---	---

mind 201:14, 208:16, 288:13, 301:11	296:21, 297:7, 297:16, 297:18, 303:13, 306:13, 309:5, 309:16, 312:1, 313:21, 322:16, 324:25	217:20, 218:11, 228:21, 236:7, 300:25, 301:7	266:23, 276:22, 283:10, 285:15, 302:4, 312:11, 313:5, 314:9, 316:13, 320:11
minimalist 272:4		myself 208:15, 252:7, 283:11	needed 184:8, 208:3, 230:16, 233:23, 293:2
minimize 272:4	morrison 175:26	N	
mintz 175:14, 175:27, 177:18	most 245:2	n 309:9	needs 216:1, 221:4, 310:5, 312:21
minute 221:14, 325:9	motivates 272:23	name 327:3	neighbors 224:24
missing 290:15	motivation 313:14	named 323:12, 324:10	neither 225:15
mistaken 281:1	mountain 175:17	nanoseconds 310:22	networks 186:20
modification 278:12, 288:21	much 180:5, 275:6, 277:23, 309:16, 311:9, 324:23	narrow 209:16	neural 186:19
modifies 267:11, 267:25	multi-processing 306:20	narrower 188:13, 192:22	never 194:11, 194:23, 208:9, 208:10, 208:16, 212:14, 212:20, 215:2, 283:3, 298:19, 304:3
modify 313:12	multiple 183:21, 226:24, 228:22, 239:5, 253:3, 277:2, 277:10, 283:20, 283:22, 303:8, 304:1, 311:11, 311:17, 311:24, 311:25	narrowest 192:11	new 183:13, 187:9, 207:11, 244:24, 251:12, 300:11
moment 187:5, 194:12	multiplexing 250:6	natalie 173:30, 174:26, 177:24, 329:6, 330:25	next 189:14, 226:11, 237:6, 243:7, 244:25, 245:8, 245:11, 261:6, 274:18, 275:3, 275:4, 280:12, 281:18, 292:20, 304:21
monday 173:23, 174:25, 177:2, 327:10	multiplication 300:14	nature 281:12, 282:20	
money 324:23	multiplying 300:9	necessarily 190:8, 190:18, 210:9, 211:18, 213:16, 236:7, 244:3, 248:7, 264:13, 265:19, 282:3, 300:24, 300:25, 303:11	
monitor 177:10, 220:21, 220:24, 257:22, 257:25, 298:1	multiprocessing 308:1	necessary 208:10, 208:21, 238:2, 320:9	
moon 312:6	multiprocessor 184:17	necessity 228:20	
more 182:21, 189:18, 189:19, 190:8, 191:16, 196:15, 207:1, 230:11, 232:4, 235:16, 238:4, 241:20, 251:2, 253:2, 266:5, 270:24, 272:13, 274:14, 280:1, 282:21, 289:10, 292:21, 293:17, 295:20,	must 189:20, 191:25, 206:11, 209:6, 209:7, 209:12, 209:18, 210:6, 212:24, 213:5, 214:4, 214:20,	need 178:21, 196:18, 201:24, 206:14, 207:1, 213:22, 214:16, 236:16, 244:12, 246:17, 248:18, 257:11,	

312:17, 313:2 noise 303:4 nominal 312:10 none 203:21, 211:20 noninfringement 194:21, 195:10, 208:6 nonresponsive 281:7 normally 184:15, 257:1, 309:8 notations 330:11 note 178:10, 178:23, 319:10 nothing 187:7, 200:1, 200:21, 203:17, 204:2, 204:4, 204:11, 205:8, 205:14, 205:17, 214:24, 265:6, 291:16, 292:3, 312:8, 326:5 notice 221:16 notion 200:4, 256:4, 257:16, 280:2 novelty 275:11 number 174:27, 177:5, 177:8, 179:7, 179:8, 180:24, 197:13, 198:15, 198:24, 218:23, 218:25, 232:2, 232:5, 234:8, 271:22, 305:17, 305:24, 306:7, 306:12, 308:4, 325:4, 327:6,	329:9 O oath 329:19 oaths 329:16 object 186:18, 237:11, 237:13, 262:6, 263:20, 263:21, 293:22 objected 269:3 objection 190:24, 191:18, 192:17, 195:4, 196:11, 196:21, 197:9, 198:11, 199:19, 200:24, 202:12, 204:20, 205:16, 205:24, 210:8, 211:1, 212:21, 212:22, 213:13, 214:1, 214:19, 215:5, 215:25, 216:4, 216:17, 217:16, 224:15, 224:22, 227:14, 231:13, 234:21, 235:12, 238:11, 244:2, 251:21, 256:8, 256:22, 259:22, 268:24, 270:6, 271:12, 272:25, 276:20, 278:13, 279:14, 281:25, 285:3, 288:25, 305:14, 314:2, 318:14, 319:21 objects 179:25, 181:25, 184:4, 237:15 obvious 191:3, 247:20, 270:10, 300:24, 306:16, 310:3	obviously 220:15, 244:21, 262:10, 277:4, 285:14, 305:22 occupied 232:7 occur 238:8, 239:20, 252:11 occurrence 266:3 october 176:17 offer 330:1 offering 330:4 office 173:1, 174:1, 177:8 officer 329:22 oh 183:9, 183:20, 222:5, 261:13, 287:1, 310:24 okay 179:2, 179:5, 179:6, 179:14, 181:20, 187:5, 187:10, 187:21, 187:24, 188:4, 189:2, 189:14, 189:24, 190:22, 191:9, 195:21, 196:5, 200:18, 202:6, 206:6, 213:9, 222:1, 222:15, 222:18, 222:20, 223:18, 224:1, 224:11, 224:25, 225:5, 226:20, 227:20, 232:20, 256:17, 266:15, 266:21, 267:6, 267:10, 267:21, 268:25, 269:7, 269:20,	269:24, 270:3, 271:5, 272:21, 275:15, 275:23, 285:6, 288:14, 291:24, 296:24, 297:20, 298:13, 298:16, 299:22, 301:3, 301:9, 301:21, 302:7, 302:13, 305:3, 311:14, 312:8, 318:4, 318:11, 321:24, 325:17, 326:4 oled 183:3 once 202:3, 247:1, 257:4, 312:6 one" 241:12 one-of-n 309:18 one-to-one 279:4 ones 241:20, 251:6, 281:17, 282:19, 311:4 only 186:23, 198:9, 211:7, 212:12, 212:19, 213:19, 214:7, 214:24, 214:25, 215:9, 215:12, 215:17, 215:20, 216:14, 216:15, 217:1, 217:4, 217:13, 217:25, 219:2, 220:10, 220:12, 226:19, 231:23, 232:6, 232:7, 232:12, 238:7, 254:14, 259:21, 260:7, 263:9, 265:5, 268:20, 269:10, 277:18,
---	--	---	--

278:4, 278:16, 282:4, 284:17, 290:1, 293:18, 296:19, 296:24, 300:15, 302:19, 307:9, 309:24, 311:18, 312:5, 312:19, 316:24 ooo 326:14 open 221:3, 284:1, 296:18, 296:20, 297:14 opening 196:1 operable 206:11, 211:3, 214:20, 215:20, 215:21, 216:1, 216:6, 216:10, 216:15, 216:24, 217:10, 217:14, 217:20, 217:22, 218:3, 218:12, 219:1, 220:8 operated 303:14 operating 319:19, 322:20 operation 212:11, 245:24, 246:3, 300:14 operations 224:6, 224:7, 244:11, 274:20, 300:5, 300:20, 301:4 operative 194:2, 194:3, 214:8, 214:10 opine 208:3, 229:25, 230:2 opined 194:14, 262:19, 283:3 opinion 190:25, 194:11,	195:14, 256:6, 267:10, 267:25, 287:7, 325:24 opinions 188:6, 193:20, 193:22, 207:20, 267:23, 298:5 opposed 293:19 optimization 291:6 optimizations 181:4, 292:5, 296:9, 321:18 optimize 294:11 optimized 294:19 option 301:8 options 306:20, 306:25 order 176:15, 189:3, 198:16, 198:17, 199:15, 200:20, 221:4, 227:11, 227:24, 228:6, 228:12, 228:16, 229:12, 229:22, 231:6, 250:21, 259:1, 259:2, 260:7, 279:19, 290:15, 305:12, 325:3 ordering 228:19 orders 178:18 ordinary 325:21, 325:25 organize 239:2 organizing 273:25 orrick 175:4, 177:20 other 180:4, 183:12,	185:8, 189:21, 195:1, 199:1, 203:1, 203:15, 207:8, 207:16, 209:8, 209:9, 213:7, 216:11, 216:14, 216:16, 216:18, 216:22, 228:20, 230:21, 230:23, 232:9, 232:10, 236:22, 239:7, 239:13, 242:16, 245:7, 250:25, 252:22, 252:24, 254:5, 254:15, 254:17, 261:7, 263:2, 264:9, 267:16, 268:16, 271:17, 273:12, 273:23, 274:2, 274:22, 279:9, 284:20, 285:21, 285:22, 287:24, 289:18, 292:18, 293:3, 294:3, 294:6, 296:7, 301:8, 303:17, 306:4, 308:18, 310:14, 312:24, 314:20, 315:7, 318:8, 318:21, 326:4 others 257:5 otherwise 179:5, 202:2, 301:1 out 181:15, 191:6, 198:16, 200:3, 203:15, 205:6, 206:24, 230:16, 248:24, 259:24, 260:7, 260:19, 276:7, 286:6, 305:22, 305:23, 306:25, 313:21 output 224:25, 227:10,	298:6, 298:14 outside 248:9 over 179:19, 179:25, 185:22, 279:12, 280:18, 281:4, 281:11, 284:2, 294:17, 321:3, 322:9 overhead 290:8 overlay 179:25, 184:4 overlying 181:25 overwriting 294:25 own 273:16, 288:9, 290:17, 293:11, 319:7 owner 173:13, 174:13, 174:21, 175:13, 177:19, 187:18 P pacific 177:10, 326:12 packed 296:4 page 176:3, 176:9, 189:6, 189:14, 191:14, 192:15, 193:18, 193:19, 193:23, 193:24, 223:17, 223:18, 224:2, 224:9, 225:13, 225:23, 225:25, 227:8, 228:4, 229:1, 229:11, 230:24, 232:20, 234:15, 242:4, 242:9, 242:15, 242:17, 246:19, 246:20,
---	---	---	---

250:4, 250:10, 250:14, 251:25, 252:4, 253:17, 255:24, 256:12, 258:5, 259:16, 283:13, 287:18, 318:5, 319:13 panel 249:2 papers 180:24, 322:17, 323:4, 323:7 paragraph 188:4, 188:5, 188:8, 191:21, 223:5, 234:15, 240:15, 249:9, 258:5, 261:13, 266:15, 267:8, 267:13, 267:20, 267:22, 268:18, 268:20, 269:10, 299:23, 300:3, 301:12, 302:2, 325:11, 325:18, 325:20, 326:1 paralegal 175:27 parallel 181:3, 181:18, 181:19, 185:8, 185:10, 185:18, 226:19, 238:25, 245:4, 245:8, 245:12, 251:6, 254:7, 254:18, 279:4, 283:1, 314:12, 321:11, 321:22, 323:1, 323:9, 323:22 part 185:2, 186:5, 189:20, 191:19, 191:22, 225:3, 225:20, 233:18, 234:3, 234:14, 245:17, 248:12, 251:23, 257:13,	257:15, 259:10, 259:21, 261:12, 262:3, 263:24, 269:6, 275:24, 277:18, 299:1, 301:18, 307:12, 313:10, 313:14, 317:17, 319:15, 330:3, 330:14, 330:18 partes 176:12 partial 194:6, 194:17 participates 268:6 particular 191:4, 191:22, 207:12, 207:18, 213:23, 219:5, 220:4, 220:5, 227:4, 228:7, 228:17, 231:1, 231:20, 234:25, 242:24, 258:2, 284:25, 285:8, 286:25, 296:25, 298:6, 299:6, 304:16 particularly 212:4 parties 177:13, 329:14, 330:4, 330:7 partition 230:8 parts 270:4, 274:6 party 330:3, 330:12, 330:13, 330:14, 330:17, 330:18 party's 330:2, 330:13 parvizi 177:25 parvizi-azad 173:30, 174:26,	329:6, 330:25 pass 217:5 passed 225:10, 237:19, 244:6 passes 202:18 past 264:8, 264:9 patent 173:1, 173:2, 173:9, 173:13, 174:1, 174:2, 174:9, 174:13, 174:21, 175:13, 176:13, 176:25, 177:8, 177:19, 179:8, 180:24, 186:10, 187:17, 188:2, 188:11, 189:8, 194:24, 196:6, 205:22, 206:6, 212:4, 219:24, 223:7, 258:7, 298:8, 299:9, 302:8, 307:8, 321:2, 322:25, 323:14, 326:2, 327:8 patents 321:2, 324:10, 324:11, 324:16 path 272:17 paths 289:12 pay 322:11, 322:12 pdf 187:8 pe 225:21, 225:22, 226:10 pedram 173:20, 174:20, 176:4, 176:11, 177:6, 178:5,	178:17, 179:2, 188:1, 193:16, 201:19, 204:19, 206:7, 219:12, 222:24, 258:4, 269:24, 284:22, 298:7, 320:17, 326:6, 326:10, 327:12, 328:6, 328:17 penalty 178:6, 328:7 pending 320:9, 324:12 people 262:1, 322:7 perform 199:25, 230:21, 241:5, 263:4, 290:17, 301:4 performance 185:19, 267:16, 272:17, 296:7 performed 196:17, 246:22, 264:14, 264:19, 290:22, 303:12, 319:24 performing 198:4, 300:20 performs 240:24 perhaps 181:12, 224:23, 261:21, 265:8 period 203:3, 257:10 periodic 311:8 perjury 178:6, 328:7 permanently 221:5 person 197:2, 204:22, 205:4, 205:10, 208:14, 260:25, 263:3, 268:9,
--	--	--	---

273:16, 277:3, 282:9, 295:16, 306:8, 306:16, 307:24, 308:8, 308:24, 309:4, 310:4, 322:6, 325:21, 325:25 personal 330:15 personally 183:24, 186:2 perspective 195:17, 235:23, 239:12, 304:6, 312:24 pertaining 287:22, 287:23 pes 225:21, 226:13, 227:10 petition 176:12 petitioner 173:7, 174:7, 175:3 pg 327:15 ph 173:20, 174:20, 176:4, 178:5, 327:12, 328:6, 328:17 phase 192:24, 232:13 phd 321:9, 322:3, 322:4 phonetic 185:10, 209:4, 296:4, 310:18 phrase 196:16, 214:15, 285:23 phrased 304:11 physical 321:18 physically 271:24	pi 180:2 pick 217:1, 217:4, 261:3, 271:17, 282:17, 309:11, 311:2, 323:6 picks 218:19 piece 255:14 pipeline 232:16, 236:3, 240:12, 272:12, 272:18 pixel-related 281:16 pixel-type 283:17 pixels 234:8, 238:8, 240:7 place 198:21, 199:8, 206:22, 209:21, 247:10, 255:22, 268:20 placed 271:23, 284:9, 284:10, 288:22, 329:19 placement 321:19 places 185:8, 242:16 planet 177:13, 177:25 plays 268:7 please 177:15, 178:2, 178:19, 187:3, 188:17, 193:12, 199:11, 200:17, 202:2, 221:1, 246:18, 258:2, 266:19, 266:24, 269:5, 282:1,	289:21, 298:3, 302:5, 317:23 pluralities 215:7, 216:10 plurality 189:10, 190:9, 190:10, 190:11, 194:2, 194:4, 206:12, 206:13, 211:4, 211:5, 212:25, 213:1, 213:6, 213:7, 214:9, 214:10, 214:17, 214:18, 214:21, 214:22, 215:1, 215:2, 215:12, 216:7, 216:8, 216:11, 216:25, 217:21, 218:2, 218:5, 218:6, 218:16, 218:19, 218:25, 220:7, 283:19, 301:24, 302:10, 306:25 plus 225:1, 267:3, 275:4, 275:5, 276:6, 292:21, 298:23 point 181:14, 191:1, 191:21, 194:5, 195:17, 196:4, 199:9, 199:17, 200:22, 202:10, 203:24, 205:20, 206:18, 207:12, 207:17, 207:23, 212:9, 213:23, 215:6, 227:2, 229:7, 244:18, 245:24, 246:5, 246:6, 250:25, 253:13, 258:10, 261:9, 261:12, 264:20, 267:7, 268:14, 268:15,	274:15, 280:8, 281:20, 287:14, 298:6, 301:22, 305:21, 313:8, 317:1, 322:7 pointer 275:2 pointers 274:24 pointing 249:4 points 194:15, 263:16, 263:19 policies 314:23 policy 309:18, 319:7 polished 263:22 polling 311:9 polling-based 310:8, 310:15, 310:20 pollution 186:19 polygon 263:13, 263:16, 263:19, 264:6, 264:7 popeo 175:15 portion 194:1, 194:3, 214:8, 214:10, 236:14, 236:19, 236:22, 237:1, 237:3, 273:18, 273:19, 274:1, 274:13, 282:16, 284:11, 284:19, 284:20, 284:25, 285:21, 289:21, 289:23, 290:11, 290:25, 291:8, 292:13, 292:14, 293:5, 293:6,
---	---	--	---

293:9, 294:2, 294:4, 294:14, 294:15, 297:5, 314:11 portions 207:6, 213:6, 214:11, 224:13, 239:19, 273:9, 284:18, 285:19, 285:25, 288:22, 292:1, 293:8, 293:19, 294:21, 315:10, 315:11 pose 201:13 posita 230:19, 264:17, 264:24, 265:9, 270:9, 271:20, 272:6, 272:21, 273:23, 278:8, 278:16, 285:24, 287:4, 288:19, 291:15, 300:23, 315:20 position 208:7 positions 194:21 possibilities 271:18, 272:19, 278:20, 279:8, 287:3, 307:17 possibility 199:2, 228:21, 236:25, 237:21, 247:12, 278:22, 278:25, 304:10, 305:5, 305:10, 311:10 possible 194:14, 198:21, 203:1, 209:1, 227:6, 230:16, 232:2, 236:3, 237:18, 238:24, 239:9, 262:13, 263:8, 265:25,	280:13, 307:18, 308:11, 308:15, 308:22 possibly 262:15 post-docs 183:11 postdoctoral 322:5 postgraduate 183:11 potential 184:7, 265:20 potentially 230:11, 237:23, 241:20, 247:9, 248:24, 251:11, 261:4 powell 320:21 power 181:16, 296:4 practice 195:13, 262:25, 312:4 practiced 238:22, 291:14, 308:7 preallocate 278:23 precise 197:10, 245:1, 292:22 predict 278:14 preempt 248:25 preemption 247:11 preemptive 247:11 preferably 283:14 preliminary 187:18 prepares 226:1 present 175:24, 201:11,	214:5, 296:8, 322:4, 330:12 presented 204:8, 204:9, 204:12, 234:24 presided 189:1 presumes 296:11 previewed 178:10 previous 212:22, 222:13, 256:9 previously 255:25, 259:9 primitive 226:2, 227:10, 227:23, 229:18, 230:5, 230:9, 231:4, 234:25, 235:4, 235:10, 235:15, 263:14 primitives 230:9, 234:17, 235:19, 237:7, 238:10, 239:22 principle 180:3, 183:9, 183:20 prior 191:2, 191:8, 200:2, 206:16, 208:15, 208:23, 238:14, 252:11, 319:23, 329:18 priorities 198:3, 207:9, 214:23, 242:13, 247:23, 302:17, 304:17, 304:18, 307:11 priority 197:14, 197:24, 247:17, 248:1, 249:5, 249:12, 249:16, 250:8, 250:20, 250:21,	252:13, 252:20, 254:23, 255:2, 255:6, 255:11, 255:15, 256:1, 256:2, 256:5, 256:7, 256:15, 256:20, 257:3, 257:17, 260:17, 260:24, 304:23, 317:10, 318:1, 318:6, 318:8, 319:6 probability 311:10 probably 217:17, 225:3, 280:7 problem 201:3, 201:6, 201:7, 219:22, 291:25 proc 329:15, 329:20, 329:26, 330:7, 330:19 procedure 305:10, 329:18 proceeding 187:18, 188:2, 329:24, 330:5, 330:13 proceedings 207:23 process 185:17, 186:4, 196:19, 211:25, 218:12, 219:2, 220:8, 223:13, 225:4, 227:2, 229:7, 231:17, 234:19, 236:15, 240:5, 241:24, 252:6, 253:1, 262:15, 263:12, 295:7, 304:15, 306:24 processed 203:5, 203:18,
---	--	--	---

226:3, 229:8, 232:23, 233:10, 235:17, 236:1, 241:22 processes 280:9 processing-type 224:17 processor 241:23, 244:23, 245:10, 245:16, 246:4, 247:18, 250:11, 250:24 processors 240:23, 242:6, 242:12, 244:1, 244:4, 244:9, 244:15, 244:16, 244:21, 245:6, 246:21, 247:1, 248:3, 249:15, 259:12, 260:15, 314:10, 316:2 produce 264:19 product 330:9, 330:17 products 330:2 prof 329:11 professor 320:20 program 233:19, 237:19, 242:8, 250:8, 274:21, 274:22, 295:25, 323:13, 324:5 programmability 296:6 project 180:3, 182:8, 183:8, 183:19, 184:20, 185:4, 186:6, 324:18, 324:24 projects 179:18, 184:21,	185:6 proper 305:12, 308:13 properly 229:16 properties 261:20 property 263:21 proportions 284:14, 313:12 propose 270:4 proposing 271:8 prosecution 190:17 protective 178:18 protocol 305:10 provide 192:19, 193:8, 196:14, 204:25, 229:21, 272:16, 283:25, 289:1, 289:22, 291:9, 291:11, 292:6, 309:1, 330:1, 330:9 provided 193:1, 210:14, 227:24, 228:6, 228:11, 228:16, 231:5, 267:24, 297:14, 310:2, 317:23, 330:17 provides 204:5, 228:5, 229:3, 268:9, 298:24 providing 190:4, 273:24 ptab 205:3, 277:24, 278:2 public 178:13	publications 309:18 publish 322:14 published 322:16, 322:24, 323:8 pulled 259:24 pulling 222:1 purpose 184:22, 186:24 purposes 188:12, 267:12 pursuant 329:17 pushed 259:24 put 191:20, 192:11, 203:19, 203:20, 225:9, 244:14, 273:8, 276:9, 301:11, 309:21, 318:20, 321:7 puts 302:22 putting 180:12, 273:21, 284:24, 285:7 Q quadrant 236:13, 236:17 qualified 329:7 quality 180:20, 186:12 questions 202:13, 253:15, 326:4 queue 260:19, 304:3 quick 178:10 quickly 221:10, 251:3,	295:13 quite 180:11, 184:12, 186:21, 186:22, 194:14, 195:6, 218:14, 247:15, 253:7, 277:1, 323:11 quote 189:16, 189:17 R radar 184:25, 248:9 raise 178:2 raised 207:24, 208:2, 208:10, 212:10 raising 206:18, 208:18 random 274:17, 275:13, 297:9 random-access 314:16 randomly 297:6 randomness 297:10 range 296:20, 321:6 range-finding 184:5 ranging 321:16 rash 309:15 rastering 240:2 rasterization 240:2, 240:5, 240:9 rather 193:22, 242:1, 267:2, 298:3, 304:15 rdokhanchy@mintz 175:21
--	---	--	---

re-mark 222:10 reactive 310:18 read 190:23, 213:20, 215:18, 217:7, 219:23, 220:15, 225:21, 246:19, 250:15, 251:25, 252:6, 252:7, 253:4, 253:5, 253:14, 267:14, 282:6, 292:14, 293:6, 293:9, 327:15 reader 303:21 reading 210:9, 211:2, 218:14, 218:15, 250:16, 293:15 readout 313:15, 313:16, 313:18 reads 211:6, 242:5, 249:19, 256:13, 313:21, 327:15 ready 242:14, 247:7, 248:16, 248:19, 248:21, 248:23, 249:6, 249:23, 251:18, 252:12, 252:13, 253:13, 253:19, 253:24, 254:9, 255:10, 256:1, 256:3, 256:7, 256:20, 257:11, 257:13, 257:14, 303:19, 308:5, 316:11, 316:23, 316:24, 317:1, 317:6, 317:11, 317:20, 317:22, 318:8, 318:9, 319:6,	320:10 real 179:25 reality 182:21 realize 306:11, 320:12 really 182:9, 198:22, 206:25, 238:24, 277:18, 295:12, 307:23, 312:11, 324:6 realtek 173:5, 174:5, 175:3, 177:6, 177:21, 188:24, 195:1, 195:21, 327:3 realtek-amd_ohs@- orrick 175:11 realtime 197:23, 197:25, 257:8 reap 277:8 reason 179:11, 203:2, 203:6, 217:5, 247:6, 248:13, 251:9, 260:12, 292:1, 313:10, 316:16 reasoning 319:17 reasons 198:15, 267:17, 271:22, 277:2, 310:3, 327:15 recalculates 252:9 recall 207:21, 220:2, 235:14, 251:23, 255:4, 255:8, 255:13, 261:2, 280:19, 281:21,	289:20, 307:20, 307:21, 308:16, 315:17, 318:15, 318:20, 324:21 recalled 308:8 receive 184:4, 202:21, 218:12, 219:2, 220:8, 223:11, 251:18, 260:10 received 186:11, 197:8, 199:18, 200:9, 200:23, 202:11, 219:3, 232:21, 232:22, 290:20 receiver 200:8 receives 197:5, 197:13, 197:16, 198:10, 220:10, 220:13, 249:21, 284:4 receiving 199:24 recently 185:4 reception 200:10, 219:4 recess 220:22, 257:23, 297:24 recognition 186:18 recognize 217:24 recognizing 218:7 recollection 319:2 recommended 176:20 reconfigurable 282:14 record 178:11, 191:21, 220:21, 220:24,	257:20, 257:22, 257:25, 297:23, 298:1, 326:11, 329:25 recorded 329:23 recording 233:23 recover 180:23, 186:12 redacted 178:14 redactions 221:4 redirect 320:9 reduce 272:16 redundancy 291:20 refer 187:9, 242:1 reference 176:28, 176:30, 200:13, 222:25, 227:9, 232:24, 233:6, 233:12, 233:15, 234:5, 235:5, 238:23, 246:18, 250:9, 251:24, 259:16, 262:3, 264:17, 264:25, 265:12, 267:2, 268:11, 269:25, 270:13, 276:25, 278:10, 317:9 references 191:2, 208:15, 242:9 referring 188:22 refers 227:16, 266:25 reflection 263:15 reflections 263:13, 263:23
--	--	---	---

refresh 221:12 refreshing 288:13 regard 280:17, 301:21 regarding 179:7, 179:8, 186:4, 188:2, 188:11, 189:3, 208:22, 211:25, 219:25, 240:15, 247:3, 330:11 regardless 191:4, 195:13 region 227:11, 227:13, 227:15, 227:16, 227:23, 229:19, 230:5, 230:10, 231:4, 231:23, 231:25, 232:1, 235:10, 235:15, 235:18, 236:11, 236:17, 236:18, 238:10 region-based 227:25, 228:13, 229:13, 229:17, 229:22, 230:20, 230:25, 231:6, 231:7, 231:11, 231:17, 231:18, 232:1, 232:8, 232:11, 235:11, 236:21, 238:7, 239:19, 239:23 regions 227:5, 228:7, 228:17, 228:23, 230:8, 230:9, 230:11, 230:15, 231:20, 232:4, 232:5, 232:6, 232:12, 236:20 regret 309:16 regular 310:8, 310:20,	311:8 regularly 321:4 relate 321:1, 322:24, 324:13 related 184:1, 214:3, 290:24, 291:2, 323:13, 324:16 relates 191:23, 269:8, 277:18, 284:6, 319:16 relating 269:9 relative 207:9, 214:23, 242:13, 256:15, 307:11, 329:13 relayed 287:13 release 257:4 releases 304:21 relevant 209:22, 280:7, 319:23 rely 222:25, 269:25 remaining 232:9 remains 303:22 remedy 176:21 remember 179:20, 180:8, 182:4, 183:13, 191:20, 255:20, 255:21, 279:23, 280:22, 282:1, 287:15, 304:11, 316:22, 317:7, 318:17, 319:2, 325:1 remind 266:24, 283:10	remote 177:12 remotely 173:22, 174:22, 177:1, 177:14 remove 221:5, 222:12 removing 221:18 render 191:3 rendered 247:20 rendering 207:19, 230:6, 261:18 repeat 217:17, 251:2, 302:4, 302:5 repeating 229:15 rephrase 290:13, 313:23 replace 244:24, 251:10, 251:11, 252:20, 254:23, 259:14, 260:17, 260:23, 268:8, 271:10, 275:19 replaced 201:1, 254:25, 258:11, 260:13 replaces 247:17, 247:18, 252:23 report 191:20, 191:22, 192:5, 192:8, 192:11, 196:1, 222:25, 240:21, 242:1, 246:20, 249:7, 266:16, 268:21, 269:19, 269:25, 299:14, 299:23, 300:3, 301:14, 301:17 reported 173:30	reporter 174:27, 177:24, 181:10, 324:1, 329:8 reporters 329:10 reports 207:21 represent 177:16 representing 177:13, 177:25 requesting 312:18, 313:4 require 196:6, 196:9, 212:4, 265:21 required 195:19, 207:11, 207:19, 211:20, 232:24, 240:25, 241:6, 267:16, 277:15, 284:17, 290:8, 293:18, 293:23 requirement 191:13, 192:14, 192:25, 193:1, 193:5, 194:13, 194:16, 208:21, 209:18, 210:21, 210:22, 211:13, 211:17, 213:2, 213:10, 213:11, 213:15, 214:2, 214:7, 214:13, 214:24, 216:6, 216:9, 216:12, 249:12, 284:16, 302:15, 307:10 requirements 209:7, 209:8, 209:9, 216:22 requires 189:16, 190:17, 191:14, 258:7, 299:7, 300:14 reschedule 252:17
---	--	--	--

research 179:18, 180:2, 182:12, 184:21, 185:5, 321:24, 322:6 reside 206:21 residing 207:13, 209:13, 209:14, 244:18, 254:16 resolution 234:2 resolved 308:2 resonate 314:21 resource 197:15, 198:4, 257:4, 257:6, 306:11, 308:1, 308:6, 308:20, 312:13, 313:3, 313:4 resources 311:6 respect 192:18, 195:10, 196:14, 208:6, 208:7, 232:17, 267:20, 267:23, 268:18, 277:19, 278:7, 281:20, 298:25, 299:6, 299:10, 302:18, 303:6 respectfully 190:15, 194:19, 201:5, 239:15, 265:16 respond 192:1, 197:11, 198:12, 207:16, 207:24, 208:18, 260:12, 266:9, 314:20 responded 227:15	respondent's 178:14 respondents 188:23, 195:1 response 187:14, 187:17, 187:18, 192:20, 192:21, 193:8, 243:2, 265:23, 314:18 responsible 183:18, 186:2, 227:4 responsive 266:12 rest 252:7, 273:22 restoring 282:6 restricting 299:12 restriction 295:6 restrictive 204:1 result 235:9 results 309:7 retiring 279:17 rev-1 310:18 review 176:13, 208:11 reza 175:16, 177:18 right 178:3, 181:23, 182:17, 183:21, 184:2, 185:22, 187:1, 187:19, 187:25, 188:16, 188:24, 189:2, 189:4, 190:20, 193:11, 196:7, 199:21, 206:9, 209:1, 209:3,	210:11, 217:3, 217:5, 219:7, 220:18, 220:25, 221:11, 223:3, 223:14, 223:16, 229:17, 230:1, 230:4, 233:13, 237:5, 237:7, 239:14, 240:24, 241:12, 242:21, 243:22, 246:1, 246:2, 246:5, 246:25, 248:10, 254:10, 254:20, 255:17, 257:19, 258:1, 259:3, 260:3, 268:2, 271:5, 282:18, 282:19, 284:12, 294:10, 294:23, 295:4, 295:22, 297:20, 298:2, 298:9, 298:18, 299:8, 300:6, 300:17, 301:15, 302:11, 302:23, 304:17, 307:13, 308:14, 310:6, 310:7, 310:13, 311:21, 315:11, 315:25, 316:11, 316:17, 317:1, 317:7, 318:3, 320:8, 322:12, 322:13, 323:16, 326:7 road 175:17 robin 306:1 robust 322:9 role 183:7, 197:6, 241:23, 245:13, 268:7 round 306:1	routed 291:3 routinely 195:12 routing 321:19 row 242:17 rpr 173:30 rsr 173:30 rulings 193:22 run 232:10, 245:3, 247:10, 248:14, 257:14, 261:6, 312:9, 317:1, 317:20, 320:1 running 237:14, 237:23, 244:8, 247:6, 247:13, 247:14, 248:6, 248:13, 248:17, 248:22, 248:23, 251:7, 252:9, 252:22, 254:17, 317:20 S said 197:6, 199:12, 217:18, 219:15, 227:15, 239:25, 241:10, 243:3, 245:18, 265:14, 269:2, 273:6, 282:8, 297:11, 298:12, 298:19, 299:10, 300:3, 303:1, 303:3, 308:11, 311:15, 311:17, 315:19, 317:25 same 185:11, 192:23, 198:16, 214:12,
---	---	--	---

221:16, 228:14, 230:21, 232:4, 232:19, 237:3, 237:15, 237:24, 256:22, 259:2, 259:8, 259:17, 260:16, 266:22, 267:7, 267:17, 270:19, 275:21, 277:7, 279:21, 281:5, 285:16, 290:25, 291:2, 291:3, 291:4, 291:11, 291:22, 294:5, 295:17, 295:18, 296:12, 303:14, 304:7, 313:3, 314:5, 318:9, 330:4, 330:6 san 175:19 satisfies 216:21 satisfy 209:6, 209:8, 298:8 save 221:4, 221:18 saw 229:11, 318:5 say 188:8, 188:20, 188:21, 191:23, 192:8, 192:9, 192:10, 200:5, 200:6, 201:23, 205:7, 209:20, 212:13, 217:7, 219:14, 231:18, 242:10, 243:20, 248:19, 250:5, 257:13, 257:18, 258:19, 268:19, 268:21, 275:6, 282:12, 292:10, 297:9, 297:10, 298:11, 303:17, 304:19, 305:4, 307:6, 308:17, 312:12, 313:10, 314:23, 316:15, 318:22, 320:3 saying 181:15, 181:22, 202:25, 203:8, 203:23, 204:6, 204:8, 208:8, 208:9, 216:5, 216:19, 219:13, 224:19, 226:21, 227:1, 228:3, 228:14, 242:19, 243:8, 248:14, 253:1, 262:19, 266:2, 266:5, 287:1, 293:17, 293:21, 293:24, 297:16, 297:18, 311:15, 313:20 says 197:18, 197:19, 221:3, 227:12, 230:24, 231:3, 243:13, 245:25, 246:14, 249:14, 252:17, 253:17, 255:24, 283:14, 287:1, 287:20, 318:3 scale 325:6, 325:7 scaling 186:10, 324:14 scenario 198:9, 200:18, 202:8, 274:3, 275:16, 294:19, 296:17, 296:18, 296:21, 297:13, 297:14, 297:16, 297:18, 307:15, 308:10, 308:15, 309:19, 311:18 scenarios 311:16	scene 179:25, 233:18, 233:21, 234:4, 236:8, 237:4, 237:12, 237:13, 237:15, 240:4 schedule 243:15, 249:16, 259:12, 317:4 scheduler 240:22, 241:1, 241:5, 241:15, 241:19, 242:11, 242:20, 242:23, 243:9, 243:12, 243:14, 243:18, 245:15, 246:10, 247:2, 247:17, 248:1, 248:10, 248:24, 249:11, 249:15, 249:24, 250:12, 250:20, 251:4, 251:10, 251:13, 252:8, 252:16, 254:14, 256:1, 268:4, 270:21, 276:5, 276:6, 276:15, 277:14, 277:16, 278:18, 279:2, 279:6, 282:14, 286:16, 287:13, 298:24, 303:23, 306:15, 308:3, 312:15, 314:10, 315:15, 315:22, 316:2, 316:4, 316:6, 316:18, 317:4, 317:14, 317:23, 317:25, 318:11, 318:23, 319:18, 319:25 scheduler's 245:12 schedules 242:13 scheduling 250:6, 252:10, 277:21, 306:8, 306:19, 308:21, 308:23, 309:17, 312:21, 314:23, 323:2 sciences 320:22 scope 209:17, 220:5 scratch 184:11, 184:13 screen 182:13, 182:16, 182:18, 182:19, 182:20, 182:22, 187:13, 187:25, 221:17, 227:5, 227:16, 232:24, 233:6, 233:11, 233:22, 233:24, 234:16, 235:1, 236:11, 240:5, 240:7, 248:9, 299:22, 300:12, 301:12, 325:15 scroll 206:9, 299:14 second 194:3, 214:10, 222:16, 223:10, 225:12, 237:13, 263:24, 275:7, 283:18, 307:16 second-order 295:3 section 176:19, 192:7, 193:2, 249:7, 252:4, 253:15, 253:16, 259:25, 329:18 see 182:6, 189:7, 189:12, 189:15, 189:22, 194:7, 194:9, 208:1, 209:24, 210:1, 210:12, 210:17,
--	---

210:19, 212:2, 221:9, 221:11, 221:13, 227:8, 228:1, 232:25, 235:5, 240:17, 241:17, 243:22, 244:17, 252:3, 253:1, 255:1, 261:13, 269:4, 275:22, 286:22, 288:18, 291:24, 299:10, 310:9, 310:24, 315:24, 319:14 seem 267:4 seemingly 198:25 seems 279:23, 283:4 seen 195:7, 195:14, 199:3, 207:22, 318:25 sees 182:1, 287:12 segmented 225:8 segments 225:8 segregate 274:6, 290:8 select 189:9, 190:7, 196:6, 196:9, 196:16, 197:7, 198:10, 201:20, 201:22, 202:1, 205:15, 206:11, 207:8, 211:3, 211:18, 211:19, 213:11, 213:17, 213:19, 214:17, 214:20, 214:25, 215:2, 215:7, 215:9, 215:11, 215:20, 215:21, 216:2, 216:6,	216:10, 216:15, 216:24, 217:11, 217:14, 217:20, 217:23, 218:3, 241:10, 247:10, 250:21, 260:9, 298:13, 298:17, 304:16, 305:12, 306:22, 307:2, 311:25, 312:1 selected 191:25, 199:17, 200:22, 202:11, 210:14, 218:24, 246:10 selecting 192:13, 196:20, 196:24, 197:21, 197:25, 200:1, 217:10, 218:4, 240:25, 241:6, 301:22, 302:9, 302:14, 307:6, 307:9, 307:10, 307:13 selection 189:16, 189:19, 191:14, 197:16, 197:20, 198:1, 198:20, 202:17, 203:4, 203:24, 205:19, 205:21, 206:14, 206:19, 206:22, 207:11, 209:12, 213:18, 242:20, 243:10, 243:14, 247:25, 249:12, 251:14, 252:18, 298:21, 302:19, 304:22, 304:23, 304:24, 306:14, 306:15, 307:18, 308:4, 309:23, 309:25, 311:21, 312:25, 313:1, 313:5, 313:7, 318:16, 318:17, 318:22,	319:3 selections 198:2, 199:8 selector 308:3 selects 197:1, 208:22, 211:7, 211:8, 212:12, 212:14, 216:14, 298:19 semiconductor 173:5, 174:5, 175:3, 177:6, 327:3 semiform 253:20 send 218:22, 219:21, 241:21, 251:6, 274:10, 282:18, 289:12, 290:20, 299:2, 314:12 sending 197:25 sends 211:9, 218:20, 218:24, 248:25 seniority 324:7 sense 200:12, 200:13, 244:10, 247:6, 273:23, 284:21, 290:12, 313:2 sent 245:13, 250:2, 284:2, 294:16 sentence 210:14, 223:10, 228:14, 229:15, 267:19, 285:4, 286:10 separate 274:1, 315:11 sequence 244:14, 245:16 sequential 273:15, 274:21,	275:13, 294:7, 314:17 sequentially 274:25, 289:10 series 258:12, 259:15 serve 305:25, 308:21 served 323:15, 323:16, 324:4 service 310:14, 330:9, 330:17 services 330:2 serving 272:10 set 202:16, 204:1, 225:9, 238:8, 239:3, 241:14, 253:8, 273:13, 274:11, 291:3, 309:22, 312:2 setting 302:18 several 188:9, 189:7, 305:5 shader 184:22 shading 233:8, 235:2, 235:7, 261:10, 264:18, 264:21, 265:1, 265:13, 265:19, 265:21, 266:1, 266:3, 266:10 shall 330:1, 330:9, 330:15 share 221:17 shared 307:25, 308:6, 308:19, 311:13
---	---	---	--

sheet 327:1 shorthand 174:26, 329:7 should 184:7, 198:22, 208:25, 217:23, 220:3, 221:10, 221:23, 246:7, 249:25, 252:5, 252:9, 253:14, 261:6, 263:15, 280:20, 280:25, 286:14, 318:19, 319:16, 327:15 show 191:19, 270:9, 276:22, 276:25, 285:15, 289:21 showcase 183:15 showing 192:23, 283:25 shown 241:18, 243:15, 315:19 shows 316:23 sic 206:6, 216:8, 229:2 side 293:4, 293:13, 293:14, 293:15, 293:16, 312:23, 313:15, 313:16, 313:19 sides 236:22, 295:17, 295:18, 313:17 signal 184:23, 184:25, 186:13 signature 327:30 signature-plkal 330:23 silent 286:21, 287:4,	287:7 simd 239:2, 239:4 similar 185:15 similarity 185:19, 306:5 similarly 218:23 simple 198:25 simplicity 314:3 simply 191:11, 192:12, 220:6, 235:16, 239:17, 250:17, 304:14, 308:13 simultaneous 211:23 simultaneously 211:15, 214:5, 242:8, 304:7 since 186:15, 227:23 single 189:17, 191:15, 210:2, 234:4, 265:6, 265:7 sir 179:10, 187:23, 188:3, 188:15, 189:5, 189:13, 193:9, 193:21, 194:9, 196:4, 196:8, 201:5, 208:5, 210:19, 223:9, 226:7, 226:12, 226:25, 265:23, 268:14, 268:19, 302:4, 315:18, 316:15, 326:3 sit 291:22 sitting 195:23, 218:17, 219:6, 270:17,	270:19, 287:14 situation 200:8, 204:14, 204:15, 304:13, 311:7, 312:5, 317:19 situations 198:24 six 232:8 size 226:18 sizes 322:2 skill 205:4, 325:21, 325:25 skilled 197:2, 204:22, 205:10, 208:14, 260:25, 263:3, 268:10, 273:16, 277:3, 282:9, 295:16, 306:9, 306:16, 307:24, 308:9, 308:24, 309:4, 310:4 slots 316:25 small 226:18, 232:5, 234:7 smaller 182:18 smart 219:6 so-called 179:22, 185:11, 203:23, 204:10, 205:8, 205:13, 205:17, 234:5, 314:15 society 323:25 sold 248:24 soldier 179:24, 180:13,	182:1, 182:3, 182:6 solid 233:7 some 180:4, 180:8, 182:20, 183:1, 183:12, 185:7, 193:19, 194:20, 198:15, 202:16, 203:2, 203:3, 203:6, 206:23, 225:10, 227:3, 230:22, 236:3, 236:13, 245:21, 248:13, 248:15, 248:16, 250:24, 252:24, 257:10, 257:11, 257:17, 263:2, 264:13, 265:3, 265:20, 269:17, 278:23, 278:24, 291:19, 295:3, 298:24, 316:16, 324:12, 324:15 somebody 203:4, 248:14, 277:8 somehow 291:18, 304:6 someone 324:7 something 195:11, 197:5, 198:13, 199:3, 199:5, 200:9, 208:3, 235:3, 241:10, 243:18, 246:12, 249:6, 250:22, 263:8, 266:6, 269:2, 272:17, 281:6, 290:2, 298:11, 304:5, 307:21, 316:21, 319:3 sometimes 225:6, 309:5,
--	--	--	--

313:1, 313:4, 316:17 somewhat 260:11 somewhere 182:14, 183:13, 209:3, 209:16, 261:4 soon 236:3, 304:4, 310:11 sorry 181:19, 190:14, 195:5, 196:22, 200:25, 219:12, 221:8, 222:5, 222:16, 240:18, 246:2, 256:9, 256:17, 261:13, 270:8, 272:2, 278:15, 295:11, 298:3, 298:10, 307:4, 316:13, 320:12, 321:2 sort 182:5, 182:16, 182:20, 231:7, 235:10, 235:15, 235:18, 240:6, 244:25, 246:12, 247:20, 262:20, 271:5, 274:20, 295:17, 301:23, 302:9, 306:16, 309:9 sorted 225:1, 227:11, 229:16, 230:10 sorting 229:20 sorts 227:22, 230:4, 231:4, 308:13, 313:22 source 229:18, 234:9, 263:13, 283:14, 283:25, 290:24	sources 262:1, 320:25 southern 320:18 space 232:24, 233:6, 233:11, 233:12, 233:15, 233:24, 234:5, 234:6, 240:4, 300:12, 300:13 spaces 289:8 special 182:5, 321:5, 321:20 specialized 219:18, 219:25, 220:11, 220:13 specific 191:19, 204:23, 244:13, 261:12, 266:13, 283:23, 284:11, 293:18 specifically 199:12, 296:15 specification 201:18, 216:20, 219:25, 243:13, 296:2 specifics 266:7, 266:8 specs 190:17, 197:4, 207:2, 208:11 speculate 266:6 speed 297:1 spite 277:25 spoke 217:3 spot 222:17 ss 328:2, 329:3 st 180:14	staff 322:6 stage 229:11, 232:19, 237:16, 237:17, 238:18, 239:22, 240:11 stages 240:3 stamping 222:13 stand 258:13 standard 177:11, 262:23, 326:12 start 181:15, 236:6, 236:12, 236:19, 236:20, 237:9, 258:13, 273:3, 273:4, 289:9, 293:5 started 178:9 starting 177:16, 188:5, 189:6, 193:18, 223:18, 225:25, 226:8, 244:22, 250:10, 258:5, 287:20, 292:19, 292:20 starts 251:25, 252:5 state 177:16, 223:10, 254:24, 262:24, 262:25, 267:14, 275:12, 291:14, 302:8, 328:1, 328:8, 329:2, 329:8 stated 178:12, 191:13, 192:14, 254:22 statement 191:23, 194:10,	209:11, 215:13, 215:19, 229:19, 230:18, 231:9, 254:13, 255:5, 262:22, 269:10, 269:14, 284:13, 285:5, 289:22, 299:20, 303:4 states 173:1, 174:1, 176:25, 177:8, 193:23, 207:4, 210:13, 227:9, 227:21, 232:22, 319:1, 324:19, 324:23 statically 282:13 statistically 278:25 status 242:14, 243:22, 243:23, 247:7, 248:5, 248:21, 248:22, 249:6, 249:20, 249:21, 249:22, 249:23, 251:8, 251:17, 253:12, 255:9, 255:10, 257:9, 257:16, 279:1, 315:24, 315:25, 316:10, 317:3, 317:4, 317:6, 317:21, 319:7, 319:8 statuses 316:20 stay 203:12, 254:21 stayed 203:21 stays 203:2 stenographer 178:2 stenographer's 329:1, 330:10
--	--	--	--

stenographically 329:23 step 226:11, 227:9, 228:4, 234:14 stephanie 175:27 stick 199:11 still 179:4, 182:18, 182:25, 198:18, 207:13, 209:13, 215:18, 217:6, 221:21, 251:9, 254:3, 256:9, 265:11, 289:4, 298:21, 311:9, 312:8, 312:15, 323:19 stop 247:8, 253:6 storage 194:17, 194:24, 195:18 store 194:2, 194:3, 213:6, 213:7, 213:10, 214:9, 214:10, 277:5, 291:7, 292:2 stored 206:21, 241:18, 253:11, 255:3, 255:7, 255:13, 255:20, 255:23, 274:12, 274:23, 278:5, 285:14, 285:16, 286:20, 286:24, 289:7, 290:10 stores 207:4, 212:19, 212:20, 270:11, 317:13 storing 194:4, 194:11, 206:16, 212:25,	286:1, 286:2, 290:24, 315:9 story 182:12 straightforward 313:1 strange 221:9 strategy 309:17 stream 256:15 streams 244:5 street 175:6 stripe 292:19, 293:1 strongly 193:25 structure 258:16 student 181:5 students 183:11, 321:9, 322:4, 322:5, 322:12 stuff 184:7, 185:1 stuttard's 235:21, 258:10, 259:16, 261:9, 264:25, 269:16, 271:10, 275:19, 315:8 stuttard-based 267:1 stutzman 175:27 subject 239:25, 295:3, 321:1, 324:16 subsequence 246:7, 246:8 subsequently 211:19 subset 282:18	substance 312:1 substantial 182:8 subsystem 283:21, 283:24, 287:24 subsystems 183:19, 183:24, 186:3 success 182:12 suddenly 247:16 suffice 194:18 sufficient 230:15, 253:5 suggesting 282:2 suite 175:18 super 236:17 support 176:11, 289:22 supports 193:25 supposedly 199:25, 200:8, 203:17 sure 187:12, 196:12, 196:16, 204:17, 227:19, 228:24, 229:14, 234:22, 235:13, 241:7, 241:9, 268:13, 269:8, 291:17, 314:22, 320:7, 320:14 surface 261:19, 262:6, 263:20, 263:22, 264:7 surfaces 234:10, 262:1, 264:4	sutcliffe 175:4, 177:21 switched 183:3, 279:19 switching 321:15 sworn 178:1, 178:3 symposium 323:18, 324:3 synthesis 321:17 system-design 238:4, 238:6 system-type 322:21 systems 179:17, 180:19, 181:3, 181:16, 181:18, 181:19, 181:22, 182:24, 184:2, 184:17, 185:20, 185:21, 185:24, 185:25, 186:3, 238:13, 249:3, 266:2, 272:13, 294:11, 306:19, 306:20, 309:8, 310:19, 321:23, 323:21, 323:22, 323:24, 324:4 T table 192:23 take 188:4, 198:6, 220:18, 222:16, 233:20, 234:4, 260:7, 271:5, 276:7, 297:20, 301:16, 315:13 taken 174:21 takes 197:16, 270:21, 295:6
--	--	--	---

taking 177:17, 197:6, 198:20, 199:8, 300:8, 309:6 talk 179:14, 264:4, 275:15, 293:16, 316:9 talked 241:9, 302:25, 309:24, 313:9, 315:14 talking 199:23, 201:16, 201:19, 201:25, 209:3, 209:5, 217:9, 217:10, 226:15, 227:12, 237:25, 238:1, 264:8, 264:9, 292:12, 296:14, 296:15, 301:25, 306:23, 307:3, 307:13, 311:16, 311:18 targets 184:6 task 230:21, 232:18, 247:13, 247:14, 248:8, 248:11, 248:12, 248:22, 248:25, 255:12, 256:24, 256:25, 257:1, 257:7, 257:8, 258:19, 258:22, 258:23, 262:21, 263:9, 263:10, 264:21, 265:19, 266:9 tasks 186:23, 186:24, 199:24, 202:15, 202:18, 232:10, 237:14, 237:24, 239:5, 242:23, 248:18, 258:19, 261:10, 262:2,	264:12, 264:13, 264:14, 265:21, 266:1, 309:3, 323:10 taught 276:4, 278:7, 304:25, 321:4, 321:20 teach 320:23, 321:1, 321:3, 321:15 teaches 250:5, 277:4 team 180:4, 182:12, 183:10, 185:22, 186:21 technical 322:15, 323:17, 324:5 technician 175:26, 187:5, 221:2, 221:7, 221:12, 221:23, 222:1, 222:5, 222:9, 222:15, 222:20, 325:14 techniques 180:22, 186:14 technologies 173:11, 174:11, 175:13, 177:7, 327:4 technology 186:10, 322:24, 323:13 tell 178:6, 192:3, 202:1, 240:20, 316:18 tells 255:15, 300:10, 310:12, 316:13 ten 183:11 ten-minute 220:19, 297:20 tend 195:9	tends 309:15 term 189:8, 189:25, 190:23, 192:14, 196:9, 201:1, 204:21, 205:2, 205:5, 209:1, 217:11, 236:18, 261:22, 261:24, 262:4, 262:9, 293:23 terminals 205:18 terms 176:16, 181:3, 188:9, 189:7, 191:12, 192:9, 224:2, 234:10, 255:11, 273:24, 279:19, 290:17, 309:2 testified 178:7, 324:18 testify 179:7 testimonies 195:15 testimony 179:12, 195:16, 196:2, 274:4, 287:10, 329:23, 329:25 texas 175:8 texture 261:10, 261:15, 261:17, 261:22, 262:6, 262:7, 262:9, 262:19, 262:23, 263:10, 264:5, 264:13, 264:14, 264:22, 265:1, 265:14, 265:19, 266:4, 287:23 texture-related 281:16	textures 234:11, 262:11, 262:12, 262:17, 264:11, 265:22 th 175:7 thank 177:23, 222:22, 250:16, 288:12, 302:5, 326:6, 326:8 themselves 177:16 theory 321:15 therefore 203:9, 207:25, 208:14, 232:6, 243:10, 246:6, 248:20, 251:13, 253:14, 259:14, 291:5, 306:14 thing 183:5, 183:17, 197:7, 250:15, 263:17, 284:17, 300:17, 311:15 things 183:22, 184:15, 199:7, 224:18, 238:1, 257:11, 263:2, 275:8, 283:9, 291:22, 294:18, 296:15, 300:15, 309:6, 310:14, 315:7, 316:11, 320:14, 323:5, 323:16 think 180:10, 181:14, 182:19, 182:23, 183:1, 190:20, 194:25, 207:15, 207:25, 208:18, 208:19, 208:20, 209:19, 210:20, 210:22, 214:14, 217:9, 217:11,
---	---	--	--

222:3, 237:25, 239:12, 240:22, 248:4, 249:10, 249:19, 251:23, 254:12, 259:8, 263:6, 264:8, 264:16, 266:2, 266:5, 266:12, 271:17, 272:9, 272:19, 277:23, 278:2, 279:18, 280:1, 280:14, 280:25, 288:20, 289:22, 289:23, 290:14, 295:8, 296:14, 301:10, 303:20, 306:5, 310:1, 311:14, 311:15, 312:2, 319:22, 322:3, 322:17, 324:12, 324:17 thinking 228:9, 256:23 third 193:24, 242:5, 275:7, 330:3, 330:14, 330:18 thought 200:7, 206:17, 207:1 three 183:15, 185:6, 244:10, 300:10, 305:20, 305:23, 321:10, 325:6 three-page 317:16 through 182:1, 186:14, 186:18, 186:19, 199:3, 199:7, 202:25, 203:12, 225:24, 229:1, 243:21, 245:5, 246:23, 273:11, 273:13, 276:16, 277:13, 283:19,	285:11, 291:3, 291:10, 299:3, 314:17, 315:23, 316:16, 317:16 throughput 181:5, 185:18, 249:1, 272:14, 272:15, 315:4 throughput-inten- sive 310:17 tick 203:13 time 177:10, 177:11, 180:9, 181:1, 181:8, 182:11, 183:3, 183:22, 184:3, 185:11, 186:16, 194:5, 194:12, 194:16, 198:15, 200:22, 202:10, 203:3, 203:13, 203:24, 207:1, 207:12, 211:9, 213:17, 213:19, 213:24, 214:12, 220:21, 220:24, 227:17, 230:22, 232:19, 234:19, 234:23, 237:3, 237:24, 238:13, 238:22, 244:18, 250:1, 250:6, 251:2, 257:10, 257:22, 257:25, 260:16, 268:14, 270:19, 273:17, 277:7, 279:6, 279:12, 280:8, 280:12, 280:18, 281:5, 281:6, 281:11, 281:21, 282:19, 285:16, 290:6, 297:23, 298:1, 300:4, 303:11, 303:14, 304:8,	306:9, 306:18, 307:24, 309:6, 310:9, 310:21, 317:1, 320:11, 322:4, 324:14, 326:2, 326:11, 330:6 times 183:15, 322:3 timing 244:20, 245:1 title 320:19 today 177:12, 177:24, 179:7, 179:12, 193:7, 195:23, 222:11, 222:14, 317:17, 319:14, 324:17, 324:25 today's 177:9 together 180:12, 244:15, 253:5, 280:21, 292:6, 294:6, 294:13, 294:14, 321:7 told 310:15 top 181:25, 242:4, 242:5, 246:20, 254:22, 270:17, 270:23, 307:20 topic 321:5, 321:20 topics 321:11 tops 181:25 touched 301:9 towards 189:14 track 317:12, 323:15, 324:7	tracking 186:18 trade-off 295:19, 296:5, 296:8 trademark 173:1, 174:1, 177:8 transcript 178:22, 329:24 transfer 209:15, 224:24, 225:21, 281:16, 283:19 transferred 202:20, 226:5, 283:22 transform 232:23, 233:11 transformation 233:5, 233:23, 300:7, 300:10 transformations 233:8, 300:5, 300:22 transformers 186:19 translate 234:6 transmission 277:11 transmits 290:1 transmitted 286:15 transmitting 289:18 transparent 183:5 treat 271:15 trial 173:2, 174:2, 195:16 tried 267:10 tries 278:16
--	--	---	--

trigger 252:15 triggers 252:11 true 212:16, 229:13, 231:12, 231:23, 236:1, 236:7, 276:19, 301:3, 303:7, 328:8, 329:25 truth 178:7 truthful 179:12 try 180:22, 202:6, 221:20, 242:1, 252:1, 263:7, 278:15, 294:8, 314:12 trying 219:13, 230:6, 248:15, 262:5, 272:4, 276:1, 277:18, 291:18, 292:15, 293:4, 316:6 turn 241:2 turned 320:13 turning 212:9 twice 207:8 two 183:15, 189:18, 189:19, 190:19, 191:6, 191:7, 191:16, 191:25, 201:7, 201:15, 213:5, 214:11, 215:7, 218:18, 219:18, 223:14, 224:21, 230:14, 232:6, 232:7, 237:6, 237:14,	237:25, 241:20, 252:15, 258:7, 271:1, 273:8, 273:10, 279:7, 284:17, 285:19, 285:25, 287:5, 288:6, 288:9, 291:18, 292:1, 292:4, 292:5, 293:8, 295:17, 295:19, 295:23, 304:8, 305:17, 309:12, 310:7, 311:16, 316:11, 316:24, 322:5, 324:12 two-dimensional 233:21, 240:5 two-star 180:9 type 183:5, 189:21, 190:19, 191:7, 192:1, 211:8, 213:19, 217:14, 218:10, 219:7, 245:21, 256:7, 273:12, 273:25, 274:2, 280:11, 280:17, 283:15, 284:3, 284:6, 284:12, 285:1, 285:9, 285:20, 285:22, 286:8, 286:12, 286:22, 287:6, 287:11, 287:16, 287:25, 288:3, 288:15, 288:23, 290:7, 290:10, 293:7, 294:2, 294:3, 294:5, 302:21, 303:18, 303:24, 304:20, 309:9, 313:13, 318:11, 319:16, 319:19, 319:25, 320:2 type-specific 278:11, 281:24	types 182:23, 185:17, 213:10, 213:11, 230:23, 234:2, 274:6, 277:5, 277:11, 278:6, 279:11, 280:3, 282:4, 282:6, 287:8, 287:21, 288:7, 292:9, 293:25, 294:22, 305:16, 305:17, 313:22, 323:3, 323:21 typically 240:12, 264:5, 294:6, 295:21, 300:7, 310:16, 310:19, 322:10, 325:2 U u 173:9, 174:9 ulc 173:11, 174:11, 175:13, 177:7, 327:4 ultimate 235:19 unapplied 221:3 unaware 289:15 unclear 245:1 under 178:6, 182:3, 191:7, 204:1, 252:4, 252:15, 308:6, 328:7, 329:19 undergraduates 321:9 understand 179:6, 188:9, 192:8, 193:8, 195:7, 196:22,	197:2, 202:14, 204:10, 204:13, 205:11, 228:10, 230:19, 231:19, 239:14, 245:23, 251:1, 257:2, 261:1, 262:18, 263:4, 264:20, 264:25, 265:9, 265:14, 268:10, 272:6, 285:25, 286:10, 287:4, 290:14, 311:14 understanding 201:22, 205:4, 207:18, 217:19, 229:24, 230:5, 294:4 understood 208:13, 268:13, 295:15 unified 184:22 unique 261:25 unit 223:12, 227:22, 229:18, 230:4, 231:3, 235:9, 235:14, 253:21, 254:3, 254:4, 254:6, 254:10, 268:5, 271:2, 273:12, 273:14, 274:11, 277:5, 279:18, 290:4, 291:5, 299:5, 300:2, 300:20, 301:1, 306:4, 311:12, 319:8 united 173:1, 174:1, 176:25, 177:7, 324:19, 324:23 units 276:3, 286:15, 291:5, 300:16 universe 265:25
---	---	--	---

universities 322:11 university 320:18, 322:6 unknown 236:1 unless 187:9, 253:19 unmarked 187:21 unnecessary 209:17, 325:16 until 236:1, 239:20 update 237:12 updated 237:5, 237:8 updates 237:11 upload 221:9 usc 185:8 use 200:14, 201:20, 201:24, 205:1, 231:16, 233:23, 237:20, 247:9, 247:22, 255:17, 255:19, 257:12, 271:25, 276:24, 278:4, 278:10, 282:5, 288:4, 290:9, 291:7, 304:16, 305:11, 307:11, 308:3, 316:4 useful 229:20, 276:5, 276:17 user 233:22, 234:1, 235:1, 300:12 uses 235:10, 235:15, 250:11, 250:20 using 186:16, 200:11,	223:23, 277:10, 290:18, 293:24, 312:17, 314:15 usually 258:19, 262:1, 272:18 V vague 196:12 valid 294:24 validity 210:4 valuable 324:25 valuations 261:18 various 180:18, 181:2, 181:15, 188:23, 276:10 vector 300:8, 300:10, 300:14 verbal 316:14 versa 211:20, 219:17, 295:21 version 178:13, 221:10 versus 217:24, 242:14, 249:23, 254:15, 254:20, 255:10, 255:12, 257:17, 259:9, 265:13, 271:17, 317:9, 317:11, 317:22, 318:18, 318:23, 319:6, 320:6 vertexes 236:1, 264:6 vertices 239:21 verticis 226:2, 264:3,	264:6 via 174:22 vice 211:20, 219:16, 295:21 video 177:10, 177:14, 220:21, 220:24, 257:22, 257:25, 298:1 videoconferencing 174:22 videographer 175:25, 177:4, 177:12, 177:23, 220:20, 220:23, 257:21, 257:24, 297:22, 297:25, 326:9 videotaped 177:5 view 190:16, 191:2, 191:24, 195:17, 201:13, 204:22, 207:2, 208:1, 208:12, 212:11, 213:9, 216:13, 216:20, 239:8, 264:20, 268:11, 274:15, 275:18, 277:22, 297:2, 298:16, 306:17 viewing 182:3, 234:9, 235:2, 235:6, 261:21 violation 176:19 vision 186:19 visit 183:12, 183:14 visiting 181:5 visually 233:21	vlsi 321:6 voice 177:15 volume 173:21, 174:20 W wait 238:2, 248:20, 257:5, 257:10, 309:10, 309:12, 309:13, 314:24 waiting 253:19, 253:20, 253:21, 254:2, 254:4, 254:5, 298:10, 304:2, 315:2 walk 225:24 want 178:10, 179:14, 184:18, 200:6, 200:15, 201:21, 201:25, 202:3, 211:24, 217:25, 218:2, 223:16, 231:6, 231:25, 239:14, 246:14, 250:15, 253:4, 274:16, 278:20, 279:9, 280:20, 281:5, 296:1, 298:5, 299:10, 299:16, 301:11, 301:17, 313:8, 313:18, 315:12, 325:15 wanted 222:10, 265:15 wants 215:17, 260:9, 297:13, 297:15 warrior 179:22, 179:23, 186:6 waste 281:6
---	--	--	---

way 187:8, 193:24, 198:5, 199:1, 200:16, 204:9, 204:12, 207:16, 208:6, 209:17, 211:9, 220:4, 228:21, 231:25, 253:8, 260:6, 262:16, 262:23, 263:2, 270:7, 272:1, 284:21, 290:1, 292:25, 300:1, 302:20, 303:7, 303:16, 310:1, 312:23, 314:6, 314:20, 318:20, 318:21, 321:18 ways 249:23, 269:17, 287:5, 296:13, 303:22, 310:7 we'll 251:10, 299:16, 305:1 we're 220:20, 237:25, 240:13, 257:21, 258:4, 264:8, 296:14, 311:16, 326:10 welcome 281:3 well-known 308:4 well-posed 202:13 went 192:21, 246:5, 317:16 whatever 200:6, 201:14, 217:5, 259:1, 301:7 whenever 252:10, 307:25 wherein 256:14	whether 191:5, 207:19, 238:1, 243:14, 246:4, 246:6, 256:19, 280:11, 286:22, 291:6, 299:25, 312:21, 319:4, 319:9, 320:5 whichever 218:10, 218:12 whitaker 247:22 white 315:24 whittaker 187:16, 301:14, 302:18, 304:25, 305:1, 307:11 whoever 313:3 whole 250:15 wide 306:7 william 269:15, 270:13 williams 176:30, 187:16, 247:21, 266:18, 267:3, 267:7, 268:3, 268:5, 268:12, 269:25, 270:4, 270:8, 270:15, 270:21, 271:7, 271:19, 272:5, 272:23, 275:20, 275:24, 276:4, 276:19, 276:23, 276:25, 277:9, 277:17, 278:3, 278:5, 278:9, 278:17, 279:10, 279:24, 280:10, 280:16, 281:22, 282:2, 282:5, 282:10, 283:7, 283:13,	284:9, 284:23, 285:6, 285:10, 285:17, 285:20, 285:23, 285:24, 286:4, 286:7, 286:21, 287:3, 287:7, 287:15, 287:19, 287:20, 288:6, 288:14, 288:19, 289:17, 289:21, 298:13, 299:4, 299:20, 299:21, 299:25, 300:25, 301:14, 302:7, 303:13, 305:2, 305:4, 305:9, 305:22, 306:10, 306:14, 306:17, 307:19, 308:9, 308:17, 309:1, 310:2, 312:3, 313:6, 313:11, 314:7 withdrawn 186:1, 210:21, 241:4, 253:9, 262:14, 284:8, 286:5, 289:16, 307:15, 313:22 within 225:1, 226:14, 237:15, 276:23, 282:23, 296:20, 313:25 without 197:5, 199:16, 203:11, 221:18, 236:22, 264:14, 294:20, 301:4, 314:15, 330:4 witness 176:3, 177:22, 178:1, 181:11, 187:6, 269:1, 269:4, 324:2, 326:8, 330:12, 330:16 word 200:14, 201:19,	201:20, 201:22, 201:24, 202:1, 202:6, 211:22, 323:8 work 181:17, 238:5, 239:17, 306:5, 320:17, 322:6, 322:7, 324:13 worked 180:17, 181:2, 181:7, 181:15, 181:22 working 186:22, 242:11, 245:6, 296:13 works 185:13, 249:15, 302:20 wouldn't 209:16 write 271:25, 292:13, 292:18, 292:19, 293:18, 293:20, 293:25, 294:1, 294:13, 294:15, 294:22, 296:17, 296:19, 296:25, 297:13, 297:15, 313:16 writing 192:4, 295:7, 296:16, 297:4, 297:6 written 220:4, 233:25, 236:23, 237:20, 285:12, 323:5 wrote 180:24, 292:23 X xyz 246:1 Y yeah 183:23, 186:5,
--	---	---	--

192:18, 222:12, 222:15, 225:7, 229:14, 233:14, 243:16, 250:16, 251:1, 252:2, 257:18, 261:13, 264:2, 267:19, 280:13, 284:21, 298:12, 299:15, 303:16, 307:9, 312:19, 316:5, 322:16, 322:22	zoom 174:22	323:14, 326:2	1036 249:21
	\$	0564 179:7	104 250:2
	\$16 325:5	06 180:25	105 209:2
	\$6 325:3	07 180:25, 298:1	106 223:11, 224:3, 226:15, 270:15
	.	074 319:10	107 266:15, 267:8, 267:13, 268:18, 268:20, 269:10
	.14125 330:25	08 181:1	108 250:3, 250:14, 270:17
	0	1	11 242:9, 259:16, 324:11, 330:20
year 181:6, 183:15, 322:13	00564 173:7, 174:7, 177:9, 327:6	1 266:22, 267:8, 268:1, 268:17, 297:23	116 270:23, 270:25, 271:2, 278:11, 279:10, 279:16, 280:17, 280:22, 281:15, 283:19, 302:23
years 179:19, 185:6, 185:23, 321:3, 322:9	02 326:11	10 220:21, 220:24, 243:25	12 242:15, 242:16, 250:4, 257:22, 257:25, 322:4, 322:12, 324:11
yes-or-no 192:16, 192:19, 192:20	03 174:24, 326:11, 326:13	100 322:13	13 250:4, 287:20
yield 251:11	04 174:24, 177:2, 177:10, 180:10	1005 221:11	1318 188:22, 193:15
yielding 242:14, 247:7, 248:5, 248:6, 248:12, 248:13, 248:21, 248:23, 249:5, 249:23, 251:8, 251:11, 251:17, 251:19, 252:14, 252:23, 253:12, 254:19, 254:20, 254:24, 255:9, 255:25, 256:7, 256:20, 256:25, 257:3, 257:9, 257:15, 260:24, 316:11, 317:6, 317:12, 317:22, 318:9, 319:7	05 257:22	102 223:12, 249:20, 250:2, 283:21, 283:24	14 253:17
	053 187:16, 188:2, 188:11, 189:8, 191:4, 191:8, 194:24, 195:2, 195:20, 196:6, 197:4, 199:3, 199:21, 199:22, 200:2, 201:17, 204:23, 205:1, 205:11, 205:22, 205:25, 206:6, 208:11, 212:3, 213:16, 215:19, 217:8, 219:24, 220:15, 223:7, 238:14, 238:23, 258:6, 259:25, 298:8, 299:8, 302:8, 307:7, 321:2, 322:24,	1024 223:12, 241:17, 242:21, 243:5, 243:25, 253:11, 258:11, 259:14, 317:21	14125 173:31, 174:27, 329:9
		1025 240:22, 241:1, 249:11, 249:15, 249:24, 317:4, 317:14, 319:18	148 299:23, 300:3
yields 248:7		1026 240:23, 241:24, 242:6, 244:1, 248:3, 249:15, 250:11	15 189:6, 227:8, 322:7
yourself 272:22		1030 249:20, 317:3	1596 175:20
		1032 250:1	
z			
zero 253:20			

16 189:14, 191:14, 192:15, 322:3 178 176:5 18 176:17, 227:22, 242:16 187 176:10 188 176:15 19 283:13 193 176:18 1996 323:19 1b 267:15, 301:25, 302:8, 302:15 2 2 267:12, 298:1 20 223:17, 223:18, 224:2, 224:9, 225:23, 225:25, 227:22, 228:4, 229:1, 310:23, 311:3 2000 179:20 2003 180:10 2005 180:25 2006 182:11 2010 183:2 2018 186:15 2019 186:15 202 278:7, 283:14	2022 176:17 2023 173:7, 174:7, 176:24, 177:9, 179:7, 327:6 2024 173:23, 174:25, 177:2, 177:9, 327:10, 328:11, 330:20 2025.320 329:15, 329:20, 330:8, 330:19 2025.540 329:21, 329:26 204 270:21, 283:20, 287:21, 287:25, 288:2 206 176:25 2093 329:18 21 180:14, 223:18, 224:2, 224:9, 227:8, 229:2, 229:11, 230:24, 232:20, 234:15 22 187:2, 225:13, 250:4 222 176:28 23 206:3, 224:2, 250:10, 251:25, 252:4, 259:16, 287:18, 319:13 24 243:25, 253:17, 318:5 25 221:1, 221:6, 221:7, 221:8, 225:25, 229:1, 242:16, 250:10,	255:24, 318:5, 319:13 26 220:21, 242:17, 250:14, 269:21 269 176:30 27 242:9 28 256:14 29 188:17, 256:14 2d 245:20, 282:21, 320:4 3 3 174:24, 326:11, 326:13 30 226:6, 256:14 300 175:18 31 176:15, 256:12 314 175:20 32 193:19, 225:13, 226:8, 242:17, 250:10 320 176:6 33 193:12, 193:14, 242:4, 242:9 337 176:19 35 242:16, 242:17, 250:10 3580 175:17 37 193:18, 193:23, 257:25	38 246:19 39 246:20 396 292:21 3d 245:20, 282:22, 282:23, 320:4 3j6ptabdoCKET@or- rick 175:10 4 40 175:7 400 292:21 402 271:2, 279:18 404 271:4, 274:11 41 325:11, 325:18, 325:20, 326:1 42 188:4, 188:5, 188:8 43 220:24 45 258:5 5 519722 173:32 55 297:23 6 609 175:6 62 223:5 6400 175:9 65 256:12
---	--	--	---

658 175:9 68 240:15 69 249:9 7 7,742,053 173:10, 174:10, 176:14, 176:26, 327:8 7,752,053 179:8 713 175:9 73 258:5 77002 175:8 8 800 322:16 8016 329:11 84 261:13 858 175:20 9 9 174:24, 177:2, 177:10 92130 175:19 § §§ 329:20	
---	--