United States District Court,
S.D. California.

HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P,
Plaintiff.
v.
GATEWAY, INC,
Defendant.
Gateway, Inc,
Counterclaim-Plaintiff.
v.
Hewlett-Packard Development Company L.P. Hewlett-Packard Company and Compaq Information Technologies Group, L.P,
Counterclaim-Defendants.
Intel Corp,
Intervenor.

Civil No. 04CV0613-B(LSP)

Dec. 12, 2005.

John Allcock, DLA Piper US, San Diego, CA, for Plaintiff/Counterclaim-Defendants.

Darryl J. Adams, Dean M. Munyon, James D. Smith, Wayne Harding, Dewey Ballantine, W. Bryan Farney, Dechert LLP, Austin, TX, Jonathan D. Baker, Dechert LLP, Mountain View, CA, for Defendant.

CLAIM CONSTRUCTION ORDER FOR UNITED STATES PATENT NUMBER 5,596,759

RUDI M. BREWSTER, **Senior District Judge.**

Pursuant to Markman v. Westview Instruments, Inc., 517 U.S. 370, 116 S.Ct. 1384, 134 L.Ed.2d 577 (1996), on November 1-3, 2005, the Court conducted a Markman hearing in the above-titled patent infringement action regarding construction of the disputed claim terms for U.S. Patent Number 5,596,759 ("the '759 patent"). Plaintiff Hewlett-Packard Development Company, L.P. ("HP") was represented by the law firm of DLA Piper Rudnick Gray Cary U.S. LLP, Defendant Gateway, Inc. ("Gateway") was represented by the law firm of Dewey Ballantine LLP, and Intervenor Intel Corporation ("Intel") was represented by the law firm of Weil, Gotshal & Manges LLP.

At the Markman hearing, the Court, with the assistance of the parties, analyzed the claim terms in order to prepare jury instructions interpreting the pertinent claims at issue in the '759 patent. Additionally, the Court prepared a case glossary for terms found in the claims and the specification for the '759 patent considered to be technical in nature which a jury of laypersons might not understand clearly without specific definition.

After careful consideration of the parties' arguments and the applicable statutes and case law, the Court **HEREBY CONSTRUES** the claims in dispute in the '759 patent and **ISSUES** the relevant jury instructions as written in Exhibit A, attached hereto. Further, the Court **HEREBY DEFINES** all pertinent technical terms as written in Exhibit B, attached hereto.

IT IS SO ORDERED.

EXHIBIT A FN1

UNITED STATES PATENT NUMBER 5,596,759-CLAIM CHART

VERBATIM CLAIM LANGUAGE	***COURTS CONSTRUCTION***
Claim 1	
1. A method of initializing a multi-processor computer system, the multi-processor computer system including at least two processors, one of which is considered a primary processor during initialization, said processors being powered up together; common peripherals; and a common storage element which stores initialization code used during start up of the computer system, the common storage element being common to each processor, the initialization code having processor and common peripheral portions, the method comprising the steps of:	1. A method of initializing a multi-processor computer system, the multi-processor computer system including at least two ***processors,*** [*circuitry that has the ability to fetch, decode, and execute instructions and to transfer information to and from other resources over the computer's main data transfer path, the bus*] one of which is considered a ***primary processor*** [*the processor that performs a complete* ***initialization,*** *among other things*] during ***initialization*** [*the process by which the computer system is made ready for use*], ***said processors being powered up together*** [*power is supplied to both* ***processors*** *at the same time*]; ***common peripherals*** [*hardware devices, other than the computer (processor, memory, and data paths), such as a hard disk drive or printer, that can be accessed by both* ***processors***]; and a ***common storage element*** [*a device that stores data and that can be accessed by both* ***processors***] which stores ***initialization code*** [*computer code used to start up the computer system]* used during start up of the computer system, the ***common storage element*** being common to each processor, ***the initialization code having processor and common peripheral portions*** [*the* ***initialization code*** *has a section that is capable of initializing* ***processors*** *and a section that is capable of initializing* ***common peripherals***], the method comprising the steps of:
(a) each processor executing initialization code stored in the common storage element;	(a) ***each processor executing initialization code stored in the common storage element*** [*each processor executes* ***initialization code*** *stored in the designated device that stores data and that can be accessed by both* ***processors***];
(b) each processor performing processor initialization code to initialize itself; and	(b) each processor ***performing*** [*executing*] processor ***initialization code*** to initialize itself; and
(c) each processor determining if it is the primary processor and performing common peripheral initialization code	(c) each processor determining if it is the ***primary processor*** and ***performing*** common peripheral ***initialization code*** only if it is said ***primary processor .***

only if it is said primary processor.	
Claim 6	
6. The method of claim 1, further comprising the steps of:	6. The method of claim 1, further comprising the steps of:
(h) applying and releasing a system reset to the multiprocessor computer system before steps (a)-(c), the system reset causing all but one of said processors to be restrained; and	(h) applying and releasing a ***system reset*** [*a signal that causes the system to return to the state it would be in if the power were turned off and then on again*] to the multiprocessor computer system before steps (a)-(c), the *system reset* causing all but one of said ***processors*** to be ***restrained*** [*held in an inactive state*]; and
(i) said primary processor causing each other processor to be released.	(i) said ***primary processor*** causing each other ***processor*** to be released.
Claim 7	
7. The method of claim 6, wherein the multi-processor computer system further includes an active processor identifying value, the method further including the steps of:	7. The method of claim 6, wherein the multi processor computer system further includes an ***active processor identifying value*** [*the value that indicates which one of the* ***processors,*** *if any, is currently active*], the method further including the steps of:
(j) each processor acquiring the active processor identifying value; and	(j) each processor acquiring the ***active processor identifying value;*** and
(k) each processor selecting a portion of said initialization code to execute based on the active processor identifying value,	(k) each processor selecting a portion of said ***initialization code*** to execute based on the ***active processor identifying value,***
wherein each processor is identified as a primary or secondary processor, and	wherein each processor is identified as a primary or secondary processor, and
wherein each processor identified as a primary processor initializes the common peripherals and each processor identified as a secondary processor does not initialize the common peripherals.	wherein each processor identified as a ***primary processor*** initializes the ***common peripherals*** and each processor identified as a secondary processor does not initialize the ***common peripherals.***
Claim 8	
8. The method of claim 6, wherein the multi-processor computer system further includes a main memory for storing redirection vectors and wherein step (i) further comprises the steps of:	8. The method of claim 6, wherein the multiprocessor computer system further includes a main memory for storing ***redirection vectors*** [*an address obtained from a memory location that directs the processor to a new memory location containing code to be executed*] and wherein step (i) further comprises the steps of:
(1) said primary processor writing an address into a redirection vector location of main memory, said address pointing to a starting address of a portion of said initialization code not causing the common peripherals to be initialized; and	(1) said ***primary processor*** writing an address into a ***redirection vector location of main memory*** [*a location in main memory where a redirection vector is stored*], said address pointing to a starting address of a portion of said ***initialization code*** not causing the ***common peripherals*** to be initialized; and
(m) said primary processor causing each other processor to be released after setting the redirection vector.	(m) said ***primary processor*** causing each other processor to be released after ***setting the redirection vector*** [*storing the address of code to be executed in the redirection vector*

	location].
Claim 11	
11. A multiprocessor computer system, comprising:	11. A multiprocessor computer system, comprising:
at least two processors, one of which is considered a primary processor during initialization, said processors being powered up together;	at least two ***processors,*** one of which is ***considered a primary processor during initialization, said processors being powered up together;***
a common storage element containing processor executable initialization code used during start-up of the computer system, the common storage element being common to each processor, said initialization code having processor and common peripheral initialization code;	a ***common storage element*** containing processor executable ***initialization code*** used during start-up of the computer system, the ***common storage element*** being common to each processor, said ***initialization code having processor and common peripheral initialization code*** [*the* ***initialization code*** *has a section that is capable of initializing* ***processors*** *and a section that is capable of initializing* ***common peripherals***];
a common peripheral including a hard disk;	a common peripheral including a hard disk;
wherein said initialization code when executed by said processors causes said processors to perform the steps of:	wherein said ***initialization code*** when executed by said ***processors*** causes said *processors* to perform the steps of:
(a) each processor executing said initialization code stored in the common storage element;	(a) ***each processor executing said initialization code stored in the common storage element*** [*each processor executes* ***initialization code*** *stored in the designated device that stores data and that can be accessed by both* ***processors***];
(b) each processor performing said processor initialization code to initialize itself; and	(b) each processor ***performing*** said processor ***initialization code*** to initialize itself; and
(c) each processor determining if it is the primary processor and performing said common peripheral initialization code only if it is said primary processor.	(c) each processor determining if it is the ***primary processor*** and ***performing*** said common peripheral ***initialization code*** only if it is said ***primary processor.***
Claim 16	
16. The multiprocessor computer system of claim 11, wherein said initialization code when executed by said processors causes said processors to further perform the steps of:	16. The multiprocessor computer system of claim 11, wherein said ***initialization code*** when executed by said ***processors*** causes said ***processors*** to further perform the steps of:
(h) applying and releasing a system reset to the multiprocessor computer system before steps (a)-(c), the system reset causing all but one of said processors to be restrained; and	(h) applying and releasing a ***system reset*** to the multiprocessor computer system before steps (a)-(c), the ***system reset*** causing all but one of said ***processors*** to be ***restrained;*** and
(i) said primary processor causing each other processor to be released.	(i) said ***primary processor*** causing each other ***processor*** to be released.
Claim 17	
17. The multiprocessor computer system of claim 16, wherein the multi-processor computer system further includes an active processor identifying	17. The multiprocessor computer system of claim 16, wherein the multi-processor computer system further includes an ***active processor identifying value*** and

value and wherein said initialization code when executed by said processors causes said processors to further perform the steps of:	wherein said ***initialization code*** when executed by said ***processors*** causes said ***processors*** to further perform the steps of:
(j) each processor acquiring the active processor identifying value; and	(j) each processor acquiring the ***active processor identifying value;*** and
(k) each processor selecting a portion of said initialization code to execute based on the active processor identifying value,	(k) each processor selecting a portion of said ***initialization code*** to execute based on the ***active processor identifying value,***
wherein each processor is identified as a primary or secondary processor, and	wherein each processor is identified as a primary or secondary processor, and
wherein each processor identified as a said primary processor initializes the common peripherals and each processor identified as a secondary processor does not initialize the common peripherals.	wherein each processor identified as a said ***primary processor*** initializes the ***common peripherals*** and each processor identified as a secondary processor does not initialize the ***common peripherals.***
Claim 18	
18. The multiprocessor computer system of claim 16, wherein the multi-processor computer system further includes a main memory for storing redirection vectors and wherein step (i) of said initialization code when executed by said processors causes said processors to further perform the steps of:	18. The multiprocessor computer system of claim 16, wherein the multi-processor computer system further includes a main memory for storing ***redirection vectors*** and wherein step (i) of said ***initialization code*** when executed by said ***processors*** causes said ***processors*** to further perform the steps of:
(1) said primary processor writing an address into a redirection vector location of main memory, said address pointing to a starting address of a portion of said initialization code not causing the common peripherals to be initialized; and	(1) said ***primary processor writing an address*** into a ***redirection vector location of main memory,*** said address pointing to a starting address of a portion of said ***initialization code*** not causing the ***common peripherals*** to be initialized; and
(m) said primary processor causing each other processor to be released after setting the redirection vector.	(m) said ***primary processor*** causing each other processor to be released after ***setting the redirection vector.***

EXHIBIT B

UNITED STATES PATENT NUMBER 5,596,759-GLOSSARY OF TERMS

TERM	***DEFINITION***
Active processor identifying value	the value that indicates which one of the processors, if any, is currently active
Common peripherals	hardware devices, other than the computer (processor, memory, and data paths), such as a hard disk drive or printer, that can be accessed by both processors
Common storage element	a device that stores data and that can be accessed by both processors
Each processor executing initialization code stored in the common storage element	each processor executes initialization code stored in the designated device that stores data and that can be accessed by both processors
Each processor executing said initialization code stored in the	each processor executes initialization code stored in the designated device that stores data and that can be accessed by both processors

common storage element	
Initialization	the process by which the computer system is made ready for use
Initialization code	computer code used to start up the computer system
Initialization code having processor and common peripheral initialization code	the initialization code has a section that is capable of initializing processors and a section that is capable of initializing common peripherals
Performing	executing
Primary processor	the processor that performs a complete initialization, among other things
Processors	circuitry that has the ability to fetch, decode, and execute instructions and to transfer information to and from other resources over the computer's main data-transfer path, the bus
Redirection vectors	an address obtained from a memory location that directs the processor to a new memory location containing code to be executed
Redirection vector location of main memory	a location in main memory where a redirection vector is stored
Restrained	held in an inactive state
Said processors being powered up together	power is supplied to both processors at the same time
Setting the redirection vector	storing the address of code to be executed in the redirection vector location
System reset	a signal that causes the system to return to the state it would be in if the power were turned off and then on again
The initialization code having processor and common peripheral portions	the initialization code has a section that is capable of initializing processors and a section that is capable of initializing common peripherals

FN1. All terms appearing in bold face type and underlined have been construed by the court and appear with their definitions in the glossary in Exhibit B. The definition for each construed term appears in italics after its first use in the patent.

S.D.Cal.,2005.
Hewlett-Packard Development Company, L.P. v. Gateway, Inc.