

**IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE**

OPTIMORPHIX, INC.,

Plaintiff,

v.

INTEL CORPORATION,

Defendant.

C.A. No. 24-1291-MN

JURY TRIAL DEMANDED

FIRST AMENDED COMPLAINT FOR PATENT INFRINGEMENT

OptiMorphix, Inc. (“OptiMorphix” or “Plaintiff”) brings this action and makes the following allegations of patent infringement relating to U.S. Patent Nos.: 7,099,273 (the “‘273 Patent”); 8,521,901 (the “‘901 Patent”); 10,412,388 (the “‘388 Patent”); 9,894,361 (the “‘273 Patent”); 10,123,015 (the “‘015 Patent”); 9,621,896 (the “‘896 Patent”); and 9,749,713 (the “‘713 Patent” and collectively, the “Patents-in-Suit”). Defendant Intel Corporation (“Intel” or “Defendant”) infringes the Patents-in-Suit in violation of the patent laws of the United States of America, 35 U.S.C. § 1 *et seq.*

THE PARTIES

1. Plaintiff OptiMorphix, Inc. (“Plaintiff” or “OptiMorphix”) is a Delaware corporation that holds a portfolio of over 250 patent assets that were developed at Citrix Systems, Inc. (“Citrix”) and Bytemobile, Inc.

2. Bytemobile, Inc. (“Bytemobile”) was a global leader in mobile internet solutions for network operators. The company was founded in 2000. Bytemobile’s mission was to optimize video and web content services for mobile network operators to improve users’ experiences while maximizing the efficiency of network infrastructure.

3. Bytemobile was established during a time when the mobile landscape was evolving rapidly. The advent of 3G technology, coupled with increasingly sophisticated smartphones, led to a surge in demand for data services. However, mobile networks at the time were not optimized to handle this influx, particularly for data-rich services like video streaming. Recognizing this opportunity, Bytemobile sought to create solutions that would enable network operators to deliver high-quality, consistent mobile data services. By 2011, Bytemobile was a “market leader in video and web optimization, with more than 125 cumulative operator deployments in 60 countries.”¹



Andrew Zipern, *Vodafone in Deal with Start-Up Bytemobile*, NYTimes at C4 (January 29, 2002) (“Bytemobile, a wireless data start-up . . . reached a deal with Vodafone, Britain’s largest mobile phone operator”); *NTT DoCoMo Launches Bytemobile Optimization Solution in its Core Network*, WIRELESSWATCH IP (October 5, 2004) (“NTT DoCoMo has deployed Bytemobile’s optimization solution in its core network”); *China Mobile Selects Bytemobile for Nationwide Web Gateway Project*, BUSINESS WIRE (July 8, 2009) (“A Bytemobile customer since 2004, CMCC has deployed its web optimization solutions”); *Bytemobile Juices Up Orange*, ESPICOM TELECOMMUNICATION NEWS (October 10, 2002) (“Orange customers will experience faster application performance and Web page downloads”); *ByteMobile Wins 2013 LTE Award for Best LTE Traffic Management Product*, MARKETSCREENER (July 1, 2013) (“ByteMobile technology has been deployed . . . in networks serving nearly two billion subscribers.”).

¹ *Bytemobile: Importance of Video and Web Optimizations*, TELECOM REVIEW at 58 (2011); see also *Bytemobile Secures Its 36th Video Optimisation Win for MNO Deployment*, TOTAL TELECOM & TOTAL TELECOM MAGAZINE (March 21, 2011).

4. Bytemobile products, such as the Unison platform and the T3100 Adaptive Traffic Manager, were designed to optimize mobile data traffic in real-time, ensuring a high-quality mobile internet experience for end-users. This approach was groundbreaking at the time and set the stage for many of the mobile data optimization techniques used today.

5. Bytemobile’s innovative technologies and customer-centric approach led to rapid growth and success. Bytemobile’s innovative product portfolio included: the T3100 Adaptive Traffic Manager which was designed to handle high volumes of traffic efficiently and provide real-time optimization, compression, and management of mobile data; Bytemobile’s T2000 Series Video Cache, which supported transparent caching of content; and Bytemobile’s T1000 Series Traffic Director, which enabled traffic steering and load balancing for high availability of applications.

T3100 Adaptive Traffic Manager

The ByteMobile T3100 Adaptive Traffic Manager is the cornerstone of the ByteMobile Adaptive Traffic Management Solution. As the central “brain” for Adaptive Traffic Management, the T3100 system leverages ByteMobile applications and integrates deep packet inspection (DPI), video, web and Internet radio optimization, analytics and policy control to dynamically adapt to changing network conditions and ensure mobile subscribers have the best user experience possible.

The T3100 incorporates the ByteMobile Orchestration System, allowing the T3100 to act as a single network element for the above applications. This eliminates the cost and complexity of deploying and managing multiple network elements from different vendors for traffic management. Acting as an intelligent, content-aware control point between the Internet and the mobile network, the T3100 improves the utilization and performance of existing mobile network capacity by 30-50%.

The T3100 is a 12 RU, carrier-grade, NEBS Level 3-compliant, fault-tolerant system with built-in

T2000 Series Video Cache

The T2000 Series Video Cache improves subscriber quality of experience (QoE) and reduces data volume by delivering popular content from within the mobile operator’s network. The T2000 integrates with the T3100 to deliver superior video quality by leveraging both offline and online video optimization and supporting policy enforcement on a per-subscriber basis.

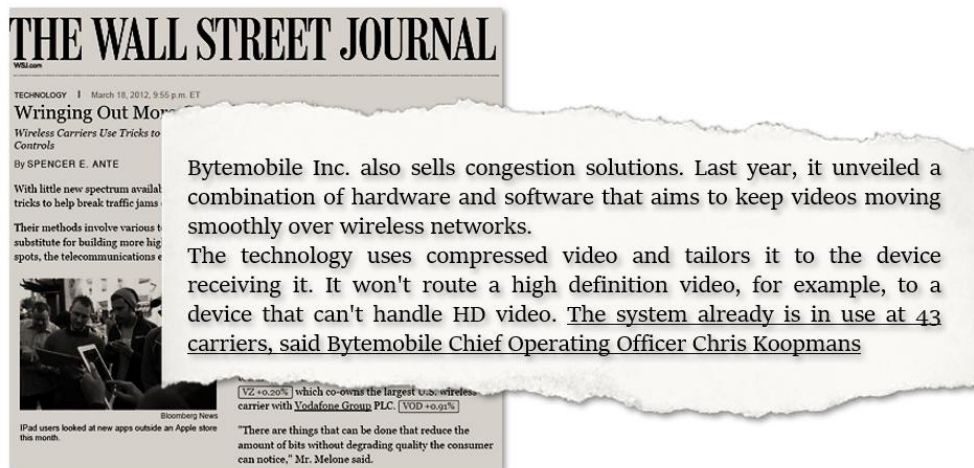
The T2000 supports transparent caching and can process traffic from every major website without requiring any changes in content server configuration. The T2000 caches up to 60% of video data volume on average, reducing the need for videos to be fetched across Internet links. Because the T2000 is tightly integrated with the ByteMobile video optimization application, operators can compress cached videos by up to 40%, providing additional data reduction for heavily constrained networks or fulfilling a mandate for intelligent capacity growth.

T1000 Series Traffic Director

The T1000 Series Traffic Director steers traffic and manages load for the T3100 platform and other operator elements on the data plane, control plane and application plane. The T1000 facilitates network integration and intelligently maintains high availability for applications running on the T3100. The T1000 offers deployment flexibility to rapidly insert Adaptive Traffic Management applications to control subscriber mobile data traffic.

ByteMobile Adaptive Traffic Management Product Family, BYTEMOBILE DATA SHEET at 1-2 (2014).

6. Bytemobile's groundbreaking technologies also included products for data optimization. Bytemobile's data optimization solutions were designed to compress and accelerate data transfer. By reducing the size of data packets without compromising quality, these technologies allowed faster data transmission and minimized network congestion. Bytemobile also offered solutions to analyze and manage network traffic, allowing network operators to identify patterns, allocate bandwidth intelligently, and prioritize different types of content.



Spencer E. Ante, *Wringing Out More Capacity*, WALL STREET JOURNAL at B3 (March 19, 2012) (emphasis added).

7. In July 2012, Bytemobile was acquired by Citrix Systems, Inc. ("Citrix") for \$435 million. Bytemobile "became part of [Citrix's] Enterprise division and extend[ed] [Citrix's] industry reach into the mobile and cloud markets."²

8. OptiMorphix owns a portfolio of patents developed at Bytemobile and later Citrix. Highlighting the importance of the patents-in-suit is the fact that the OptiMorphix's patent portfolio has been cited by over 4,800 U.S. and international patents and patent applications

² CITRIX SYSTEMS, INC. 2012 ANNUAL REPORT at 33 (2013).

assigned to a wide variety of the largest companies operating in the networking, content delivery, and cloud computing fields. OptiMorphix's patents have been cited by companies such as:

- Amazon.com, Inc. (263 citing patents and applications)³
- Oracle (59 citing patents and applications)⁴
- Alphabet, Inc. (103 citing patents and applications)⁵
- Broadcom Ltd. (93 citing patents and applications)⁶
- Cisco Systems, Inc. (277 citing patents and applications)⁷
- Lumen Technologies, Inc. (77 citing patents and applications)⁸
- **Intel Corporation** (49 citing patents and applications)⁹
- Microsoft Corporation (150 citing patents and applications)¹⁰
- AT&T, Inc. (93 citing patents and applications)¹¹
- Verizon Communications, Inc. (31 citing patents and applications)¹²
- Juniper Networks, Inc. (29 citing patents and applications)¹³

9. Defendant Intel Corporation ("Intel"), is a Delaware corporation with its principal place of business at 2200 Mission College Blvd., Santa Clara, California 95054. Intel may be served through its registered agent The Corporation Trust Company, 1209 Orange Street, Wilmington, Delaware 19801.

JURISDICTION AND VENUE

10. This action arises under the patent laws of the United States, Title 35 of the United States Code. Accordingly, this Court has exclusive subject matter jurisdiction over this action under 28 U.S.C. §§ 1331 and 1338(a).

³ See e.g., U.S. Patent Nos. 7,817,563; 9,384,204; 9,462,019; 11,343,551; and 11,394,620.

⁴ See e.g., U.S. Patent Nos. 7,475,402; 7,574,710; 8,589,610; 8,635,185; and 11,200,240.

⁵ See e.g., U.S. Patent Nos. 7,743,003; 8,458,327; 9,166,864; 9,665,617; and 10,733,376.

⁶ See e.g., U.S. Patent Nos. 7,636,323; 8,448,214; 9,083,986; 9,357,269; and 10,091,528.

⁷ See e.g., U.S. Patent Nos. 7,656,800; 7,930,734; 8,339,954; 9,350,822; and 10,284,484.

⁸ See e.g., U.S. Patent Nos. 7,519,353; 8,315,179; 8,989,002; 10,511,533; and 11,233,740.

⁹ See e.g., U.S. Patent Nos. 7,394,809; 7,408,932; 9,515,942; 9,923,821; and 10,644,961.

¹⁰ See e.g., U.S. Patent Nos. 8,248,944; 9,071,841; 9,852,118; 10,452,748; and 11,055,47.

¹¹ See e.g., U.S. Patent Nos. 8,065,374; 8,429,302; 9,558,293; 9,800,638; and 10,491,645.

¹² See e.g., U.S. Patent Nos. 8,149,706; 8,930,559; 9,253,231; 10,003,697; and 10,193,942.

¹³ See e.g., U.S. Patent Nos. 8,112,800; 8,509,071; 8,948,174; 9,407,726; and 11,228,631.

11. This Court has personal jurisdiction over Intel in this action because Intel has committed acts within the State of Delaware giving rise to this action and has established minimum contacts with this forum such that the exercise of jurisdiction over Defendant would not offend traditional notions of fair play and substantial justice. Intel, directly and/or through subsidiaries or intermediaries (including distributors, retailers, and others), has committed and continues to commit acts of infringement in this District by, among other things, offering to sell and selling products and/or services that infringe the patents-in-suit. Moreover, Intel actively directs its activities to customers located in the State of Delaware.

12. The Court further has personal jurisdiction over Intel because it is organized and existing under the laws of the State of Delaware and maintains a registered agent in Delaware.

13. Venue is proper in this District under 28 U.S.C. §§ 1391(b)-(d) and 1400(b). Defendant is organized and existing under the laws of the State of Delaware.

THE ASSERTED PATENTS

U.S. PATENT NO. 7,099,273

14. U.S. Patent No. 7,099,273 entitled, *Data Transport Acceleration and Management Within a Network Communication System*, was filed on January 29, 2002. The '273 Patent is subject to a 35 U.S.C. § 154(b) term extension of 1,021 days. The '273 Patent claims priority to U.S. Provisional Patent Application No. 60/309,212 filed on July 31, 2001, and U.S. Provisional Patent Application No. 60/283,542 filed on April 12, 2001. A true and correct copy of the '273 Patent is attached hereto as Exhibit 1.

15. The '273 Patent has been in full force and effect since its issuance. OptiMorphix, Inc. owns by assignment the entire right, title, and interest in and to the '273 Patent.

16. The technologies disclosed in the ‘273 Patent improve the efficiency and speed of data transmission within network communication systems. The ‘273 Patent introduces methods and apparatuses that enhance data transport, especially in environments where network conditions are variable or unpredictable and “provide systems and method for data transport acceleration and management within a network communication system.” (‘273 Patent, col. 3:31-33.)

17. The ‘273 Patent is directed to solving the problem of inefficient data transport within network communication systems. This inefficiency can lead to poor utilization of network resources, increased latency, and reduced overall performance.

18. The ‘273 Patent identifies the shortcomings of the prior art. Specifically, the specification describes that traditional methods of data transport in network communication systems often fail to efficiently manage and accelerate data transport, especially in environments with variable or unpredictable network conditions. These methods may not adequately handle network congestion, leading to poor utilization of network resources, increased latency, and reduced overall performance. “This bursty nature of data transmission may under-utilize the available bandwidth on the downlink channel, and may cause some applications requiring a steady flow of data, such as audio or video, to experience unusually poor performance.” (‘273 Patent, col. 2:1-6.)

19. The ‘273 Patent identifies several shortcomings of the prior art, particularly in the context of the Transport Control Protocol (TCP) which is commonly used in modern data communication networks. The patent specification describes that:

Many of the problems associated with conventional TCP architectures stem from the flow control, congestion control and error recovery mechanisms used to control transmission of data over a communication network.

(‘273 Patent, col. 1:38-41.)

20. Conventional TCP architectures assume that the network employs symmetric communication channels that enable data packets and acknowledgements to be equally spaced in time. This assumption often does not hold true in networks that employ asymmetric uplink and downlink channels, such as wireless communication networks. Bursty data transmission might result in the inefficient use of the available bandwidth on the downlink channel, leading to suboptimal performance in applications that need a consistent data flow, such as those involving audio or video.

21. Another shortcoming identified is that conventional TCP architectures react to both random loss and network congestion by significantly and repeatedly reducing the congestion window, which can lead to significant and potentially unjustified deterioration in data throughput. This is particularly problematic in wireless and other bandwidth constrained networks where random packet loss due to fading, temporary degradation in signal quality, signal handoffs or large propagation delays occur with relatively high frequency.

22. The '273 Patent also points out that conventional TCP congestion control mechanisms tend to exhibit sub-optimal performance during initialization of data connections over reduced-bandwidth channels, such as wireless links. When a connection is initiated, the congestion control mechanism aggressively increases the size of the congestion window until it senses a data packet loss. This process may adversely impact other connections that share the same reduced-bandwidth channel as the connection being initialized attempts to maximize its data throughput without regard of the other pre-existing connections. This can lead to inefficient use of resources with decreased overall throughput.

23. The '273 Patent teaches the use of various techniques to accelerate and manage data transport in network communication systems. These techniques include the use of congestion

control mechanisms, timers, and other methods to optimize data transmission. By implementing these techniques, the patent aims to improve the efficiency of data transport, particularly in environments with variable or unpredictable network conditions. This can lead to better utilization of network resources, reduced latency, and improved overall performance. The inventions disclosed in the '273 Patent provide significant benefits and improvements to the function of the hardware in a computer network.

24. On March 8, 2024, Unified Patents, LLC filed a Request for *Ex Parte* Reexamination of the '273 Patent with the United States Patent and Trademark Office. The Patent Office entered an Order Granting *Ex Parte* Reexamination of the '273 Patent on April 29, 2024. On September 11, 2024, the Primary Examiner assigned to the Reexamination of the '273 Patent issued an Order confirming the patentability of all claims of the '273 patent. On November 14, 2024, the United States Patent and Trademark Office issued *Ex Parte* Reexamination Certificate No. 12770 confirming the patentability of Claims 1-15 of the '273 Patent. A true and correct copy of that Certificate is attached hereto as Exhibit 2.

25. The '273 Patent family has been cited by 1,466 United States and international patents and patent applications as relevant prior art. Specifically, patents issued to the following companies and research institutions have cited the '273 Patent family as relevant prior art:

- Cisco Technology, Inc.
- Qualcomm Incorporated
- International Business Machines Corporation
- ***Intel Corporation***
- Microsoft Corporation
- Broadcom Corporation
- Google Inc.
- F5 Networks, Inc.
- Adobe Systems Incorporated
- Apple Inc.
- Lumen Technologies, Inc
- Oracle Corporation

- Amazon.com, Inc.

U.S. PATENT NO. 8,521,901

26. U.S. Patent No. 8,521,901 entitled, *TCP Burst Avoidance*, was filed on December 22, 2008. The '901 Patent claims priority to Provisional Patent Application No. 61/017,275, filed on December 28, 2007. The '901 Patent is subject to a 35 U.S.C. § 154(b) term extension of 525 days. A true and correct copy of the '901 Patent is attached hereto as Exhibit 3.

27. The '901 Patent has been in full force and effect since its issuance. OptiMorphix, Inc. owns by assignment the entire right, title, and interest in and to the '901 Patent.

28. The '901 Patent generally relates to methods and systems for minimizing packet bursts. The '901 Patent teaches implementing a packet scheduler layer between the network layer and the transport layer of a device, which smooths the delivery of TCP packets by delaying their delivery, thus addressing the challenges posed by the rapid and bursty transmission of data packets in network communications.

29. The '901 Patent is directed to solving the problem of TCP packet bursts in high-speed data networks, which can result from the buffering of TCP acknowledgment packets. These bursts can cause packet loss and inefficient use network bandwidth.

30. The '901 Patent identifies the shortcomings of the prior art. Specifically, the specification describes that the prior art does not adequately address the issues of packet loss and inefficient bandwidth utilization resulting from the bursty nature of TCP packet transmission in data networks. The prior technologies do not effectively manage the sudden bursts of TCP acknowledgment packets, which can be caused by buffering, leading to suboptimal utilization of available bandwidth and undesirable packet loss.

31. The '901 Patent teaches the use of a packet scheduler layer, which is positioned between the network and transport layers of a device. This layer receives, smoothens (by

delaying), and sends TCP packets to ensure that the delivery of these packets is managed in a manner that mitigates the issues of packet bursts. The packet scheduler layer manages both incoming and outgoing packets, ensuring that the transmission of these packets is smoothed out, thereby minimizing packet loss and ensuring more efficient use of available bandwidth. This approach provides benefits that differ from conventional methods by ensuring that TCP packet transmission is managed in a way that minimizes packet loss and ensures efficient bandwidth utilization, thereby addressing the specific challenges posed by TCP packet bursts in high-speed data networks.

32. The invention taught by the '901 Patent solves discrete, technological problems associated with computer systems; specifically, it addresses the issues of packet loss and inefficient bandwidth utilization in high-speed data networks by managing the transmission of TCP packets in a manner that smoothens their delivery, thereby ensuring that the available bandwidth is utilized efficiently, and that packet loss is minimized.

33. The '901 Patent family has been cited by 21 United States and international patents and patent applications as relevant prior art. Specifically, patents issued to the following companies have cited the '901 Patent family as relevant prior art:

- Lenovo Group Limited
- Telefonaktiebolaget Lm Ericsson
- Qualcomm, Inc.
- Nippon Telegraph & Telephone Corp.
- Hitachi, Ltd.
- Cisco Systems, Inc.
- Akamai Technologies, Inc.
- Huawei Technologies Co., Ltd.

U.S. PATENT NO. 10,412,388

34. U.S. Patent No. 10,412,388 entitled, *Framework for Quality-Aware Video Optimization*, was filed on January 8, 2018. The '388 Patent claims priority to U.S. Patent Application No. 12/751,951, which was filed on March 31, 2010, and which claims priority to U.S. Provisional Patent Application No. 61/165,224, which was filed on March 31, 2009. A true and correct copy of the '388 Patent is attached hereto as Exhibit 4.

35. The '388 Patent has been in full force and effect since its issuance. OptiMorphix, Inc. owns by assignment the entire right, title, and interest in and to the '388 Patent.

36. The '388 Patent generally relates to a method and system for quality-aware video optimization. It teaches receiving an encoded video frame, decompressing it, extracting a first quantization parameter (QP), and acquiring a delta QP based on the first QP. The method also includes acquiring a second QP based on the delta QP and the first QP, compressing the decompressed video frame based on the second QP, and providing the compressed video frame. The process allows for fine control of quality degradation in byte-reduced content and can be applied to transcoding scenarios where the input and output compression formats are different.

37. The '388 Patent identifies the shortcomings of the prior art. Specifically, existing single-pass rate control techniques had a problem in that the relationship between the compressed byte size of a video frame and its quantization parameter were only known after the frame is encoded. This made it challenging to achieve byte reduction and controllable quality degradation in a single pass.

38. The '388 Patent teaches the use of a quality-aware video optimization technique that modifies a video frame sequence to reduce the byte size while limiting perceptual quality degradation to a controllable level.

39. The inventions disclosed in the '388 Patent provide significant benefits and improvements to the function of hardware in a computer network by enabling efficient video optimization. The method allows for single-pass, on-the-fly quality-aware optimization, making it well-suited for various environments, including live video feeds and storage arrays.

40. The '388 patent family has been cited by 30 United States and international patents and patent applications as relevant prior art. Specifically, patents issued to the following companies and research institutions have cited the '388 Patent family as relevant prior art:

- Interdigital, Inc.
- Tencent Holdings Ltd
- Microsoft Corporation
- Qualcomm, Inc.
- Lattice Semiconductor
- Openwave Mobility, Inc.
- Samsung Electronics Co., Ltd.
- Beijing Dajia Interconnection Information Technology Co., Ltd.

U.S. PATENT NO. 9,894,361

41. U.S. Patent No. 9,894,361 entitled, *Framework for Quality-Aware Video Optimization*, was filed on March 31, 2010. The '361 Patent claims priority to U.S. Provisional Application No. 61/165,224, which was filed on March 31, 2009. The '361 Patent is subject to a 35 U.S.C. § 154(b) term extension of 1,038 days. A true and correct copy of the '361 Patent is attached hereto as Exhibit 5.

42. The '361 Patent has been in full force and effect since its issuance. OptiMorphix, Inc. owns by assignment the entire right, title, and interest in and to the '361 Patent.

43. The '361 Patent relates to a method and system for quality-aware video optimization. Specifically, it teaches receiving an encoded video frame, decompressing it, extracting a first quantization parameter (QP), and acquiring a delta QP based on the first QP. The method further includes acquiring a second QP based on the delta QP and the first QP, compressing

the decompressed video frame based on the second QP, and providing the compressed video frame. The process is designed to reduce the byte size of the video stream as much as possible while limiting perceptual quality degradation to a controllable level.

44. The '361 Patent is directed to solving the problem of optimizing video quality in a way that balances the reduction of byte size with the preservation of perceptual quality. This involves a nuanced understanding of how quantization parameters (QPs) affect both the perceptual quality and the bitrate of a video frame, and how to manipulate these QPs to achieve the desired balance.

45. The '361 Patent identifies the shortcomings of the prior art. Specifically, existing single-pass rate control techniques had a problem in that the relationship between the compressed byte size of a video frame and its quantization parameter was only known after the frame was encoded. This made it challenging to achieve byte reduction and controllable quality degradation in a single pass.

46. The '361 Patent teaches the use of a quality-aware video optimization technique that requires only a single pass over the previously encoded video frame sequence to optimize the video frame sequence. It introduces a novel function that defines ΔQP according to the value of QP_{Input} , allowing fine control of quality degradation in the byte-reduced content. It also considers differences between input and output compression formats (codecs) and computes codec adjustment that accounts for these differences.

47. The inventions disclosed in the '361 Patent provide significant benefits and improvements to the function of hardware in a computer network by enabling efficient video optimization. By allowing for single-pass, on-the-fly, quality-aware optimization, the patent's methods can be applied in various environments, including optimizing live video feeds before they

traverse a low-capacity network segment, or optimizing surveillance video before archiving, thus saving storage space and network bandwidth.

48. The '361 Patent family has been cited by 30 United States and international patents and patent applications as relevant prior art. Specifically, patents issued to the following companies and research institutions have cited the '361 Patent family as relevant prior art:

- Interdigital, Inc.
- Tencent Holdings Ltd
- Microsoft Corporation
- Qualcomm, Inc.
- Lattice Semiconductor
- Openwave Mobility, Inc.
- Samsung Electronics Co., Ltd.
- Beijing Dajia Interconnection Information Technology Co., Ltd.

U.S. PATENT NO. 10,123,015

49. U.S. Patent No. 10,123,015 entitled, *Macroblock-Level Adaptive Quantization in Quality-Aware Video Optimization*, was filed on April 10, 2017. The '015 Patent claims priority to U.S. Application No. 13/492,619, which was filed on June 8, 2012, and which issued as U.S. Patent No. 9,621,896. The '015 Patent claims priority to U.S. Provisional Application No. 61/495,951, which was filed on June 10, 2011. A true and correct copy of the '015 Patent is attached hereto as Exhibit 6.

50. The '015 Patent has been in full force and effect since its issuance. OptiMorphix, Inc. owns by assignment the entire right, title, and interest in and to the '015 Patent.

51. The '015 Patent teaches systems and methods for macroblock-level quality-aware video optimization. Unlike traditional methods that apply uniform compression settings across video frames, this approach focuses on adjusting compression at the macroblock level—small sections of a video frame. By analyzing each macroblock's visual and compression characteristics,

the system dynamically determines the appropriate quantization parameter (QP) to optimize the balance between file size and visual quality.

52. The technologies taught in the '015 Patent address the challenge of efficiently compressing video while preserving critical details, especially in regions with high visual complexity. Traditional methods fail to account for differences within a frame, leading to unnecessary quality degradation or inefficient compression. By tailoring QP settings to individual macroblocks, this system ensures better preservation of important visual elements, such as faces or text, while reducing the overall data size.

53. The '015 Patent provides significant benefits and improvements to the function of the hardware in a computer network by significantly reducing the necessary bitrate necessary to transmit video data with minimal perceptual quality loss. This is critical for computer networks to be able to deliver live-streamed video content and to deliver video content through low-bandwidth networks.

54. The '015 Patent has been cited by at least 29 United States and international patents and patent applications as relevant prior art. Specifically, patents issued to the following companies and research institutions have cited the '015 Patent family as relevant prior art:

- Microsoft Technology Licensing, LLC
- Google LLC
- Apple Inc.
- Samsung Electronics Co., Ltd.
- Qualcomm Incorporated
- Netflix, Inc.
- Sharp Corporation
- Tencent Technology (Shenzhen) Co., Ltd.
- Huawei Technologies Co., Ltd.
- Magnum Semiconductor, Inc.
- Integrated Device Technology, Inc.

U.S. PATENT NO. 9,621,896

55. U.S. Patent No. 9,621,896 entitled, *Macroblock-Level Adaptive Quantization in Quality-Aware Video Optimization*, was filed on June 8, 2012. The '896 Patent claims priority to U.S. Provisional Application No. 61/495,951, which was filed on June 10, 2011. A true and correct copy of the '896 Patent is attached hereto as Exhibit 8.

56. The '896 Patent has been in full force and effect since its issuance. OptiMorphix, Inc. owns by assignment the entire right, title, and interest in and to the '896 Patent.

57. The '896 Patent relates to macroblock-level quality-aware video optimization, a method for improving video compression efficiency while maintaining perceptual quality. The '896 Patent discloses technologies for decoding video frames into small pixel blocks (macroblocks), analyzing visual and compression characteristics, and dynamically adjusting the quantization parameter (QP) for each block to optimize quality and file size.

58. The '896 Patent addresses the challenge of compressing video streams effectively without degrading user experience. Traditional methods apply uniform compression across entire frames, leading to unnecessary quality loss in detailed regions or excessive data usage in simpler regions. The technologies disclosed in the '896 Patent introduce precise control at the macroblock level, balancing file size reduction with quality retention.

59. One key benefit of the technologies disclosed in the '896 Patent that is important to the ubiquitousness of the distribution of video content over the internet and cellular networks is the ability to improve important visual details while minimizing the bandwidth necessary to transmit the video data.

60. The '896 Patent has been cited by at least 29 United States and international patents and patent applications as relevant prior art. Specifically, patents issued to the following companies and research institutions have cited the '896 Patent family as relevant prior art:

- Microsoft Technology Licensing, LLC
- Apple Inc.
- Qualcomm Incorporated
- Netflix, Inc.
- Google LLC
- Sharp Kabushiki Kaisha
- Huawei Technologies Co., Ltd.
- Magnum Semiconductor, Inc.
- Tencent Technology (Shenzhen) Co., Ltd.
- Integrated Device Technology, Inc.

U.S. PATENT NO. 9,749,713

61. U.S. Patent No. 9,749,713 entitled, *Budget Encoding*, was filed on October 15, 2009. The ‘713 Patent claims priority to U.S. Patent Application No. 12/580,212, which was filed on October 15, 2009. The ‘713 Patent is subject to a 35 U.S.C. § 154(b) term extension of 1654 days. A true and correct copy of the ‘713 Patent is attached hereto as Exhibit 9.

62. The ‘713 Patent has been in full force and effect since its issuance. OptiMorphix, Inc. owns by assignment the entire right, title, and interest in and to the ‘713 Patent.

63. The ‘713 Patent is directed to solving the problem of inefficient allocation and management of network resources. Traditional methods often fail to prioritize applications effectively, leading to suboptimal performance for critical applications and inefficient utilization of network resources.

64. The ‘713 Patent identifies the shortcomings of the prior art. Specifically, existing systems lacked the ability to dynamically allocate resources based on real-time needs and priorities of applications. This leads to either over-provisioning, which wastes resources, or under-provisioning, which can cause critical applications to suffer or fail.

65. The ‘713 Patent teaches the use of a resource manager that dynamically allocates network resources to applications based on a set of defined policies and priority levels. It involves continuous monitoring of the network and applications, and the resource manager makes real-time

decisions to allocate or deallocate resources as needed. This ensures that critical applications always have the necessary resources, while other applications receive resources as available.

66. The inventions disclosed in the '713 Patent provide significant benefits and improvements to the function of the hardware in a computer network. By implementing a dynamic, policy-driven approach to resource allocation, the system ensures optimal performance for critical applications and efficient utilization of network resources. This leads to improved overall network performance, reduced waste, and the ability to adapt to changing conditions and demands.

67. The '713 Patent family has been cited by 41 United States and international patents and patent applications as relevant prior art. Specifically, patents issued to the following companies and research institutions have cited the '713 Patent family as relevant prior art:

- Samsung Electronics Co., Ltd.
- Openwave Mobility, Inc.
- Huawei Investment & Holding Co., Ltd.
- Cisco Systems, Inc.
- Flash Networks Ltd.
- ZTE Corporation
- Vizio, Inc.
- Wangsu Science & Technology Co., Ltd.
- Akamai Technologies, Inc.
- SK Telecom Co., Ltd.
- Sugon Information Industry(Beijing) Co., Ltd.
- Netscout Systems, Inc.
- Microsoft Corporation
- Telefonaktiebolaget Lm Ericsson

COUNT I
INFRINGEMENT OF U.S. PATENT NO. 7,099,273

68. Plaintiff references and incorporates by reference the preceding paragraphs of this Complaint as if fully set forth herein.

69. Intel designs, makes, uses, sells, and/or offers for sale in the United States products comprising systems and methods for data transport acceleration and management within a network communication system.

70. Intel designs, makes, sells, offers to sell, imports, and/or uses software, including but not limited to versions of the Linux kernel (version 4.9 and later) incorporating TCP-BBR congestion control algorithms, Intel designs, makes, sells, offers to sell, imports, and/or uses processors, networking hardware, and other computing devices designed, marketed, or distributed for use with software employing TCP-BBR congestion control algorithms (this includes but is not limited to Intel's processors, chipsets, and other hardware components that are integrated with or otherwise configured to operate with the Linux kernel supporting TCP-BBR) (collectively, the "Intel '273 Product(s)").

71. One or more Intel subsidiaries and/or affiliates use the Intel '273 Products in regular business operations.

72. One or more of the Intel '273 Products include technology that performs the step of establishing a data connection between a sender and receiver using a handshake process. Specifically, the Intel '273 Products perform TCP-BBR congestion control.

```

77
78  /* BBR has the following modes for deciding how fast to send: */
79  enum bbr_mode {
80      BBR_STARTUP,    /* ramp up sending rate rapidly to fill pipe */
81      BBR_DRAIN,     /* drain any queue created during startup */
82      BBR_PROBE_BW,  /* discover, share bw: pace around estimated bw */
83      BBR_PROBE_RTT, /* cut inflight to min to probe min_rtt */
84  };
85
86  /* BBR congestion control block */
87  struct bbr {
88      u32    min_rtt_us;          /* min RTT in min_rtt_win_sec window */
89      u32    min_rtt_stamp;      /* timestamp of min_rtt_us */
90      u32    probe_rtt_done_stamp; /* end time for BBR_PROBE_RTT mode */
91      struct minmax bw;          /* Max recent delivery rate in pkts/uS << 24 */
92      u32    rtt_cnt;            /* count of packet-timed rounds elapsed */
93      u32    next_rtt_delivered; /* scb->tx.delivered at end of round */
94      u64    cycle_mstamp;       /* time of this cycle phase start */
95      u32    mode:3,             /* current bbr_mode in state machine */

```

Intel Linux LTS – tcp_bbr.c, INTEL GITHUB REPOSITORY (November 2024), available at: <https://github.com/intel/linux-intel-lts> (emphasis added).

73. The Intel ‘273 Products send a TCP packet with the SYN (Synchronize) flag set to the server. This packet contains an initial sequence number (ISN), which helps the server and client synchronize their sequence numbers. The ISN used by the Intel ‘273 Products are represented as “x.” Upon receiving the SYN packet, the Intel ‘273 Products sends a TCP packet back with both the SYN and ACK flags set. This packet contains two pieces of information: the responsive ISN, usually represented as ‘y,’ and an acknowledgment number, which is the ISN plus one (x+1). The acknowledgment number is used to confirm that the sender has received the SYN packet.

74. In establishing a connection between the sender and the receiver after receiving the SYN-ACK packet, the Intel ‘273 Products send another packet with the ACK flag set. This packet contains an acknowledgment number, which is the ISN plus one (y+1).

75. The Intel '273 Products measure round trip times (RTT) of data packets sent from the sender to the receiver. Specifically, the Intel '273 Products measure the round-trip propagation time (RTprop) using the minimum round-trip time (RTT) for the connection by keeping track of the lowest observed RTT in the recent past. This value represents the round-trip propagation time (RTprop) of the connection.

76. The Intel '273 Products perform timestamping. Specifically, when a Intel '273 Product transmits a data packet, it records the current time as a timestamp. The timestamp is stored in the transmission control block (TCB), which maintains the state of the TCP connection, including RTT measurements and other relevant information.

77. The Intel '273 Products perform acknowledgment processing. Specifically, the Intel '273 Products send an acknowledgment (ACK) for a specific packet, the sender processes the ACK and identifies the corresponding packet in the TCB. By matching the ACK with the original packet, the Intel '273 Products retrieve the original timestamp associated with that packet.

78. The Intel '273 Products perform a round-trip time (RTT) calculation. Specifically, the Intel '273 Products calculate the RTT for a specific packet by subtracting the original timestamp from the current time when the ACK is received. This gives an individual RTT sample for that packet as explained in the below excerpt.

One way to stay near (max BW, min RTT) point:

Model network, update windowed **max BW and min RTT estimates on each ACK**

Control sending based on the model, to...

- Probe both max BW and min RTT, to feed the model samples
- Pace** near estimated BW, to reduce queues and loss [move queue to sender]
- Vary pacing rate to keep inflight near BDP (for full pipe but small queue)

That's **BBR** congestion control:

- BBR** = Bottleneck Bandwidth and Round-trip propagation time
- BBR seeks high tput with small queue by probing BW and RTT sequentially**

Neal Cardwell, Yunchung Cheng, et al., *BBR Congestion Control*, GOOGLE IETF 97: SEOUL PRESENTATION at 9 (November 2016) (emphasis added) (describing $RTT_sample = ACK_receive_time - original_timestamp$).

79. The Intel '273 Products perform the step of MinRTT estimation. Specifically, the Intel '273 Products maintain a running estimate of the minimum RTT observed (MinRTT) over a specified time window. The MinRTT is used by the Intel '273 Products to estimate the base round-trip propagation time without queuing delay. When a new RTT sample is calculated, the Intel '273 Products compare it with the current MinRTT value. If the new sample is lower than the existing MinRTT, the Intel '273 Products update MinRTT with a new value.

80. The Intel '273 Products perform round-trip time-based pacing. Specifically, the Intel products use the MinRTT estimate in performing pacing rate and congestion window calculations to ensure the sending rate is adapted based on the observed network conditions. BBR's pacing rate and congestion window calculations factor in the MinRTT value to maintain a balance between efficient data transfer and minimal congestion.

To match the packet-arrival rate to the bottleneck link's departure rate, BBR paces every data packet. BBR must match the bottleneck *rate*, which means pacing is integral to the design and fundamental to operation— *pacing_rate* is BBR's primary control parameter. A secondary parameter, *cwnd_gain*, bounds inflight to a small multiple of the BDP to handle common network and receiver pathologies (see the later section on Delayed and Stretched ACKs). Conceptually, the TCP send routine looks like the following code. (In Linux, sending uses the efficient FQ/pacing queuing discipline,⁴ which gives BBR line-rate single-connection performance on multigigabit links and handles thousands of lower-rate paced connections with negligible CPU overhead.)

Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, Van Jacobson, *BBR: Congestion-Based Congestion Control*, ACM Queue, Sep/Oct 2016 and CACM, Feb 2017 (emphasis added).

81. The Intel '273 Products determine a congestion window parameter that specifies a maximum number of unacknowledged data packets that may be sent to the receiver. The Intel '273 Products determine a congestion window parameter that limits unacknowledged data packets. In the Intel '273 Products, this parameter is called *cwnd* (congestion window) and is a key component of the Intel '273 Products' congestion control. The primary function responsible for determining the congestion window is *bbr_set_cwnd()*, which is called from the main control function *bbr_main()*.


```

1023     {
1024         struct bbr *bbr = inet_csk_ca(sk);
1025         u32 bw;
1026
1027         bbr_update_model(sk, rs);
1028
1029         bw = bbr_bw(sk);
1030         bbr_set_pacing_rate(sk, bw, bbr->pacing_gain);
1031         bbr_set_cwnd(sk, rs, rs->acked_sacked, bw, bbr->cwnd_gain);
1032     }

```

Intel LTS - tcp_bbr.c, INTEL GITHUB SOURCE CODE REPOSITORY, available at: https://github.com/intel/linux-intel-lts/net/ipv4/tcp_bbr.c (last visited February 2025) (emphasis added).

82. The congestion window is calculated based on several factors by the Intel ‘273 Products. These factors include: (1) the bandwidth-delay product (BDP) via the `bbr_bdp()` function, which multiplies the estimated bandwidth by the minimum RTT; (2) additional buffer for ACK aggregation via `bbr_ack_aggregation_cwnd()`; and (3) quantization adjustments via `bbr_quantization_budget()` to handle packet boundaries and delayed ACKs. The Intel ‘273 Products determine a congestion window which controls the maximum number of unacknowledged packets allowed in flight.

83. The Intel ‘273 Products determine a congestion window parameter (`cwnd`) that limits the unacknowledged data packets in transit. This parameter is not determined by setting a specific parameter for the maximum number of unacknowledged packets. Instead, the Intel ‘273 Products calculate the congestion window based on network characteristics. The determination of the congestion control parameter based on network conditions is shown in the below excerpt from Intel’s source code.

```

3  * BBR congestion control computes the sending rate based on the delivery
4  * rate (throughput) estimated from ACKs. In a nutshell:
5  *
6  *   On each ACK, update our model of the network path:
7  *       bottleneck_bandwidth = windowed_max(delivered / elapsed, 10 round trips)
8  *       min_rtt = windowed_min(rtt, 10 seconds)
9  *       pacing_rate = pacing_gain * bottleneck_bandwidth
10 *       cwnd = max(cwnd_gain * bottleneck_bandwidth * min_rtt, 4)
11 *
12 * The core algorithm does not react directly to packet losses or delays,
13 * although BBR may adjust the size of next send per ACK when loss is
14 * observed, or adjust the sending rate if it estimates there is a
15 * traffic policer, in order to keep the drop rate reasonable.

```

Intel LTS - tcp_bbr.c, INTEL GITHUB SOURCE CODE REPOSITORY, available at: https://github.com/intel/linux-intel-lts/net/ipv4/tcp_bbr.c (last visited February 2025) (emphasis added).

84. The Intel ‘273 Products determine a congestion window parameter using `bbr_set_cwnd()`. The Intel products determine this parameter using the: bandwidth-delay product (BDP): `target_cwnd = bbr_bdp(sk, bw, gain)`; additional buffer for ACK aggregation: `target_cwnd += bbr_ack_aggregation_cwnd(sk)`; and quantization adjustments: `target_cwnd = bbr_quantization_budget(sk, target_cwnd, gain)`. Thus, the Intel ‘273 products determine the congestion window parameter based on estimated network conditions rather than being directly set as a fixed parameter.

85. The Intel ‘273 Products calculate a pacing rate based on these estimates to determine how quickly it should transmit data.

86. The Intel ‘273 Products transmit additional data packets to the receiver in response to expiration of a transmit timer, the period of the transmit timer based on the round trip time measurements and the congestion window parameter.

87. Intel has directly infringed and continues to directly infringe the ‘273 Patent by, among other things, making, using, offering for sale, and/or selling technology for transferring

data from a sender to a receiver in a communication network, including but not limited to the Intel ‘273 Products.

88. The Intel ‘273 Products are available to businesses and individuals throughout the United States.

89. The Intel ‘273 Products are provided to businesses and individuals located in this District.

90. By making, using, testing, offering for sale, and/or selling products and services for transferring data from a sender to a receiver in a communication network, including but not limited to the Intel ‘273 Products, Intel has injured Plaintiff and is liable to Plaintiff for directly infringing one or more claims of the ‘273 Patent, including at least claim 1 pursuant to 35 U.S.C. § 271(a).

91. Intel also indirectly infringes the ‘273 Patent by actively inducing infringement under 35 U.S.C. § 271(b).

92. Intel has had knowledge of the ‘273 Patent since at least March 23, 2010, when U.S. Patent No. 7,684,319, which is owned by Intel and cites the ‘273 Patent family as relevant prior art, was issued. Alternatively, Intel has had knowledge of the ‘273 Patent since at least December 5, 2013, when U.S. Patent Appl. No. 13/899,935, which is owned by Intel and cites the ‘273 Patent family as relevant prior art, was published. Alternatively, Intel has had knowledge of the ‘273 Patent since at least December 10, 2013, when U.S. Patent No. 8,606,956, which is owned by Intel and cites the ‘273 Patent family as relevant prior art, was issued. Alternatively, Intel has had knowledge of the ‘273 Patent since at least December 6, 2016, when U.S. Patent No. 9,515,942, which is owned by Intel and cites the ‘273 Patent family as relevant prior art, was issued. Alternatively, Intel has had knowledge of the ‘273 Patent since at least July 11, 2017,

when U.S. Patent No. 9,705,964, which is owned by Intel and cites the ‘273 Patent family as relevant prior art, was issued.

93. Intel intended to induce patent infringement by third-party customers and users of the Intel ‘273 Products and had knowledge that the inducing acts would cause infringement or was willfully blind to the possibility that its inducing acts would cause infringement. Intel specifically intended and was aware that the normal and customary use of the accused products would infringe the ‘273 Patent. Intel performed the acts that constitute induced infringement, and would induce actual infringement, with knowledge of the ‘273 Patent and with the knowledge that the induced acts would constitute infringement. For example, Intel provides the Intel ‘273 Products that have the capability of operating in a manner that infringe one or more of the claims of the ‘273 Patent, including at least claim 1, and Intel further provides documentation and training materials that cause customers and end users of the Intel ‘273 Products to utilize the products in a manner that directly infringe one or more claims of the ‘273 Patent.¹⁴ By providing instruction and training to

¹⁴ See, e.g., *Intel Wireless Wi-Fi Drivers for Linux*, Intel Drivers & Software Website (July 29, 2024), available at: <https://www.intel.com/content/www/us/en/download/824804/intel-wireless-wi-fi-drivers-for-linux.html>; *Release Notes – Intel Wireless Wi-Fi Driver Package Cor87 for Linux*, INTEL DOCUMENTATION (July 2024); *Linux Drivers Support for Intel Killer Wireless Products*, INTEL PRODUCT SUPPORT WEBSITE (September 16, 2022), available at: <https://www.intel.com/content/www/us/en/support/articles/000088040/wireless.html>; *Intel/LT S/net/ipv4/tcp_bbr.c*, INTEL GITHUB REPOSITORY (last visited November 2024), available at: https://github.com/intel/linux-intel-lts/blob/20bd5cdc7663c3b902be4c39650ad8cdb80ac2fb/net/ipv4/tcp_bbr.c; *Intel/Mainline-Tracking/net/ipv4/tcp_bbr.c*, INTEL GITHUB REPOSITORY (last visited November 2024), available at: https://github.com/intel/mainline-tracking/blob/e0582fea0d501d507dffe716dae8d99c2f8b3f2e/net/ipv4/tcp_bbr.c; *Intel/xdp-bpf-acceleration/net/ipv4/tcp_bbr.c*, INTEL GITHUB REPOSITORY (last visited November 2024), available at: https://github.com/intel/xdp-bpf-acceleration/blob/52f1250b1f735cae7daa15e38469c436c6c4f7eb/net/ipv4/tcp_bbr.c; *Pat Gelsinger and Linus Torvalds talk Linux, open source, technology and more*, INTEL NEWSROOM YOUTUBE CHANNEL (November 2, 2022), available at: <https://www.youtube.com/watch?v=0m4hlWx7oRk>; *Linux Xeon Scalable Processors – Minimum OS Support Matrix*, Intel Documentation (April 25, 2024); *Intel Virtual RAID on CPU (Intel VROC) for Linux -Release Notes for Intel VROC 9.0*, and Intel Documentation (June 2024); *SR-IOV Configuration Guide - Intel Ethernet 800 Series on Red Hat Enterprise Linux 8 Technical Brief Revision 1.2*, Intel Documentation (June 2021).

customers and end-users on how to use the Intel ‘273 Products in a manner that directly infringes one or more claims of the ‘273 Patent, including at least claim 1, Intel specifically intended to induce infringement of the ‘273 Patent. Intel engaged in such inducement to promote the sales of the Intel ‘273 Products, e.g., through Intel user manuals, product support, marketing materials, and training materials to actively induce the users of the accused products to infringe the ‘273 Patent. Accordingly, Intel has induced and continues to induce users of the accused products to use the accused products in their ordinary and customary way to infringe the ‘273 Patent, knowing that such use constitutes infringement of the ‘273 Patent.

94. The ‘273 Patent is well-known within the industry as demonstrated by multiple citations to the ‘273 Patent in published patents and patent applications assigned to technology companies and academic institutions. Intel is utilizing the technology claimed in the ‘273 Patent without paying a reasonable royalty. Intel is infringing the ‘273 Patent in a manner best described as willful, wanton, malicious, in bad faith, deliberate, consciously wrongful, flagrant, or characteristic of a pirate.

95. To the extent applicable, the requirements of 35 U.S.C. § 287(a) have been met with respect to the ‘273 Patent.

96. As a result of Intel’s infringement of the ‘273 Patent, Plaintiff has suffered monetary damages, and seeks recovery in an amount adequate to compensate for Intel’s infringement, but in no event less than a reasonable royalty for the use made of the invention by Intel together with interest and costs as fixed by the Court.

COUNT II
INFRINGEMENT OF U.S. PATENT NO. 8,521,901

97. Plaintiff references and incorporates by reference the preceding paragraphs of this Complaint as if fully set forth herein.

98. Intel designs, makes, uses, sells, and/or offers for sale in the United States products comprising technology for a data packet scheduler that reduces packet bursts.

99. Intel designs, makes, sells, offers to sell, imports, and/or uses the following products: Intel Ethernet 800 Series Network Adapters (including models: E810-2CQDA2, E810-CQDA1, E810-CQDA1 for OCP 3.0, E810-CQDA2, E810-CQDA2 for OCP 3.0, E810-CQDA2T, E810-XXVDA2, E810-XXVDA2 for OCP 3.0, E810-XXVDA4, E810-XXVDA4 for OCP 3.0, E810-XXVDA4T) and Intel Ethernet 800 Series Controllers (including models: E810-CAM1; E810-CAM2; E810-XXVAM2) (collectively, the “Intel ‘901 Product(s)”).

100. One or more Intel subsidiaries and/or affiliates use the Intel ‘901 Products in regular business operations.

101. The Intel ‘901 Products contain a packet scheduler layer identified in Intel documentation as the “Tx-Scheduler layer.” Specifically, according to Intel documentation “[t]ransmitted packets pass through different hierarchies in the network . . . reflected in the Tx-Scheduler Layers structure.”

- Transmitted packets pass through different hierarchies in the network. **The network hierarchies are reflected in the Tx-Scheduler layers structure.** The capacity of each network unit is reflected in the rate limit and bandwidth allocation configuration of the nodes.
- Each unit in the network branch can deal with different portions of the packets.
- Typically, for the network hop, the whole packet including the IPG and the L1 header needs to be taken into account in the bandwidth calculation. Next hops, bridges, or the target station can consider a portion of the packet. For example, in tunneled environment, the target station does not see the whole packet, but only inner L4 part.
- The Tx-Scheduler is required to calculate up to four different adjustments of packets lengths for different Tx-Scheduler layers or nodes.
- **The Tx-Scheduler is also required to calculate different adjustments of packets lengths for different types of packets.**

INTEL ETHERNET CONTROLLER E810 DATASHEET REVISION 2.8 at § 8.3.2.4.1(September 2024) (emphasis added).

102. The Intel ‘901 Products implement the packet scheduler layer functionality as claimed in the ‘901 Patent. This packet scheduler layer operates within a first device, positioned

between the network interface layer and transport layer of the first device's networking stack. The Intel '901 Products contain the packet scheduler layer. The Intel '901 Products both receive the TCP packets and store the requisite connection data - operating as a first device.

103. The Intel '901 Products implement a packet scheduler layer on a first device. As detailed in Intel's documentation, the Intel '901 Products contain a packet scheduler layer residing between network interface and transport layers of the networking stack within the first device. The packet scheduler layer contained in the Intel '901 Products is a component within the first device that receives packets from another layer (either the network interface layer or transport layer) on that same device. As evidenced by the Intel documentation the Intel '901 Products implement packet processing pipelines that receive TCP packets from one layer and deliver them to another layer within the same computing system, functioning as the packet scheduler layer.

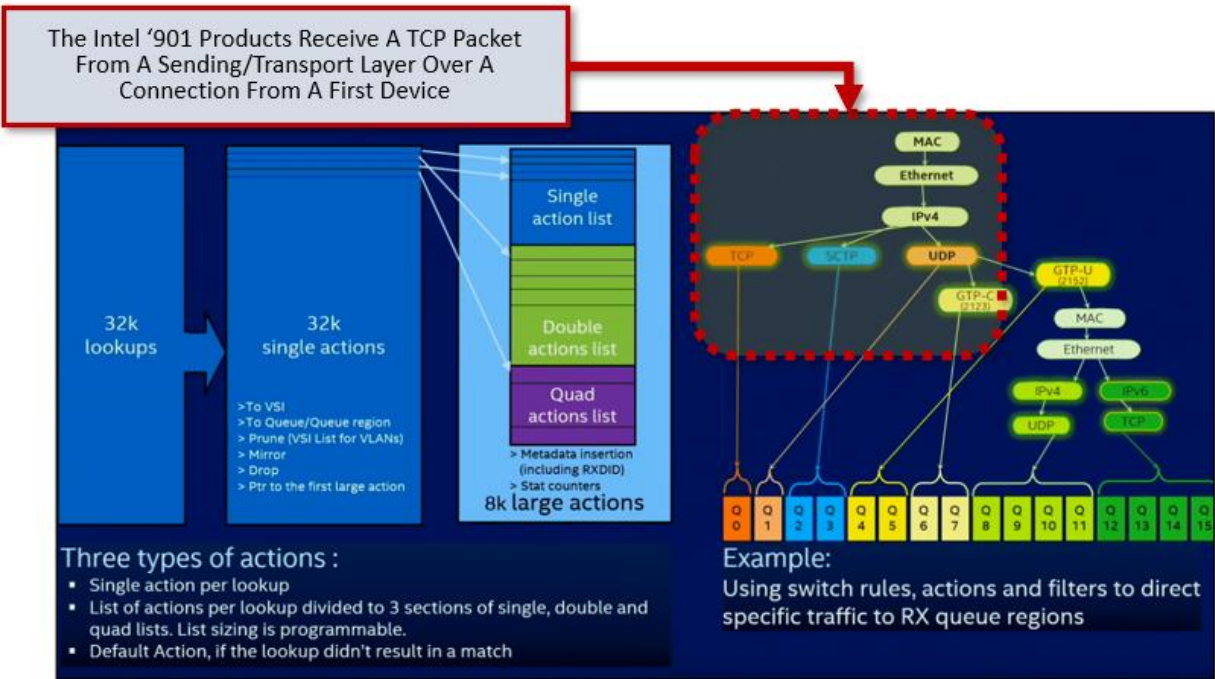
Once the Tx-Scheduler selects a transmit queue (either from LAN or from RDMA), it services some amount of transmit data (called a "quanta"). Packet data is read from host memory and any packet processing or packet manipulation takes place. These can include partial parsing of the packet, performing Transmit Segmentation Offload (TSO), calculating relevant checksums and CRCs, L2 header manipulations UP enforcement, and inner protocol TTL decrement.

The packet is then passed to the Packet Processing Pipeline, where it queries the on-die switch to identify its destinations. A packet can be forwarded locally to a Host destination (for example VM-VM communication) or to an on-board BMC. The packet also goes through the ACL block, where permissions are checked, and might alter the forwarding decisions made in the switch.

The LAN Controller then forwards the packet to one of the LAN ports and/or to the Receive path. If the destination port is busy, the packet is stored in a local Transmit Buffer. Transmission to a LAN port is done via a 802.3 MAC that interfaces an On-Chip Network Packet Interface (ONPI).

INTEL ETHERNET CONTROLLER E810 DATASHEET REVISION 2.8 at § 1.3.1 (September 2024) (emphasis added).

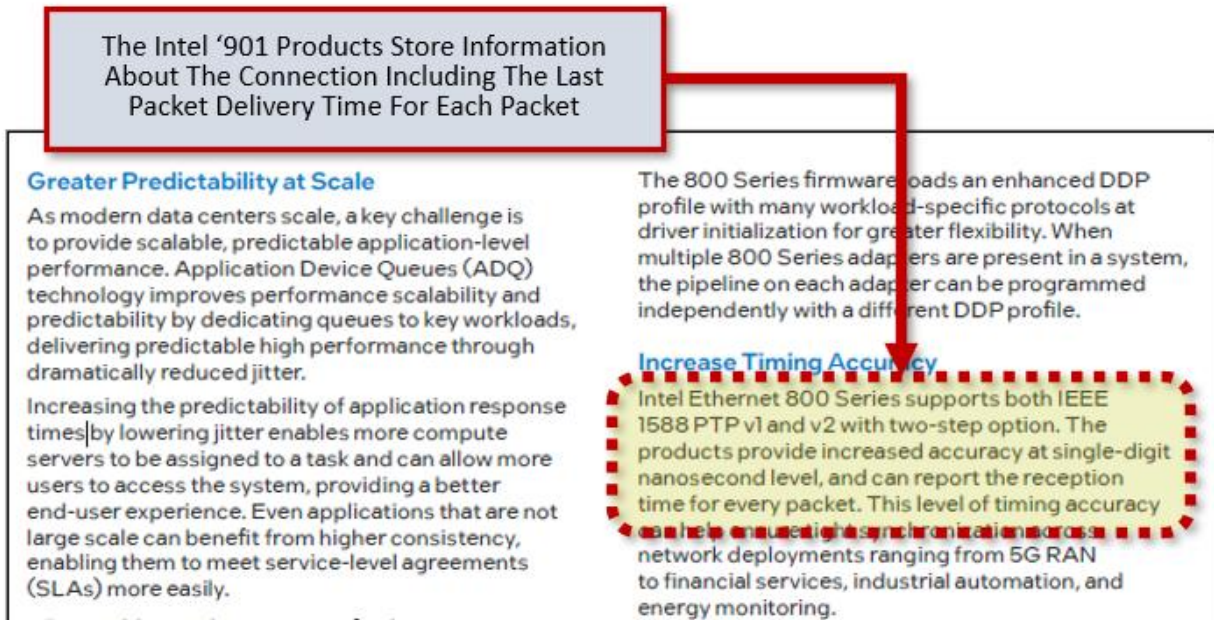
104. The Intel '901 Products receive a transmission control protocol (TCP) packet from a sending layer on the first device. The sending layer is one of the network interface layer or the transport layer and the TCP packet is sent over a connection between the first device and a second device. The receipt of TCP packets is shown in the following excerpt from an Intel presentation.



Brian Johnson and Jesse Brandeburg, *Intel Ethernet 800 Series Technical Feature Discussion*, TECH FIELD DAY YOUTUBE CHANNEL at 4:38 (October 2, 2019), available at: <https://www.youtube.com/watch?v=T38CWUWaeto> (annotation added).

105. Intel ‘901 Products contain functionality for receiving and sending TCP packets and comprise functionality for optimizing the flow of data between devices over various network paths.

106. Intel ‘901 Products store information about the connection between a first device and the second device. The information stored by the Intel ‘901 Products include a last packet delivery time for the connection as shown in the below excerpt from Intel documentation of the Intel ‘901 Products.



Intel Ethernet Network Adapter E810-XXVDA4T Data Sheet, INTEL DOCUMENTATION at 5 (2023) (annotation added).

107. Intel '901 Products determine if a TCP packet is part of a bursty transmission on the connection by looking at whether a burst count for the connection is greater than a burst-count threshold. The Intel '901 Products determine if a TCP packet is part of a bursty transmission by ascertaining that a burst count of the connection is greater than a burst-count threshold as shown in the following excerpt from Intel documentation.

Name	Byte.Bit	Value	Remarks
Tx-Scheduler Layer	0	Valid values = 1-9	Logical Layer number visible to software (Max = 9). <i>Note:</i> In case of RLs aimed to Queues, the layer is the max logical layer independently of where the queue was attached.
Flags	1	Bits 1.0-1.1: Profile Type 00b = CIR 01b = EIR 10b = Shared 11b = Reserved Bits 1.2-1.7: Reserved (0)	Type of RL profile to be added.
Profile ID	2-3		Command: Reserved as part of command (0). Response: The profile index value returned by firmware.
Max Burst Size	4-5	Max valid value is 0xFFFF (12 bits in hardware).	Max burst size.
RL Multiply	6-7	Max valid value is 0x7FF (11 bits in hardware).	Reserved when adding shared RL.

Intel Ethernet Controller E810 Version 2.7, INTEL DOCUMENTATION NEX CLOUD NETWORKING GROUP at 1341 (March 2024) (emphasis added).

108. Intel ‘901 Products calculate a delay time for a connection using the last packet delivery time after determining that the TCP packet is part of a bursty transmission. This measurement is then used to determine the burstiness of a TCP packet transmission.

Each LAN Tx-Queue and RDMA QSet is associated with a single TC inside the port space and mapped into a single CGD. To handle sudden PFC XOFF notifications received from the link, the transmit data path is provided with buffers in its different stages. To reduce impact on-die size, provision is made for only two types of traffic: High-Priority (also called Low-Latency (LL) in this Document) and Bulk (B). Each CGD in Tx is configured with four main behavioral parameters:

- **Advanced Mode Host Interface** — The Advanced Transmit mode enables software to control tightly the burstiness of the traffic from this queue, to interleave transmissions between multiple data sources while limiting the burst size from any one given queue. When a CGS is recognized as Burst Sensitive, all the Tx-Queues associated with this CGD must be configured to operate in Advanced Transmit Mode.
- **Bulk vs. Low Latency** — Each CGD is configured as LL (Low Latency) or B (Bulk) CGD. High-priority CGDs are selected first in the Tx arbiters, allowing them running faster through the Tx-Pipe.
- **Congested vs. Not Congested CGD** — This defines if the specific CGD is declared as Droppable in the network, and therefore no Priority Flow Control (PFC) is expected for the CGD or PFC support for it is required.

Intel Ethernet Controller E810 Version 2.7, INTEL DOCUMENTATION NEX CLOUD NETWORKING GROUP at 1265 (March 2024) (emphasis added).

109. The Intel ‘901 Products contain functionality for delivering the TCP packet to a receiving layer based on the calculated delay time, wherein the receiving layer is either the network interface layer or the transport layer that is not the sending layer. Specifically, the Intel ‘901 Products manage packet transmission times and delays as part of the Intel ‘901 Products’ traffic optimization and prioritization functionality.

110. The Intel ‘901 Products enable sending the TCP packet to the receiving layer.

111. The Intel ‘901 Products perform the steps at a packet scheduler layer located between a network interface layer and a transport layer. Specifically, the Intel ‘901 Products use a Transmit Scheduler that operates at the packet scheduling layer. The packet scheduling layer in the Intel ‘901 Products sits between the network layer (the Ethernet MAC/PHY) and the transport interface layer (which includes protocols such as TCP). The Transmit Scheduler manages transmission requests across multiple Tx-Queues. Intel’s documentation for the Intel ‘901 Products confirm the Transmit Scheduler supports up to 16K Tx-Queues. The Intel ‘901 Products Transmit Scheduler manages the “transmission of all Tx-Queues” and “decides which [] Tx-Queue is allowed to transmit and how many bytes.”

usage type called a user priority. The Tx-Scheduler is responsible for controlling the transmission of all Tx-Queues. It executes a scheduling algorithm and decides which the Tx-Queue is allowed to transmit and how many bytes. To meet all QoS agreements while considering the network capacity, and to reduce packet drop or flow control events, the Tx-Scheduler takes into account the following configuration parameters:

- Priority.
- Bandwidth allocation, bandwidth limit, minimum guaranteed bandwidth.
- Network topology, the network infrastructure fed by device ports, its capacity, and the deployment of the Rx units (switches, routers, target NICs, and so on) in the network.
- Those Rx units might be buffer-limited or limited by packet processing power. To address both types, the E810 can configure each topology node to arbitrate and shape for bytes per second (BPS) or packets per second (PPS).
- Quality of Service (QoS) and service level agreement (SLA).

INTEL ETHERNET CONTROLLER E810 DATASHEET REVISION 2.8 at § 8.3.1 (September 2024) (emphasis added).

112. Intel has directly infringed and continues to directly infringe the '901 Patent by, among other things, making, using, offering for sale, and/or selling technology for a data packet scheduler that reduces packet bursts, including but not limited to the Intel '901 Products.

113. The Intel '901 Products are available to businesses and individuals throughout the United States.

114. The Intel '901 Products are provided to businesses and individuals located in this District.

115. By making, using, testing, offering for sale, and/or selling products and services comprising technology for a data packet scheduler that reduced packet bursts, including but not limited to the Intel '901 Products, Intel has injured Plaintiff and is liable to Plaintiff for directly infringing one or more claims of the '901 Patent, including at least claim 1 pursuant to 35 U.S.C. § 271(a).

116. Intel has had knowledge of the '901 Patent and its infringement of the '901 Patent since at least service of the Original Complaint in this action or shortly thereafter. Moreover, the '901 Patent is well-known within the industry as demonstrated by multiple citations to the '901 Patent in published patents and patent applications assigned to technology companies and academic institutions. Intel has continued to infringe the '901 Patent despite knowing of the '901 Patent and its infringement thereof. Intel is infringing the '901 Patent in a manner best described as willful, wanton, malicious, in bad faith, deliberate, consciously wrongful, flagrant, or characteristic of a pirate.

117. To the extent applicable, the requirements of 35 U.S.C. § 287(a) have been met with respect to the '901 Patent.

118. As a result of Intel's infringement of the '901 Patent, Plaintiff has suffered monetary damages, and seeks recovery in an amount adequate to compensate for Intel's infringement, but in no event less than a reasonable royalty for the use made of the invention by Intel together with interest and costs as fixed by the Court.

COUNT III
INFRINGEMENT OF U.S. PATENT NO. 10,412,388

119. Plaintiff references and incorporates by reference the preceding paragraphs of this Complaint as if fully set forth herein.

120. Intel designs, makes, uses, sells, and/or offers for sale in the United States products comprising technology for video compression using adaptive re-quantization using extracted and derived quantization parameters.

121. Intel designs, makes, sells, offers to sell, imports, and/or uses Intel products that perform encoding of media data in compliance with the H.265 High Efficiency Video Coding (HEVC) compression standard, including but not limited to the products identified in the list attached hereto as Exhibit 7 (collectively, the "Intel '388 Product(s)").

122. Intel designs, makes, sells, offers to sell, imports, and/or uses Intel '388 products that comply with the H.265 video encoding standard.

123. The Intel '388 Products perform video processing compliant with the High Efficiency Video Coding (HEVC) standard, which is also often referred to as the H.265 standard. Specifically, the Intel '388 Products perform HEVC encoding.

Intel® Quick Sync Video uses the dedicated media processing capabilities of Intel® Graphics Technology to decode and encode fast. This enables the processor to complete other tasks at the same time and improves system responsiveness.

The High Efficiency Video Coding (HEVC), also known as H.265 and MPEG-H Part 2, is a video compression standard designed as part of the MPEG-H project as a successor to the widely used Advanced Video Coding (AVC, H.264, or MPEG-4 Part 10).

Resolution

Hardware accelerated support for the H.265/HEVC codec starts with 6th generation Intel® Core™ processors.

[This site](#) contains some more detailed information about the encoding/decoding capabilities of 7th generation and newer Intel® Core™ processors.

H.265/HEVC Hardware Encoding and Decoding Support, INTEL PRODUCT SUPPORT WEBSITE (last visited November 2024), *available at*: <https://www.intel.com/content/www/us/en/support/articles/000037112/graphics.html> (emphasis added).

124. One or more Intel subsidiaries and/or affiliates use the Intel ‘388 Products in regular business operations.

125. The Intel ‘388 Products identify an initial quantization parameter employed to compress a previously decoded frame.

126. The Intel ‘388 Products, as part of the encoding process use an initial quantization parameter (QP) for encoding each frame or coding unit (CU). In conforming to the HEVC standard, the Intel ‘388 Products must set an initial QP value that serves as the baseline for encoding the decoded frame.

127. The Intel ‘388 Products calculate a delta quantization parameter as influenced by the initial quantization parameter, where the function is designed to yield this delta parameter at least in part to achieve a bitrate reduction while sustaining a given quality threshold.

128. The Intel ‘388 Products calculate a delta QP based on the initial quantization parameter. This function aims to minimize bitrate while retaining the required video quality.

129. The Intel '388 Products ascertain a subsequent quantization parameter for the purpose of compressing the decoded frame, based on both the initial and delta quantization parameters.

130. The Intel '388 Products determine a second quantization parameter using the initial QP and the delta QP. The Intel '388 Products calculate the second quantization parameter as $QP1 + \text{Delta QP}$. This second quantization parameter is the one used for encoding either the entire frame or specific coding units within the frame.

131. The Intel '388 Products compress the decoded frame utilizing the second quantization parameter.

132. The Intel '388 Products encode the video frames using the newly derived second quantization parameter.

133. By complying with the HEVC standard, the Intel '388 Products necessarily infringe the '388 Patent. Mandatory sections of the HEVC standard require the elements required by certain claims of the '388 Patent, including but not limited to claim 1. High Efficiency Video Coding, Series H: Audiovisual And Multimedia Systems: Infrastructure Of Audiovisual Services – Coding Of Moving Video Rec. ITU-T H.265 (August 2021). The following sections of the HEVC Standard are relevant to Intel's infringement of the '388 Patent: "7.3.2.2.3 Sequence parameter set screen content coding extension syntax;" "7.3.8.4 Coding quadtree syntax;" "7.3.8.14 Delta QP syntax;" "7.4.3.3.1 General picture parameter set RBSP semantics;" "7.4.7.1 General slice segment header semantics;" "7.4.9.14 Delta QP semantics;" "8.6.1 Derivation process for quantization parameters;" and "9.3.3.10 Binarization process for $cu_qp_delta_abs$."

134. All implementations of the HEVC standard necessarily infringe the '388 Patent as every implementation of the standard requires compliant devices to carry out the following: Each

frame or coding unit (CU) is encoded using a pre-defined initial Quantization Parameter (QP) which serves as a baseline for various optimizations. The standard mandates that a first QP (QP1) be identified before any encoding can occur. The Intel '388 Products are, therefore, required to have mechanisms to set this initial QP1 for the to-be-encoded (or re-encoded) frame. Further, the HEVC standard sets out a structured way to adjust this initial QP based on a delta value. The objective of introducing a delta QP is generally to adapt to the complexity variations within a video sequence and to optimize rate-distortion performance. The HEVC encoding standard sets forth calculating a new QP (QP2) after determining the delta QP. This is done by adding the initial QP (QP1) and the delta QP. This step is essential for maintaining granular control over the rate-distortion tradeoff during encoding. Finally, the final encoding of the frame or CU takes place using QP2. The HEVC standard specifies that this is a requisite step for the encoding process to be considered compliant. The Intel '388 Products must, therefore, encode frames using this newly computed QP2 to meet the standard's rate and quality stipulations.

135. Intel has directly infringed and continues to directly infringe the '388 Patent by, among other things, making, using, offering for sale, and/or selling technology for video compression using adaptive re-quantization using extracted and derived quantization parameters, including but not limited to the Intel '388 Products.

136. The Intel '388 Products are available to businesses and individuals throughout the United States.

137. The Intel '388 Products are provided to businesses and individuals located in this District.

138. By making, using, testing, offering for sale, and/or selling products and services comprising technology for video compression using adaptive re-quantization using extracted and

derived quantization parameters, including but not limited to the Intel '388 Products, Intel has injured Plaintiff and is liable to Plaintiff for directly infringing one or more claims of the '388 Patent, including at least claim 1 pursuant to 35 U.S.C. § 271(a).

139. Intel has had knowledge of the '388 Patent and its infringement of the '388 Patent since at least service of the Original Complaint in this action or shortly thereafter. Moreover, the '388 Patent is well-known within the industry as demonstrated by multiple citations to the '388 Patent in published patents and patent applications assigned to technology companies and academic institutions. Intel has continued to infringe the '388 Patent despite knowing of the '388 Patent and its infringement thereof. Intel is infringing the '388 Patent in a manner best described as willful, wanton, malicious, in bad faith, deliberate, consciously wrongful, flagrant, or characteristic of a pirate.

140. To the extent applicable, the requirements of 35 U.S.C. § 287(a) have been met with respect to the '388 Patent.

141. As a result of Intel's infringement of the '388 Patent, Plaintiff has suffered monetary damages, and seeks recovery in an amount adequate to compensate for Intel's infringement, but in no event less than a reasonable royalty for the use made of the invention by Intel together with interest and costs as fixed by the Court.

COUNT IV
INFRINGEMENT OF U.S. PATENT NO. 9,894,361

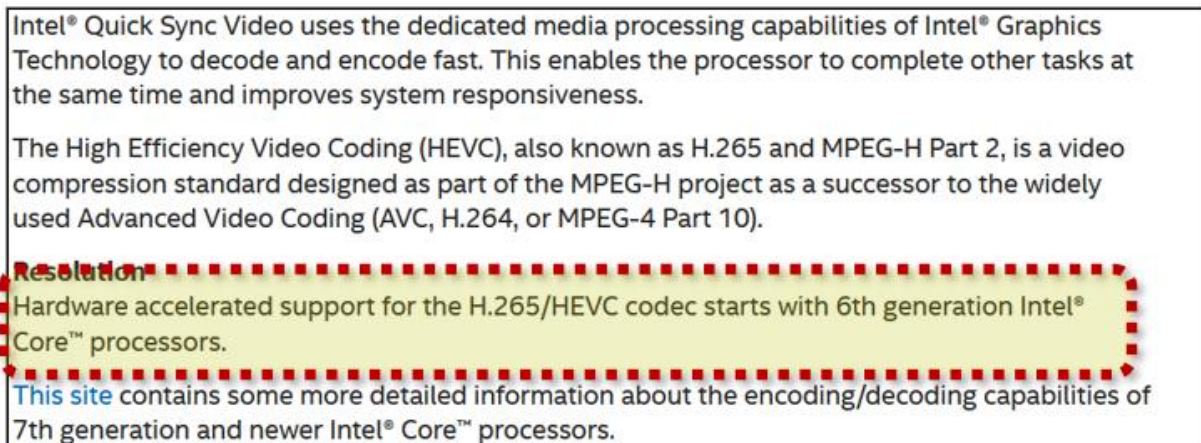
142. Plaintiff references and incorporates by reference the preceding paragraphs of this Complaint as if fully set forth herein.

143. Intel designs, makes, uses, sells, and/or offers for sale in the United States products containing technology for quality-aware video optimization.

144. Intel designs, makes, sells, offers to sell, imports, and/or uses the following products: Intel designs, makes, sells, offers to sell, imports, and/or uses Intel products that perform encoding of media data in compliance with the H.265 High Efficiency Video Coding (HEVC) compression standard, including but not limited to the products identified in the list attached hereto as Exhibit 7 (collectively, the “Intel ‘361 Product(s)”).

145. Intel designs, makes, sells, offers to sell, imports, and/or uses Intel ‘361 Products that comply with the H.265 video encoding standard.

146. The Intel ‘361 Products perform video processing compliant with the High Efficiency Video Coding (HEVC) standard, which is also often referred to as the H.265 standard. Specifically, the Intel ‘361 products perform HEVC encoding.



H.265/HEVC Hardware Encoding and Decoding Support, INTEL PRODUCT SUPPORT WEBSITE (last visited November 2024), available at: <https://www.intel.com/content/www/us/en/support/articles/000037112/graphics.html> (emphasis added).

147. One or more Intel subsidiaries and/or affiliates use the Intel ‘361 Products in regular business operations.

148. The Intel ‘361 Products unpack a compressed video frame from a series containing multiple video frames.

149. The Intel '361 Products take an encoded video frame as input. This frame is one in a series that consists of multiple frames. The encoded frame is then passed through a decoding pipeline by the Intel '361 Products. The Intel '361 Products use inverse quantization and inverse DCT (Discrete Cosine Transform) functions, to revert the video data to a decompressed state suitable for further manipulation.

150. The Intel '361 Products obtain an initial Quantization Parameter (QP) from the unpacked video frame, where this initial QP is indicative of the quantization configurations initially applied to compress the video frame.

151. The Intel '361 Products extract a first Quantization Parameter (QP) from the video frame metadata or from the bitstream itself. This first QP reflects the quantization settings initially applied during the original encoding. This first QP is read from the slice header or similar control structures and used to modulate the quantization matrices in the decoding process.

152. The Intel '361 Products calculate a delta QP influenced by the initial QP.

153. Upon acquiring the first QP, a delta QP is calculated by the Intel '361 Products. This delta QP value is computed through a set of heuristic functions to optimize for certain objectives like bitrate reduction, video quality, or computational efficiency. The delta QP acquired by the Intel '361 Products is a function of the first QP and other parameters, such as frame type (I-frame, P-frame, etc.).

154. The Intel '361 Products derive an inflation factor through comparing the total byte size of video frames after and before decompression, where both the newly received compressed frame and those previously decompressed belong to the same series of multiple video frames.

155. The Intel '361 Products compute an inflation adjustment factor based on the total byte size of previously decompressed frames and those frames post-compression. This comparison aids in estimating the compression efficiency.

156. The Intel '361 Products acquire a subsequent QP influenced by both the delta QP and the inflation factor, wherein this subsequent QP is indicative of the quantization configurations to be applied for recompressing the unpacked frame.

157. The second QP is then acquired by the Intel '361 Products by combining the calculated delta QP and the inflation adjustment. This second quantization parameter acquired by the Intel '361 Products aims to balance the trade-offs between quality and bitrate, taking into account the information gleaned from previous frames as indicated by the inflation adjustment.

158. The Intel '361 Products compress the unpacked video frame utilizing the subsequent QP.

159. The decompressed video frame is re-encoded based on the second QP by the Intel '361 Products. The frame is then serialized into a bitstream and packaged with appropriate headers and metadata for transmission or storage.

160. Intel has directly infringed and continues to directly infringe the '361 Patent by, among other things, making, using, offering for sale, and/or selling technology for quality-aware video optimization, including but not limited to the Intel '361 Products.

161. The Intel '361 Products are available to businesses and individuals throughout the United States.

162. The Intel '361 Products are provided to businesses and individuals located in this District.

163. By making, using, testing, offering for sale, and/or selling products and services comprising technology for quality-aware video optimization, including but not limited to the Intel ‘361 Products, Intel has injured Plaintiff and is liable to Plaintiff for directly infringing one or more claims of the ‘361 Patent, including at least claim 10 pursuant to 35 U.S.C. § 271(a).

164. Intel has had knowledge of the ‘361 Patent and its infringement of the ‘361 Patent since at least service of the Original Complaint in this action or shortly thereafter. Moreover, the ‘361 Patent is well-known within the industry as demonstrated by multiple citations to the ‘361 Patent in published patents and patent applications assigned to technology companies and academic institutions. Intel has continued to infringe the ‘361 Patent despite knowing of the ‘361 Patent and its infringement thereof. Intel is infringing the ‘361 Patent in a manner best described as willful, wanton, malicious, in bad faith, deliberate, consciously wrongful, flagrant, or characteristic of a pirate.

165. To the extent applicable, the requirements of 35 U.S.C. § 287(a) have been met with respect to the ‘361 Patent.

166. As a result of Intel’s infringement of the ‘361 Patent, Plaintiff has suffered monetary damages, and seeks recovery in an amount adequate to compensate for Intel’s infringement, but in no event less than a reasonable royalty for the use made of the invention by Intel together with interest and costs as fixed by the Court.

COUNT V
INFRINGEMENT OF U.S. PATENT NO. 10,123,015

167. Plaintiff references and incorporates by reference the preceding paragraphs of this Complaint as if fully set forth herein.

168. Intel designs, makes, uses, sells, and/or offers for sale in the United States products comprising technology for optimizing encoded video streams by tailoring quality settings for macroblocks.

169. Intel designs, makes, sells, offers to sell, imports, and/or uses the following products: the Intel Iris X^e; Intel Iris X^e; MAX, Intel Arc (including models: A310, A380, A580, A750, A770 8GB, A770 16GB, A350M, A370M, A530M, A550M, A570M, A730M, A770M, A30M, A40, A60M, A60); Intel Data Center GPU Max 1100; Intel Data Center GPU Flex 140; Intel Data Center GPU Flex 170; and Intel Data Center GPU Flex 170v (collectively, the “Intel ‘015 Product(s)”).

170. One or more Intel subsidiaries and/or affiliates use the Intel ‘015 Products in regular business operations.

171. The Intel ‘015 Products optimize encoded video streams comprised of video frames. Each video frame is comprised of a plurality of macroblocks.

172. The Intel ‘015 Products receive information for a macroblock of a video frame of the encoded video stream. Specifically, the Intel ‘015 Products receive information for macroblocks of a video frame. The Intel ‘015 Products process macroblock-level information during encoding including motion vectors and quantization parameters. According to the below excerpt from Intel documentation the Intel ‘015 Products receive macroblock-level information. For example, the Macroblock Quantization Parameter Control (EnableMBQP) in mfxstructures.h allows the Intel ‘015 Products to define and receive information about the quantization parameter (QP) for each macroblock. The QP is an encoding parameter that directly controls compression and quality at the macroblock level. In addition, the extended buffer (MFX_EXTBUFF_MBQP) in the Intel ‘015 Products is specifically designed for macroblock level data in encoding and this

received macroblock information is used by the `mfxEncodeCtrl` structure to configure macroblock-level parameters during per-frame encoding.

The Intel '015 Products Receive Information For A Macroblock Of A Video Frame Of The Encoded Video Stream. The Encoder Receives Input Frame Data Through: `EnableMBQP` and `MFX_EXTBUFF_MBQP`.

```

    /*! When rate control method is MFX_RATECONTROL_QVBR, this parameter specifies quality factor.
        Values are in the 1 to 51 range, where 1 corresponds to the best quality.
    */
    mfxU16    QVBRQuality;
    /*!
        Set this flag to ON to enable per-macroblock QP control. Rate control method must be MFX_RATECONTROL_CQP.
        enumerator for values of this option. This parameter is valid only during initialization.
    */
    mfxU16    EnableMBQP;
    /*!
  
```

```

    This extended buffer defines per-macroblock QP. See the mfxExtMBQP structure for details.
    The application can attach this buffer to the mfxEncodeCtrl structure for per-frame encoding configuration.
    /*!
    MFX_EXTBUFF_MBQP                = MFX_MAKEFOURCC('M','B','Q','P'),
    /*!
    This extended buffer defines per-macroblock force intra flag. See the mfxExtMBForceIntra structure for details.
    The application can attach this buffer to the mfxEncodeCtrl structure for per-frame encoding configuration.
    /*!
    MFX_EXTBUFF_MB_FORCE_INTRA      = MFX_MAKEFOURCC('M','B','F','I'),
    /*!
  
```

`mfxstructures.h`, INTEL VIDEO PROCESSING LIBRARY (INTEL VPL) RELEASE 2.13.0 (August 30, 2024) (annotation added).

173. The Intel '015 Products extract a first quantization parameter (QP) corresponding to the quantization settings originally used for compressing the macroblock. Specifically, Intel '015 Products perform constant quantization parameter (QP) mode (CQP), as shown in the below excerpt from the Intel '015 Product. In CQP mode, the Intel '015 Products assign quantization parameters for I-, P-, and B-frames using fields such as QPI, QPP, and QPB. These parameters are adjusted by the Intel '015 Products based on bit depth and codec requirements.

The Intel '015 Products Extract Quantization Parameters Originally Used For Compressing The Macroblock. The Intel '015 Products Perform Constant Quantization Parameter (CQP) Mode.

```

    /*! Quantization Parameter (QP) for I-frames for constant QP mode (CQP). Zero QP is not valid and means that the default value
    Non-zero QP1 might be clipped to supported QP1 range.
    @note In the HEVC design, a further adjustment to QPs can occur based on bit depth.
    Adjusted QP1 value = QP1 - (6 * (BitDepthLuma - 8)) for BitDepthLuma in the range [8,14].
    For HEVC_MAIN10, we minus (6*(10-8)=12) on our side and continue.
    @note In av1 design, valid range is 0 to 255 inclusive, and if QP1=QPP=QPB=0, the encoder is in lossless mode.
    @note In vp9 design, valid range is 1 to 255 inclusive, and zero QP that the default value is assigned by the library.
    @note Default QP1 value is implementation dependent and subject to change without additional notice in this document. */
    mfxU16 QP1;
    mfxU16 Accuracy; /*!< Specifies accuracy range in the unit of tenth of percent. */
};

```

mfxstructures.h, INTEL VIDEO PROCESSING LIBRARY (INTEL VPL) RELEASE 2.13.0 (August 30, 2024) (annotation added).

174. The Intel '015 Products perform per-macroblock quantization control through the *mfxExtMBQP* component, allowing the Intel '015 Product to extract a first quantization parameter (and handle both absolute QP values (QP) and delta QP values (DeltaQP)). Further, the Intel '015 Products perform iterative quantization adjustments during encoding, as shown in the below excerpt from the Intel '015 Products, where they use frame-level Luma QP (QpY) and delta QP arrays (DeltaQP) to fine-tune compression and optimize encoding. These features demonstrate the Intel '015 Products extract and utilize quantization parameters at the macroblock level to facilitate encoding.

```

mfxU8 DeltaQP[8]; /*!< Option for repack feature. Ignored if MaxNumRePak == 0 or MaxNumRePak==0. If current
frame size > MaxFrameSize and/or number of repacking (nRepack) for this frame <= MaxNumRePak,
PAK is called with QP = mfxBRCFrameCtrl::QpY + Sum(DeltaQP[i]), where i = [0,nRepack].
Non zero DeltaQP[nRepack] are ignored if nRepack > MaxNumRePak.
If repacking feature is on ( MaxFrameSize & MaxNumRePak are not zero), it is calculated by the
mfxU16 MaxNumRepak; /*!< Number of possible repacks in driver if current frame size > MaxFrameSize. Ignored if MaxFrame
See MaxFrameSize description. Possible values are in the range of 0 to 8. */
mfxU16 NumExtParam; /*!< Reserved for future use. */
mfxExtBuffer** ExtParam; /*!< Reserved for future use. */

```

mfxbrc.h, Intel Video Processing Library (Intel VPL) Release 2.13.0 (August 30, 2024) (emphasis added).

175. The Intel '015 Products determine, using the received information, motion vector information for at least one motion vector associated with the macroblock that indicates a location of a prediction block within the video frame or another video frame which was subtracted from the macroblock prior to encoding. Specifically, the Intel '015 Products extract the motion vector associated with the macroblock, which identifies the location of at least one prediction block within the video frame or another video frame, and subtract this prediction block from the macroblock prior to encoding. The Intel '015 Products use structures such as mfxCUInfo to provide detailed motion vector parameters, including ranges for motion vectors (MV) and reference indices (RefID) that link motion vectors to specific reference frames. These features enable the Intel '015 Products to associate motion vectors with prediction directions (e.g., forward, backward, or bidirectional) and partition levels within a macroblock. Additionally, the Intel '015 Products leverage the mfxCTUInfo structure to process Coding Tree Units (CTUs), specifying prediction modes (CU_pred_mode) and inter-prediction details, allowing the Intel '015 Products to determine motion vectors for a macroblock.

The Intel '015 Products Determine Motion Vector Information For At Least One Motion Vector Associated With A Macroblock Using A Reference Index - *RefID*

```

/!*
This parameter indicates the reference index associated with the MV X/Y
that is populated in the L0_MV0.X and L0_MV0.Y fields. */
    mfxU32 L0_MV0_RefID    : 4;
/!*
This parameter indicates the reference index associated with the MV X/Y
that is populated in the L0_MV1.X and L0_MV1.Y fields. */
    mfxU32 L0_MV1_RefID    : 4;
/!*
This parameter indicates the reference index associated with the MV X/Y
that is populated in the L1_MV0.X and L1_MV0.Y fields. */
    mfxU32 L1_MV0_RefID    : 4;
/!*
This parameter indicates the reference index associated with the MV X/Y
that is populated in the L1_MV1.X and L1_MV1.Y fields. */
    mfxU32 L1_MV1_RefID    : 4;

    mfxU32 reserved3      : 16;
} bitfields8;
mfxU32 dword8;
};

```

mfxencodestats.h, INTEL VIDEO PROCESSING LIBRARY (INTEL VPL) RELEASE 2.13.0 (August 30, 2024) (emphasis added).

176. The Intel '015 Products compute a second quantization parameter (QP) for re-encoding the macroblock. The second QP is based at least in part on the first QP and the motion vector information. Specifically, the Intel '015 Products adjust quantization parameters during re-encoding using the *mfxBRCFrameCtrl* structure, which applies incremental changes through the *DeltaQP* array. This process ensures that re-encoding meets compression criteria, with the quantization parameter computed as $QpY + \text{Sum}(\text{DeltaQP}[i])$ across iterations. The Intel '015

Products also use motion vector information from `mfxCUInfo` and `mfxCTUInfo`, which provide details on inter-prediction modes, partitioning, and reference indices for prediction blocks. Additionally, the `mfxBRCFrameParam` structure in the Intel '015 Products uses parameters such as frame complexity (`FrameCmplx`) and scene changes (`SceneChange`) to determine and apply the second quantization parameter for macroblock re-encoding.

```

mfxI32 QpY; /*< Frame-level Luma QP. */
mfxU32 InitialCpbRemovalDelay; /*< See initial_cpb_removal_delay in codec standard. Ignored if no HRD control:
    mfxExtCodingOption::VuiNalHrdParameters = MFX_CODINGOPTION_OFF. Calculated by encoder if
    initial_cpb_removal_delay==0 && initial_cpb_removal_offset == 0 && HRD control is switched on. */
mfxU32 InitialCpbRemovalOffset; /*< See initial_cpb_removal_offset in codec standard. Ignored if no HRD control:
    mfxExtCodingOption::VuiNalHrdParameters = MFX_CODINGOPTION_OFF. Calculated by encoder if
    initial_cpb_removal_offset==0 && initial_cpb_removal_offset == 0 && HRD control is switched on. */
mfxU8 DeltaQP[8];
mfxU16 MaxFrameSize; /*< Option for repack feature. Driver calls PAK until current frame size is
    less than or equal to MaxFrameSize, or number of repacking for this frame is equal to MaxNumRePak. Repack
    if there is driver support, MaxFrameSize !=0, and MaxNumRePak != 0. Ignored if MaxNumRePak == 0. */
mfxU16 MaxNumRePak; /*< Option for repack feature. Ignored if MaxNumRePak == 0 or MaxNumRePak==0. If current
    frame size > MaxFrameSize and/or number of repacking (nRepack) for this frame <= MaxNumRePak,
    PAK is called with QP = mfxBRCFrameCtrl::QpY + Sum(DeltaQP[i]), where i = [0,nRepack].
    Non zero DeltaQP[nRepack] are ignored if nRepack > MaxNumRePak.
    on ( MaxFrameSize & MaxNumRePak are not zero), it is calculated by the encoder. */
} mfxBRCFrameCtrl;
mfxU16 NumRepack; /*< Number of repacking in driver if current frame size > MaxFrameSize. Ignored if MaxFrameSize==0.
    tion. Possible values are in the range of 0 to 8. */
mfxExtBufParam ExtParam; /*< Reserved for future use. */
} mfxBRCFrameCtrl;

```

mfxbrc.h, INTEL VIDEO PROCESSING LIBRARY (INTEL VPL) RELEASE 2.13.0 (August 30, 2024) (emphasis added).

177. The Intel '015 Products re-encode the macroblock based on the second QP. For example, the Intel '015 Products perform re-encoding of a macroblock using a second quantization parameter through the `mfxBRCFrameCtrl` structure for frame-level adjustments. The `mfxBRCFrameCtrl` structure in the Intel '015 Products applies frame-level quantization adjustments during re-encoding. Specifically, the field `QpY` represents the base quantization parameter for Luma, which is adjusted using `DeltaQP` values during re-encoding. In addition, the Intel '015 Products use the `MaxFrameSize` and `MaxNumRepak` fields in `mfxBRCFrameCtrl` to perform iterative encoding passes. During these passes, the Intel '015 Product dynamically adjusts the quantization parameter ($QpY + \text{DeltaQP}[i]$) to meet frame size or quality criteria. This iterative process involves re-encoding macroblocks until the desired criteria are satisfied.

```

Option for repack feature. Ignored if MaxNumRePak == 0 or MaxNumRePak==0. If current
frame size > MaxFrameSize and/or number of repacking (nRepack) for this frame <= MaxNumRePak,
PAK is called with QP = mfxBRCFrameCtrl::QpY + Sum(DeltaQP[i]), where i = [0,nRepack].
Non zero DeltaQP[nRepack] are ignored if nRepack > MaxNumRePak.
If repacking feature is on ( MaxFrameSize & MaxNumRePak are not zero), it is calculated by the encoder.
Number of possible repacks in driver if current frame size > MaxFrameSize. Ignored if MaxFrameSize==0.
See MaxFrameSize description. Possible values are in the range of 0 to 8. */

```

mfxbrc.h, INTEL VIDEO PROCESSING LIBRARY (INTEL VPL) RELEASE 2.13.0 (August 30, 2024) (emphasis added).

178. The Intel '015 Products transmit the re-encoded macroblock to a user device. Specifically, the *mfxBitstream* structure holds the encoded video data, including re-encoded macroblocks. This structure includes pointers to buffer memory (*Data*) and size information (*DataLength*, *MaxLength*) and are used by the Intel '015 Products to transmit the re-encoded data to a user device.

```

mfxI64 DecodeTimeStamp;
mfxU64 TimeStamp;           /*!< Time stamp of the compressed bitstream in units of 90KHz. A value of
mfxU8* Data;                /*!< Bitstream buffer pointer, 32-bytes aligned. */
mfxU32 DataOffset;         /*!< Next reading or writing position in the bitstream buffer. */
mfxU32 DataLength;         /*!< Size of the actual bitstream data in bytes. */
mfxU32 MaxLength;          /*!< Allocated bitstream buffer size in bytes. */

mfxU16 PicStruct;           /*!< Type of the picture in the bitstream. Output parameter. */
mfxU16 FrameType;          /*!< Frame type of the picture in the bitstream. Output parameter. */
mfxU16 DataFlag;           /*!< Indicates additional bitstream properties. See the BitstreamDataFlag
mfxU16 reserved2;          /*!< Reserved for future use. */
} mfxBitstream;
MFX_PACK_END()

```

mfxstructures.h, Intel Video Processing Library (Intel VPL) Release 2.13.0 (August 30, 2024) (emphasis added).

179. Intel has directly infringed and continues to directly infringe the '015 Patent by, among other things, making, using, offering for sale, and/or selling technology for optimizing encoded video streams by tailoring quality settings for macroblocks, including but not limited to the Intel '015 Products.

180. The Intel '015 Products are available to businesses and individuals throughout the United States.

181. The Intel '015 Products are provided to businesses and individuals located in this District.

182. By making, using, testing, offering for sale, and/or selling products and services comprising technology for optimizing encoded video streams by tailoring quality settings for macroblocks, including but not limited to the Intel '015 Products, Intel has injured Plaintiff and is liable to Plaintiff for directly infringing one or more claims of the '015 Patent, including at least claim 1 pursuant to 35 U.S.C. § 271(a).

183. Intel has had knowledge of the '015 Patent and its infringement of the '015 Patent since at least service of the Original Complaint in this action or shortly thereafter. Moreover, the '015 Patent is well-known within the industry as demonstrated by multiple citations to the '015 Patent in published patents and patent applications assigned to technology companies and academic institutions. Intel has continued to infringe the '015 Patent despite knowing of the '015 Patent and its infringement thereof. Intel is infringing the '015 Patent in a manner best described as willful, wanton, malicious, in bad faith, deliberate, consciously wrongful, flagrant, or characteristic of a pirate.

184. To the extent applicable, the requirements of 35 U.S.C. § 287(a) have been met with respect to the '015 Patent.

185. As a result of Intel's infringement of the '015 Patent, Plaintiff has suffered monetary damages, and seeks recovery in an amount adequate to compensate for Intel's infringement, but in no event less than a reasonable royalty for the use made of the invention by Intel together with interest and costs as fixed by the Court.

COUNT VI
INFRINGEMENT OF U.S. PATENT NO. 9,621,896

186. Plaintiff references and incorporates by reference the preceding paragraphs of this Complaint as if fully set forth herein.

187. Intel designs, makes, uses, sells, and/or offers for sale in the United States products comprising technology for optimizing encoded video streams by tailoring quality settings for macroblocks.

188. Intel designs, makes, sells, offers to sell, imports, and/or uses the following products: the Intel Iris X^e; Intel Iris X^e; MAX, Intel Arc (including models: A310, A380, A580, A750, A770 8GB, A770 16GB, A350M, A370M, A530M, A550M, A570M, A730M, A770M, A30M, A40, A60M, A60); Intel Data Center GPU Max 1100; Intel Data Center GPU Flex 140; Intel Data Center GPU Flex 170; and Intel Data Center GPU Flex 170v (collectively, the “Intel ‘896 Product(s)”).

189. One or more Intel subsidiaries and/or affiliates use the Intel ‘896 Products in regular business operations.

190. The Intel ‘896 Products perform a method of macroblock-level quality-aware video optimization of an encoded video stream comprising one or more video frames, each video frame comprising a plurality of macroblocks, each macroblock comprising a plurality of pixels.

191. The Intel ‘896 Products receive an encoded macroblock. Specifically, the Intel ‘896 Products contain software for handling encoded macroblocks through the VDEnc (Video Decode Encode) and HEVC (High Efficiency Video Coding) components. For example, the CodechalVdencHevcState class implements functionality for processing encoded video frames, including encoded macroblocks.

```

137  ///
138  /// \class   CodechalVdencHevcState
139  /// \brief   HEVC VDEnc encoder base class
140  /// \details This class defines the base class for HEVC VDEnc encoder, it includes
141  ///          common member fields, functions, interfaces etc shared by all GENs.
142  ///          Gen specific definitions, features should be put into their corresponding classes.
143  ///
144  class CodechalVdencHevcState : public CodechalEncodeHevcBase
145  {
146  public:
147      ///
148      /// \struct   HevcVdencBrcBuffers
149      /// \brief   Hevc Vdenc brc buffers
150      ///
151  struct HevcVdencBrcBuffers
152  {

```

Intel Media Driver - codechal_vdenc_hevc.h, INTEL GITHUB SOURCE CODE REPOSITORY, available at: https://github.com/intel/media-driver/agnostic/common/codec/hal/codechal_vdenc_hevc.h (last visited February 2025) (emphasis added).

192. The Intel ‘896 Products decode the encoded macroblock. Specifically, the Intel ‘896 Products decode the encoded macroblock through various data structures like ‘CODECHAL_VDENC_STREAMIN_STATE’ which contains macroblock-level data. The function ‘SetupROIStreamIn’ processes incoming macroblock information.

```

678  //Set the ROI DeltaQp to zero for remaining array elements
679  for (auto k = numQp; k < m_maxNumROI; k++)
680  {
681      m_hevcPicParams->ROIDistinctDeltaQp[k] = 0;
682  }
683
684  m_vdencNativeROIEnabled = !(numQp > m_maxNumNativeROI || m_hevcPicParams->ROIDistinctDeltaQp[0] < -8 ||
685  }
686
687  MOS_STATUS CodechalVdencHevcState::SetupROIStreamIn(PMOS_RESOURCE streamIn)
688  {
689      MOS_STATUS eStatus = MOS_STATUS_SUCCESS;
690
691      CODECHAL_ENCODE_FUNCTION_ENTER;

```

Intel Media Driver - codechal_vdenc_hevc.cpp, INTEL MEDIA DRIVER GITHUB SOURCE CODE REPOSITORY, available at: https://github.com/intel/media-driver/agnostic/common/codec/hal/codechal_vdenc_hevc.cpp (last visited February 2025) (emphasis added).

193. The Intel ‘896 Products extract a first quantization parameter, wherein the first quantization parameter corresponds to quantization settings originally used for compressing the encoded macroblock. The Intel ‘896 Products contain functions for extracting a first quantization

parameter. For instance, in `encode_hevc_vdenc_scc.cpp` “`m_hevcPicParams->QpY`” is the base frame-level QP extracted from the encoded stream. The picture parameters (including QP values) are parsed from the bitstream header during the decoding process by the Intel ‘896 Products.

```

// Calculate ForceQp
int8_t forceQp = (int8_t)Codechal_Clip3(10, 51, m_hevcPicParams->QpY + m_hevcPicParams->ROI[i].PriorityLevelOrQp + m_hevcSliceParams->slice_qp_delta);

MOS_ZeroMemory(&streaminDataParams, sizeof(streaminDataParams));
streaminDataParams.setQpRoiCtrl = true;
if (m_vdencNativeROIEnabled)
{
    streaminDataParams.roiCtrl = (uint8_t)roiCtrl;
}
else
{
    streaminDataParams.forceQp[0] = forceQp;
    streaminDataParams.forceQp[1] = forceQp;
    streaminDataParams.forceQp[2] = forceQp;
    streaminDataParams.forceQp[3] = forceQp;
}

SetStreaminDataPerRegion(streamInWidth, top, bottom, left, right, &streaminDataParams, data);
}

```

Intel Media Driver - codechal_vdenc_hevc.cpp, INTEL MEDIA DRIVER GITHUB SOURCE CODE REPOSITORY, available at: https://github.com/intel/media-driver/agnostic/common/codec/hal/codechal_vdenc_hevc.cpp (last visited February 2025) (emphasis added).

194. The Intel ‘896 Products determine first and second thresholds as a function of past input quantization parameters, wherein the past input quantization parameters correspond to quantization settings originally used for compressing a plurality of previously received encoded macroblocks. Specifically, the Intel ‘896 Products maintain a “history buffer” to track QP data across multiple frames.

```

// BRC history buffer
allocParamsForBufferLinear.dwBytes = MOS_ALIGN_CEIL(m_brcHistoryBufSize, CODECHAL_PAGE_SIZE);
allocParamsForBufferLinear.pBufName = "VDENC BRC History Buffer";

CODECHAL_ENCODE_CHK_STATUS_MESSAGE_RETURN(m_osInterface->pfnAllocateResource(
    m_osInterface,
    &allocParamsForBufferLinear,
    &m_vdencBrcHistoryBuffer),
    "Failed to create VDENC BRC History Buffer");

```

Intel Media Driver - codechal_vdenc_hevc.cpp, INTEL MEDIA DRIVER GITHUB SOURCE CODE REPOSITORY, available at: https://github.com/intel/media-driver/agnostic/common/codec/hal/codechal_vdenc_hevc.cpp

driver/agnostic/common/codec/hal/codechal_vdenc_hevc.cpp (last visited February 2025) (emphasis added).

195. The Intel ‘896 Products determine first and second thresholds as a function of past input QP parameters as shown by the Intel ‘896 Products defining multiple threshold arrays that are selected based on historical encoding data. Specifically, the HevcVdencBrcConstSettings class in the Intel ‘896 Products contains multiple threshold arrays as shown below.

```

const int8_t HevcVdencBrcConstSettings::m_lowdelayDevThreshPB[] = {
    -45, -33, -23, -15, -8, 0, 15, 25,
};

const int8_t HevcVdencBrcConstSettings::m_lowdelayDevThreshVBR[] = {
    -45, -35, -25, -15, -8, 0, 20, 40,
};

const int8_t HevcVdencBrcConstSettings::m_lowdelayDevThreshI[] = {
    -40, -30, -17, -10, -5, 0, 10, 20,
};

const double HevcVdencBrcConstSettings::m_devThreshIFPNEG[] = {
    0.80, 0.60, 0.34, 0.2,
};

```

Intel Media Driver - codechal_vdenc_hevc.cpp, INTEL MEDIA DRIVER GITHUB SOURCE CODE REPOSITORY, available at: https://github.com/intel/media-driver/agnostic/common/codec/hal/codechal_vdenc_hevc.cpp (last visited February 2025) (emphasis added).

196. The Intel ‘896 Products compute a second quantization parameter based at least in part on the first quantization parameter, the first and second thresholds, and a number of bits occupied by the encoded macroblock. Specifically, the Intel ‘896 Products compute a second QP using delta QP calculations and QP adjustments based on the number of bits occupied by the encoded macroblock. The excerpt below from the Intel ‘896 Products shows how the products make QP adjustments in calculating a second quantization parameter.

```

setting->brcSettings.topFrmSzThrForAdapt2Pass_U8 = m_brcSettings.m_topFrmSzThrForAdapt2Pass_U8;
setting->brcSettings.botFrmSzThrForAdapt2Pass_U8 = m_brcSettings.m_botFrmSzThrForAdapt2Pass_U8;
setting->brcSettings.topQPDeltaThrForAdapt2Pass_U8 = m_brcSettings.m_topQPDeltaThrForAdapt2Pass_U8;
setting->brcSettings.botQPDeltaThrForAdapt2Pass_U8 = m_brcSettings.m_botQPDeltaThrForAdapt2Pass_U8;
setting->brcSettings.deltaQPForSadZone0_S8 = m_brcSettings.m_deltaQPForSadZone0_S8;
setting->brcSettings.deltaQPForSadZone1_S8 = m_brcSettings.m_deltaQPForSadZone1_S8;
setting->brcSettings.deltaQPForSadZone2_S8 = m_brcSettings.m_deltaQPForSadZone2_S8;
setting->brcSettings.deltaQPForSadZone3_S8 = m_brcSettings.m_deltaQPForSadZone3_S8;
setting->brcSettings.deltaQPForMvZero_S8 = m_brcSettings.m_deltaQPForMvZero_S8;
setting->brcSettings.deltaQPForMvZone0_S8 = m_brcSettings.m_deltaQPForMvZone0_S8;
setting->brcSettings.deltaQPForMvZone1_S8 = m_brcSettings.m_deltaQPForMvZone1_S8;
setting->brcSettings.deltaQPForMvZone2_S8 = m_brcSettings.m_deltaQPForMvZone2_S8;

setting->brcSettings.reEncodePositiveQPDeltaThr_S8 = m_brcSettings.m_reEncodePositiveQPDeltaThr_S8;
setting->brcSettings.reEncodeNegativeQPDeltaThr_S8 = m_brcSettings.m_reEncodeNegativeQPDeltaThr_S8;
setting->brcSettings.sceneChgPrevIntraPctThreshold_U8 = m_brcSettings.m_sceneChgPrevIntraPctThreshold_U8;
setting->brcSettings.sceneChgCurIntraPctThreshold_U8 = m_brcSettings.m_sceneChgCurIntraPctThreshold_U8;

```

Intel Media Driver - codechal_vdenc_hevc.cpp, INTEL MEDIA DRIVER GITHUB SOURCE CODE REPOSITORY, available at: https://github.com/intel/media-driver/agnostic/common/codec/hal/codechal_vdenc_hevc.cpp (last visited February 2025) (emphasis added).

197. The Intel ‘896 Products re-encode the decoded macroblock based on the second quantization parameter. Specifically, Intel ‘896 Products perform multiple encoding passes, including Picture Application Kernel (PAK)-only passes, where content is re-encoded with updated parameters, such as the second quantization parameter. The HuCBrcUpdate function in the Intel ‘896 Products is used to update the encoding parameters between encoding passes.

```

1278 MOS_STATUS CodechalVdencHevcState::HuCBrcUpdate()
1279 {
1280     MOS_STATUS eStatus = MOS_STATUS_SUCCESS;
1281
1282     CODECHAL_ENCODE_FUNCTION_ENTER;
1283
1284     MOS_COMMAND_BUFFER cmdBuffer;
1285     CODECHAL_ENCODE_CHK_STATUS_RETURN(GetCommandBuffer(&cmdBuffer));
1286
1287     if (!m_singleTaskPhaseSupported || (m_firstTaskInPhase && !m_brcInit))
1288     {
1289         // Send command buffer header at the beginning (OS dependent)
1290         bool requestFrameTracking = m_singleTaskPhaseSupported ?
1291             m_firstTaskInPhase : 0;
1292         CODECHAL_ENCODE_CHK_STATUS_RETURN(SendPrologWithFrameTracking(&cmdBuffer, requestFrameTracking));
1293     }
1294
1295     int32_t currentPass = GetCurrentPass();
1296     if (currentPass < 0)
1297     {
1298         eStatus = MOS_STATUS_INVALID_PARAMETER;
1299         return eStatus;
1300     }

```

Intel Media Driver - codechal_vdenc_hevc.cpp, INTEL MEDIA DRIVER GITHUB SOURCE CODE REPOSITORY, available at: https://github.com/intel/media-driver/agnostic/common/codec/hal/codechal_vdenc_hevc.cpp (last visited February 2025) (emphasis added).

198. The Intel ‘896 Products provide the re-encoded macroblock. Specifically, the Intel ‘896 Products use pipeline flush commands to ensure that all encoding operations (including re-encoding) are completed. As shown in the following excerpt from the Intel ‘896 Products the ReadHcpStatus and ReadSliceSize functions capture the final output data. The command buffer is then used by the Intel ‘896 Products to provide the re-encoded macroblocks to the destination buffer.

```

// Send VD_PIPELINE_FLUSH command
MHW_VDBOX_VO_PIPE_FLUSH_PARAMS vdPipelineFlushParams;
MOS_ZeroMemory(&vdPipelineFlushParams, sizeof(vdPipelineFlushParams));
vdPipelineFlushParams.Flags.bWaitDoneMFX = 1;
vdPipelineFlushParams.Flags.bWaitDoneVDENC = 1;
vdPipelineFlushParams.Flags.bFlushVDENC = 1;
vdPipelineFlushParams.Flags.bWaitDoneVDCmdMsgParser = 1;
CODECHAL_ENCODE_CHK_STATUS_RETURN(m_vdencInterface->AddVdPipelineFlushCmd(&cmdBuffer, &vdPipelineFlushParams));
}

if (m_useBatchBufferForPakSlices)
{
    CODECHAL_ENCODE_CHK_STATUS_RETURN(Mhw_UnlockBb(
        m_osInterface,
        &m_batchBufferForPakSlices[m_currPakSliceIdx],
        m_lastTaskInPhase));
}

```

Intel Media Driver - codechal_vdenc_hevc.cpp, INTEL MEDIA DRIVER GITHUB SOURCE CODE REPOSITORY, available at: https://github.com/intel/media-driver/agnostic/common/codec/hal/codechal_vdenc_hevc.cpp (last visited February 2025) (emphasis added).

199. Intel has directly infringed and continues to directly infringe the ‘896 Patent by, among other things, making, using, offering for sale, and/or selling technology for optimizing encoded video streams by tailoring quality settings for macroblocks, including but not limited to the Intel ‘896 Products.

200. The Intel ‘896 Products are available to businesses and individuals throughout the United States.

201. The Intel ‘896 Products are provided to businesses and individuals located in this District.

202. By making, using, testing, offering for sale, and/or selling products and services comprising technology for optimizing encoded video streams by tailoring quality settings for macroblocks, including but not limited to the Intel ‘896 Products, Intel has injured Plaintiff and is liable to Plaintiff for directly infringing one or more claims of the ‘896 Patent, including at least claim 1 pursuant to 35 U.S.C. § 271(a).

203. To the extent applicable, the requirements of 35 U.S.C. § 287(a) have been met with respect to the ‘896 Patent.

204. As a result of Intel’s infringement of the ‘896 Patent, Plaintiff has suffered monetary damages, and seeks recovery in an amount adequate to compensate for Intel’s infringement, but in no event less than a reasonable royalty for the use made of the invention by Intel together with interest and costs as fixed by the Court.

COUNT VII
INFRINGEMENT OF U.S. PATENT NO. 9,749,713

205. Plaintiff references and incorporates by reference the preceding paragraphs of this Complaint as if fully set forth herein.

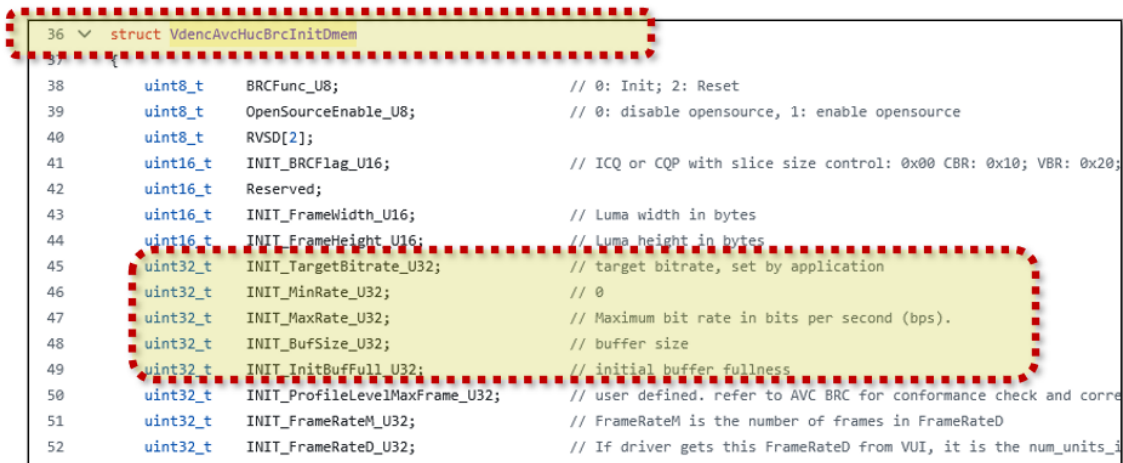
206. Intel designs, makes, uses, sells, and/or offers for sale in the United States products comprising technology for adaptive real-time streaming media frame processing.

207. Intel designs, makes, sells, offers to sell, imports, and/or uses the following products: the Intel Iris X^e; Intel Iris X^e; MAX, Intel Arc (including models: A310, A380, A580, A750, A770 8GB, A770 16GB, A350M, A370M, A530M, A550M, A570M, A730M, A770M, A30M, A40, A60M, A60); Intel Data Center GPU Max 1100; Intel Data Center GPU Flex 140; Intel Data Center GPU Flex 170; and Intel Data Center GPU Flex 170v (collectively, the “Intel ‘713 Product(s)”).

208. One or more Intel subsidiaries and/or affiliates use the Intel ‘713 Products in regular business operations.

209. The Intel ‘713 Products comprise a frame budget algorithm module configured to allocate a frame budget for an output media frame to generate an output frame index, by estimating a frame size of the output media frame based on a respective original frame size of a respective encoded frame in the received frame index. Specifically, the Intel ‘713 Products include a frame

budget allocation system that uses HuC BRC (Bitrate Control). As shown in the excerpt from Intel's materials, the Intel '713 Products estimate frame size based on original frame information from the prior frames and adjust the budget allocation based on this. The structure `VdencAvcHucBrcInitDmem` configures the initial budget. The structure `VdencAvcHucBrcUpdateDmem` tracks and allocates the frame budget after initialization.



```

36 struct VdencAvcHucBrcInitDmem
37 {
38     uint8_t BRCFunc_U8; // 0: Init; 2: Reset
39     uint8_t OpenSourceEnable_U8; // 0: disable opensource, 1: enable opensource
40     uint8_t RVSD[2];
41     uint16_t INIT_BRCFlag_U16; // ICQ or CQP with slice size control: 0x00 CBR: 0x10; VBR: 0x20;
42     uint16_t Reserved;
43     uint16_t INIT_FrameWidth_U16; // Luma width in bytes
44     uint16_t INIT_FrameHeight_U16; // Luma height in bytes
45     uint32_t INIT_TargetBitrate_U32; // target bitrate, set by application
46     uint32_t INIT_MinRate_U32; // 0
47     uint32_t INIT_MaxRate_U32; // Maximum bit rate in bits per second (bps).
48     uint32_t INIT_BufSize_U32; // buffer size
49     uint32_t INIT_InitBufFull_U32; // initial buffer fullness
50     uint32_t INIT_ProfileLevelMaxFrame_U32; // user defined. refer to AVC BRC for conformance check and corre
51     uint32_t INIT_FrameRateM_U32; // FrameRateM is the number of frames in FrameRateD
52     uint32_t INIT_FrameRateD_U32; // If driver gets this FrameRateD from VUI, it is the num_units_i

```

Intel Media Driver - encode_avc_huc_brc_init_packet.h, INTEL GITHUB SOURCE CODE REPOSITORY, available at: https://github.com/intel/media-driver/media_softlet/agnostic/common/codec/hal/enc/avc/packet/ (last visited February 2025) (emphasis added).

210. The Intel '713 Products comprise a processing algorithm module configured to determine first processing parameters based on the frame budget and the media frame, and determine, if the output frame does not fit within the frame budget, second processing parameters based on the frame budget and the first processing parameters. Specifically, the Intel '713 Products contain a processing algorithm for multi-pass encoding. The Intel '713 Products determine new parameters through a feedback loop. If the frame does not fit the budget, the Intel '713 Products recalculate parameters for a second encoding pass. Parameters adjusted by the Intel '713 Products include: `UPD_CQP_QpValue_U8` (increasing QP to reduce bitrate); `UPD_MinQpAdjustment_U8`

(controls the aggressiveness of QP changes); mode decision thresholds (favoring less complex prediction modes); and transform settings (using different transform types and/or sizes).

```

68  uint8_t  UPD_MaxNumPass_U8;           // 2
69  uint8_t  UPD_SceneChgWidth_U8[2];    // set both to MIN((NumP + 1) / 5, 6)
70  uint8_t  UPD_SceneChgDetectEn_U8;    // Enable scene change detection
71  uint8_t  UPD_SceneChgPrevIntraPctThreshold_U8; // =96. scene change previous intra percentage threshold
72  uint8_t  UPD_SceneChgCurIntraPctThreshold_U8; // =192. scene change current intra percentage threshold
73  uint8_t  UPD_IPAverageCoeff_U8;     // lowdelay ? 0 : 128
74  uint8_t  UPD_MinQpAdjustment_U8;    // Minimum QP increase step
75  uint8_t  UPD_TimingBudgetCheck_U8;  // Flag indicating if kernel will check timing budget.
76  int8_t   reserved_I8[4];            // must be zero
77  uint8_t  UPD_CQP_QpValue_U8;        // Application specified target QP in BRC_ICQ mode
78  uint8_t  UPD_CQP_FracQp_U8;        // Application specified fine position in BRC_ICQ mode
79  uint8_t  UPD_HMEDetectionEnable_U8; // 0: default, 1: HuC BRC kernel requires information from HME
80  uint8_t  UPD_HMECostEnable_U8;     // 0: default, 1: driver provides HME cost table
81  uint8_t  UPD_DisablePFrame8x8Transform_U8; // 0: enable, 1: disable

```

Intel Media Driver - encode_avc_huc_brc_update_packet.h, INTEL GITHUB SOURCE CODE REPOSITORY, available at: https://github.com/intel/media-driver/media_softlet/agnostic/common/codec/hal/enc/avc/packet/ (last visited February 2025) (emphasis added).

211. The Intel ‘713 Products comprise a processing module configured to process the media frame based on the first processing parameters. Specifically, the Intel ‘713 Products contain functionality including the *AvcVdencPkt* class, which manages the *VDenc* hardware to encode frames based on parameters. The Intel ‘713 Products utilize the *VDENC_WALKER_STATE* function, which traverses the media frame, applies specified parameters to the frame, and generates a compressed output.

```

46  AvcVdencPkt::AvcVdencPkt(
47      MediaPipeline *pipeline,
48      MediaTask *task,
49      CodechalHwInterfaceNext *hwInterface) :
50      CmdPacket(task),
51      m_pipeline(dynamic_cast<AvcVdencPipeline *>(pipeline)),
52      m_hwInterface(dynamic_cast<CodechalHwInterfaceNext *>(hwInterface))
53  {
54      ENCODE_CHK_NULL_NO_STATUS_RETURN(hwInterface);
55      ENCODE_CHK_NULL_NO_STATUS_RETURN(m_pipeline);
56      ENCODE_CHK_NULL_NO_STATUS_RETURN(m_hwInterface);
57
58      m_osInterface = hwInterface->GetOsInterface();
59      m_statusReport = m_pipeline->GetStatusReportInstance();
60      m_legacyFeatureManager = m_pipeline->GetFeatureManager();
61      m_featureManager = m_pipeline->GetPacketLevelFeatureManager(AvcVdencPipeline:
62      m_encodecp = m_pipeline->GetEncodeCp());

```

Intel Media Driver - encode_avc_vdenc_packet.cpp, INTEL GITHUB SOURCE CODE REPOSITORY, available at: https://github.com/intel/media-driver/media_softlet/agnostic/common/codec/hal/enc/avc/packet/ (last visited February 2025) (emphasis added).

212. The Intel ‘713 Products contain a frame padder configured to pad the processed media frame to generate the output media frame. The excerpt below from the Intel ‘713 Products shows how the padding function is called when the Intel ‘713 Products process frames. The Intel ‘713 Products contain passing logic including: `uint32_t pitc (psSurface->dwPitch);` `uint32_t UVPlaneOffset (psSurface->UPlaneOffset.iSurfaceOffset);` `uint32_t YPlaneOffset (psSurface->dwOffset);` and `uint32_t pad_rows (aligned_height - real_height).`


```

2022 void AvcVdencPkt::fill_pad_with_value(PMOS_SURFACE psSurface, uint32_t real_height,
2023 {
2024     ENCODE_CHK_NULL_NO_STATUS_RETURN(psSurface);
2025
2026     // unaligned surfaces only
2027     if (aligned_height <= real_height || aligned_height > psSurface->dwHeight)
2028     {
2029         return;
2030     }
2031
2032     if (psSurface->OsResource.TileType == MOS_TILE_INVALID)
2033     {
2034         return;
2035     }
2036
2037     if (psSurface->Format == Format_NV12 || psSurface->Format == Format_P010)
2038     {
2039         uint32_t pitch          = psSurface->dwPitch;
2040         uint32_t UVPlaneOffset = psSurface->UPlaneOffset.iSurfaceOffset;
2041         uint32_t YPlaneOffset  = psSurface->dwOffset;
2042         uint32_t pad_rows      = aligned_height - real_height;
2043         uint32_t y_plane_size  = pitch * real_height;
2044         uint32_t uv_plane_size = pitch * real_height / 2;
2045     }

```

Intel Media Driver - encode_avc_vdenc_packet.cpp, INTEL GITHUB SOURCE CODE REPOSITORY, available at: https://github.com/intel/media-driver/media_softlet/agnostic/common/codec/hal/enc/avc/packet/ (last visited February 2025) (emphasis added).

213. The Intel ‘713 Products contain a frame writer configured to provide the output media frame. Specifically, the Intel ‘713 Products provide the output media frame using a buffer that stores the compressed bitstream output (m_resBitstreamBuffer). In addition, the Intel ‘713 Products utilize the frame padder to format the output media frame.

```

162  MOS_STATUS AvcVdencPkt::Destroy()
163  {
164      ENCODE_FUNC_CALL();
165      ENCODE_CHK_STATUS_RETURN(m_statusReport->UnregistObserver(this));
166
167      return MOS_STATUS_SUCCESS;
168  }
169
170  MOS_STATUS AvcVdencPkt::Submit(
171      MOS_COMMAND_BUFFER* commandBuffer,
172      uint8_t packetPhase)
173  {
174      ENCODE_FUNC_CALL();
175
176      MOS_COMMAND_BUFFER* cmdBuf = commandBuffer;
177      ENCODE_CHK_STATUS_RETURN(Mos_Solo_PreProcessEncode(m_osInterface, &m_basicFeature->m_resBitstreamBuffer,

```

Intel Media Driver - encode_avc_vdenc_packet.cpp, INTEL GITHUB SOURCE CODE REPOSITORY, available at: https://github.com/intel/media-driver/media_softlet/agnostic/common/codec/hal/enc/avc/packet/ (last visited February 2025) (emphasis added).

214. Intel has directly infringed and continues to directly infringe the ‘713 Patent by, among other things, making, using, offering for sale, and/or selling technology for adaptive real-time streaming media frame processing, including but not limited to the Intel ‘713 Products.

215. The Intel ‘713 Products are available to businesses and individuals throughout the United States.

216. The Intel ‘713 Products are provided to businesses and individuals located in this District.

217. By making, using, testing, offering for sale, and/or selling products and services comprising technology for adaptive real-time streaming media frame processing, including but not limited to the Intel ‘713 Products, Intel has injured Plaintiff and is liable to Plaintiff for directly infringing one or more claims of the ‘713 Patent, including at least claim 8 pursuant to 35 U.S.C. § 271(a).

218. To the extent applicable, the requirements of 35 U.S.C. § 287(a) have been met with respect to the ‘713 Patent.

219. As a result of Intel's infringement of the '713 Patent, Plaintiff has suffered monetary damages, and seek recovery in an amount adequate to compensate for Intel's infringement, but in no event less than a reasonable royalty for the use made of the invention by Intel together with interest and costs as fixed by the Court.

PRAYER FOR RELIEF

WHEREFORE, Plaintiff OptiMorphix, Inc. respectfully requests that this Court enter:

- A. A judgment in favor of Plaintiff that Intel has infringed, either literally and/or under the doctrine of equivalents, the '273, '901, '388, '361, '015, '896, and '713 Patents;
- B. An award of damages resulting from Intel's acts of infringement in accordance with 35 U.S.C. § 284;
- C. A judgment and order finding that Intel's infringement of the '273 Patent was willful, wanton, malicious, bad-faith, deliberate, consciously wrongful, flagrant, or characteristic of a pirate within the meaning of 35 U.S.C. § 284 and awarding to Plaintiff enhanced damages;
- D. A judgment and order finding that Intel's infringement of the '901, '388, '361, and '015 Patents after the date of service of the Original Complaint in the above-captioned action was willful, wanton, malicious, bad-faith, deliberate, consciously wrongful, flagrant, or characteristic of a pirate within the meaning of 35 U.S.C. § 284 and awarding to Plaintiff enhanced damages;

- E. A judgment and order finding that this is an exceptional case within the meaning of 35 U.S.C. § 285 and awarding to Plaintiff reasonable attorneys' fees against Intel;
- F. Any and all other relief to which Plaintiff may show themselves to be entitled.

JURY TRIAL DEMANDED

Pursuant to Rule 38 of the Federal Rules of Civil Procedure, Plaintiff OptiMorphix, Inc. requests a trial by jury of any issues so triable by right.

Dated: March 3, 2025

BAYARD, P.A.

OF COUNSEL:

Dorian S. Berger
Daniel P. Hipskind
Erin E. McCracken
BERGER & HIPSKIND LLP
9538 Brighton Way, Ste. 320
Beverly Hills, CA 90210
Telephone: 323-886-3430
Facsimile: 323-978-5508
E-mail: dsb@bergerhipskind.com
E-mail: dph@bergerhipskind.com
E-Mail: eem@bergerhipskind.com

/s/ Stephen B. Brauerman
Stephen B. Brauerman (No. 4952)
Ronald P. Golden III (No. 6254)
600 N. King Street, Suite 400
P.O. Box 25130
Wilmington, Delaware 19801
(302) 655-5000
sbrauerman@bayardlaw.com
rgolden@bayardlaw.com

Attorneys for Plaintiff
OptiMorphix, Inc.