Nos. 15-1640, 15-1641

IN THE UNITED STATES COURT OF APPEALS
FOR THE FEDERAL CIRCUIT

U.S. ETHERNET INNOVATIONS, LLC,

Plaintiff-Appellant,

v.

ACER, INC., ACER AMERICA CORPORATION, APPLE INC., ASUS COMPUTER INTERNATIONAL, ASUSTEK COMPUTER INC., DELL INC., FUJITSU LIMITED, FUJITSU AMERICA, INC., GATEWAY, INC., HEWLETT-PACKARD CO., SONY CORPORATION, SONY CORPORATION OF AMERICA, SONY ELECTRONICS INC., TOSHIBA CORPORATION, TOSHIBA AMERICA INFORMATION SYSTEMS, INC., J.C. PENNEY COMPANY, INC., RENT-A-CENTER, INC., MARVELL SEMICONDUCTOR, INC., INTEL CORPORATION,

Defendants-Appellees.

Appeals from the United States District Court
for the Northern District of California
Case Nos. 4:10-cv-3724; 4:10-cv-5254, Judge Claudia Wilken

NON-CONFIDENTIAL BRIEF OF PLAINTIFF-APPELLANT

ROBBINS GELLER RUDMAN & DOWD LLP
JOHN C. HERMAN
RYAN K. WALSH
PETER M. JONES
3424 Peachtree Road, N.E., Suite 1650
Atlanta, GA 30326
Telephone: 404-504-6500

Counsel for Plaintiff-Appellant
U.S. Ethernet Innovations, LLC

AMENDED CERTIFICATE OF INTEREST

Counsel for Plaintiff-Appellant U.S. Ethernet Innovations, LLC, certifies the following:

1. The full name of every party represented by me is:

   U.S. Ethernet Innovations, LLC

2. The name of the real party in interest represented by me is:

   U.S. Ethernet Innovations, LLC

3. All parent corporations and any publicly held companies that own ten percent (10%) or more of the stock of the parties represented by me are:

   U.S. Ethernet Innovations, LLC is a subsidiary of Parallel Technology, LLC. No publicly traded company owns more than ten percent (10%) of U.S. Ethernet Innovations, LLC.

4. The names of all law firms and the partners and associates that appeared for the parties now represented by me in the trial court are:

   John C. Henman; Ryan K. Walsh; Peter M. Jones; Robert J. Leonard; Carlton R, Jones; David Lee Gann; Donald Sean Nation; E. Joseph Benz III; Eugene Mikolajczyk; Jason S. Jackson; Jessica M. Kattula; John K. Grant; Stephen R. Astley; Andrew Thomas Rees
   ROBBINS GELLER RUDMAN & DOWD LLP

   Thomas John Ward, Jr.; Bruce A. Smith; Jack Wesley Hill
   WARD, SMITH & HILL, PLLC

TABLE OF CONTENTS

I. JURISDICTIONAL STATEMENT 1

II. STATEMENT OF THE ISSUES 2

III. STATEMENT OF THE CASE 2

IV. STATEMENT OF THE FACTS 6

A. Background 6

1. Ethernet Technology 7

2. 3Com Revived Ethernet by Developing Technology That Substantially Improved Ethernet Performance. 8

3. 3Com's Novel Parallel Tasking Technology Enjoys Tremendous Success. 11

4. Intel Intentionally Copied 3Com's Parallel Tasking Technology. 13

5. The Parallel Tasking Patents Were Found Valid and Infringed by Realtek in the Northern District of California. 21

B. The Prior Art 22

C. The Patented Invention 25

D. The SONIC Is Not Large Enough to Hold Even a Minimum-Sized Ethernet Frame. 28

E. 3Com and Its Licensees Specifically Authorized to Practice the '872 Patent Marked Their Products. 30

F. The Broad, Portfolio-Wide Agreements Involving IBM and Intel Did Not Directly Relate to the Parallel Tasking Patents. 31

V. SUMMARY OF THE ARGUMENT 32

VI. ARGUMENT 34

A. Standard of Review 34

B. The Parallel Tasking Patents Are Valid over the SONIC as a Matter of Law. 35

1. Defendants Cannot Meet Their High Burden of Proof of Establishing Invalidity by Clear and Convincing Evidence. 36

2. The Overwhelming Evidence in the Record Demonstrates that the SONIC Cannot Anticipate the Parallel Tasking Patents. 38

a. Intel Admitted that Parallel Tasking Technology Was Novel in 1992. 38

b. The District Court's Order Fails to Consider the Great Weight of Evidence from Persons of Ordinary Skill in the Art. 39

c. This Understanding of Persons of Ordinary Skill in the Art Is Supported by Overwhelming Evidence 42

C. The District Court's Order Regarding Intel's Intentional Copying Should Be Reversed. 46

1. USEI Presented Overwhelming Evidence that Intel Intentionally Copied the Patented Technology 47

2. There Is Substantial Evidence that 3Com's Etherlink III Adapter Embodied the Parallel Tasking Patents. 49

D. The District Court Erred in Granting Partial Summary Judgment on the Basis of Marking 51

1. 3Com Made Reasonable Efforts to Ensure Compliance with the Marking Provision by Its Licensees. 53

a. The District Court's Order Applies An Unreasonable Standard. 54

b. The District's Ruling is Unprecedented in Case Law. 56

2. One Cannot Conclude that Patented Articles Were Sold Unmarked by a 3Com Licensee After April 30, 2005. 58

3. IBM and Intel Were Not Acting Within Their License Rights. 60

a. IBM Was Not Acting Within Its License Rights When It Purchased the Infringing Intel Products and Resold Them. 60

b. Intel Was Not Acting Within Its License Rights When It Sold the Infringing Intel Products. 62

4. At a Minimum, the District Court's Sweeping Relief Is Premature. 63

VII. CONCLUSION AND STATEMENT OF RELIEF SOUGHT 64

Confidential Material Omitted: The text and images omitted on pages 12, 14-19, 38, and 48 are materials taken from email and product documentation of Intel Corporation that contains information it has designated as confidential.

TABLE OF AUTHORITIES

CASES

3Com Corp. v. Realtek Semiconductor Corp.,
No. 03-02177-VZW (N.D. Cal.).....*passim*

A.C. Aukerman Co. v. R.L. Chaides Constr. Co.,
960 F.2d 1020 (Fed. Cir. 1992) 46, 47

Acticon Techs. LLC v. Creative Labs Inc.,
No. 2:08-cv-194-RSP, 2012 WL 708059 (E.D. Tex. Mar. 5, 2012) 53

Akamai Techs. v. Cable & Wireless Internet Servs.,
344 F.3d 1186 (Fed. Cir. 2003) 47

Am. Med. Sys. v. Med. Eng'g Corp.,
6 F.3d 1523 (Fed. Cir. 1993) 52

Anderson v. Liberty Lobby,
477 U.S. 242 (1986).....34

AstraZeneca LP v. Apotex, Inc.,
633 F.3d 1042 (Fed. Cir. 2010) 39

Bicon, Inc. v. Straumann Co.,
441 F.3d 945 (Fed. Cir. 2006) 45

Bott v. Four Star Corp.,
807 F.2d 1567 (Fed. Cir. 1986) 47

Canon Computer Sys. v. Nu-Kote Int'l,
134 F.3d 1085 (Fed. Cir. 1998) 36

Carnegie Mellon Univ. v. Marvell Tech. Group, Ltd.,
No. 09-290, 2014 U.S. Dist. LEXIS 4624 (W.D. Pa. Jan. 14, 2014).....47

Celotex Corp. v. Catrett,
477 U.S. 317 (1986).....34

Checkpoint Sys. v. All-Tag Sec. S.A.,
711 F.3d 1341 (Fed. Cir. 2013) 39

Cybiotronics, Ltd. v. Golden Source Elecs., Ltd.,
130 F. Supp. 2d 1152 (C.D. Cal. 2001) 34

Dynacore Holdings Corp. v. U.S. Philips Corp.,
363 F.3d 1263 (Fed. Cir. 2004) 34

Endo Pharm. Inc. v. Mylan Pharm. Inc., No. 11-CV-00717 (RMB/KW),
2014 U.S. Dist. LEXIS 10037 (D. Del. Jan. 28) 37, 38

Funai Elec. Co., Ltd., v. Daewoo Elecs. Corp.,
616 F.3d 1357 (Fed. Cir. 2010) 53, 55

Gasser Chair Co. v. Infanti Chair Mfg. Corp.,
60 F.3d 770 (Fed. Cir. 1995) 47

In re Baxter Travenol Labs,
952 F.2d 388 (Fed. Cir. 1991) 39

Mass. Institute of Tech. v. Abacus Software, Inc.,
No. 5:01-cv-344, 2004 WL 5268123 (E.D. Tex. Aug. 4, 2004) 58

Maxwell v. J. Baker, Inc.,
86 F.3d 1098 (Fed. Cir. 1996) 51, 52, 53, 54

Microsoft Corp. v. i4i Ltd. P'ship,
__ U.S. __, 131 S. Ct. 2238 (2011).....36

Minn. Mining & Mfg. Co. v. Johnson & Johnson Orthopaedics, Inc.,
976 F.2d 1559 (Fed. Cir. 1992) 37, 38

Morpho Detection, Inc. v. Smiths Detection, Inc.,
957 F. Supp. 2d 655 (E.D. Va. 2013) 49

Phillip M. Adams & Assocs., LLC v. Winbond Elecs. Corp.,
No. 1:05-cv-64-TS, 2010 WL 3522097 (D. Utah Sept. 8, 2010).....63

Phillips v. AWH Corp.,
415 F.3d 1303 (Fed. Cir. 2005) (en banc) 41

Power Integrations, Inc. v. Fairchild Semiconductor Int'l, Inc.,
711 F.3d 1348 (Fed. Cir. 2013) 41, 43

Renishaw PLC v. Marposs Societa Per Azioni,
158 F.3d 1243 (Fed. Cir. 1998) 41

Sciele Pharma, Inc. v. Lupin Ltd.,
684 F.3d 1253 (Fed. Cir. 2012) 36

SciMed Life Sys. v. Advanced Cardiovascular Sys.,
242 F.3d 1337 (Fed. Cir. 2001) 36

Sentry Prot. Prods. v. Eagle Mfg. Co.,
400 F.3d 910 (Fed. Cir. 2005) 51, 52

State Contracting & Eng'g Corp. v. Condotte Am., Inc.,
346 F.3d 1057 (Fed. Cir. 2003) 52

Sundance, Inc. v. DeMonte Fabricating Ltd.,
550 F.3d 1356 (Fed. Cir. 2008) 39

Tulip Computs. Int'l B.V. v. Dell Computs. Corp.,
262 F. Supp. 2d 358 (D. Del. 2003).....61, 62

U.S. Ethernet Innovations, LLC v. Acer, Inc., et al.,
No. 4:10-cv-3724 (N.D. Cal.).....2, 3, 4

U.S. Ethernet Innovations, LLC v. AT&T Mobility, LLC, et al.,
No. 4:10-cv-5254 (N.D. Cal.).....3, 4

U.S. Ethernet Innovations, LLC v. Texas Instruments Incorporated,
Nos. 15-1510, 15-1541, Dkt. No. 27 (Fed. Cir. July 1, 2015).....*passim*

V-Formation, Inc. v. Benetton Grp. SpA,
401 F.3d 1307 (Fed. Cir. 2005) 37

WiAV Sols. LLC v. Motorola, Inc.,
732 F. Supp. 2d 634 (E.D. Va. 2010) 56, 57

STATUTES, RULES AND REGULATIONS

28 U.S.C.
§1295(a)(1) 1
§1338(a) 1
§2107(a) 1

35 U.S.C.
§282.....36
§287.....*passim*

Federal Circuit Rule
Rule 28(a)(12).....1

Federal Rules of Appellate Procedure
Rule 4(a).....1

STATEMENT OF RELATED CASES

Pursuant to Federal Rule of Appellate Procedure 47.5, U.S. Ethernet Innovations, LLC ("USEI") states that there have been no prior appeals in or from the same civil action or proceeding in the lower court previously before this or any other appellate court.

There is one appeal pending in this Court that relates directly to the Court's decision in this appeal – *U.S. Ethernet Innovations, LLC v. Texas Instruments Incorporated*, No. 6:11-cv-491 (E.D. Tex.) (Appeal No. 15-1510, Cross-Appeal No. 15-1541) ("TI Action"). In an order dated May 20, 2015, this Court determined that the instant appeal and the appeal in the TI Action shall be treated as companion cases and will be heard before the same panel. Appeal No. 15-1510, Dkt. 26.

I. JURISDICTIONAL STATEMENT

(A) The district court had jurisdiction over this action pursuant to 28 U.S.C. §1338(a) because the complaint included a claim for patent infringement.

(B) This Court has jurisdiction pursuant to 28 U.S.C. §1295(a)(1) because the present appeal is from a final decision of a district court in a civil action arising under an Act of Congress relating to patents and because the present appeal presents a substantial question of patent law.

(C) This appeal was filed timely in accordance with 28 U.S.C. §2107(a) and Rule 4(a) of the Federal Rules of Appellate Procedure because the district court issued an order denying a motion by USEI to alter or amend the judgment on March 31, 2015, and the Notice of Appeal was filed on April 24, 2015.

(D) This appeal is taken from a final judgment that disposes of all of the claims and defenses in the district court.

II. STATEMENT OF THE ISSUES

(1) Whether the district court erred in denying USEI's motion for summary judgment that the '872 and '094 Patents were valid as a matter of law in view of the SONIC prior art reference disclosed, described, and distinguished in the patents' shared specification, and in granting Defendants-Appellees' motion for summary judgment of invalidity based on this same reference;

(2) Whether the district court erred in denying USEI's motion for summary judgment that Intel Corporation should be prevented from asserting any equitable defenses because it intentionally copied the patented technology; and

(3) Whether the district court erred in granting Defendants-Appellees' motion for partial summary judgment that USEI is not entitled to recover for certain acts of infringement because the patentee failed to comply with the marking statute.

III. STATEMENT OF THE CASE

Plaintiff-Appellant USEI filed suit against eight original equipment manufacturers and their various subsidiaries[1] on October 9, 2009, alleging, *inter alia*, infringement of United States Patent Nos. 5,434,872 (the "'872 Patent") and

[1] The original defendants to the Acer Action include the following current Defendants-Appellees: Acer, Inc., Acer America Corporation, Gateway, Inc.; Apple Inc.; Asus Computer International, Asustek Computer Inc.; Dell Inc.; Fujitsu Limited, Fujitsu America, Inc.; Hewlett-Packard Co.; Sony Corporation, Sony Corporation of America, Sony Electronics Inc.; and Toshiba Corporation and Toshiba America Information Systems, Inc.

5,732,094 (the "'094 Patent") (collectively, the "Parallel Tasking Patents"[2]).[3] *U.S. Ethernet Innovations, LLC v. Acer, Inc., et al.*, No. 4:10-cv-3724 (the "Acer Action"); A437-A438. Defendants-Appellees Intel Corporation ("Intel") and Marvell Semiconductor, Inc. ("Marvell") intervened in the case, arguing that USEI's allegations of infringement were directed to their respective products. *See* A682-A684; A691-A693. USEI asserted counterclaims against both Intel and Marvell.[4] *See* A703; A707-A708; A825-A826; A954.

USEI sued Defendants-Appellees J.C. Penney Company, Inc. and Rent-A-Center, Inc., asserting infringement of the same Parallel Tasking Patents in *U.S. Ethernet Innovations, LLC v. AT&T Mobility, LLC, et al.*, No. 4:10-cv-5254 (the

---

[2] For ease of reference in this appeal, the '872 and '094 Patents will be referred to as the Parallel Tasking Patents. As discussed in more detail below, "Parallel Tasking" was a trademark of 3Com Corporation that referred to technology embodied in the '872 and '094 Patents. While the evidence below indicated that additional patents may have included aspects of Parallel Tasking technology, unrebutted testimony from virtually all former 3Com employees confirmed that the '872 and '094 Patents were "Parallel Tasking" Patents. *See, e.g.*, A6595:6-A6596:13.

[3] In addition to the related TI Action, the Parallel Tasking Patents were previously litigated from 2003 to 2008 in the Northern District of California, resulting in a jury finding the patents valid and infringed. *3Com Corp. v. Realtek Semiconductor Corp.*, No. 03-02177-VZW (N.D. Cal.) (the "Realtek Litigation"); A6752-A6756; A4297-A4340.

[4] All defendants and intervenors will be referred to collectively herein as "Defendants."

"AT&T Action"), which the district court related to the Acer Action[5] for the purposes of claim construction. A1307-A1309.

On November 8, 2010, Defendants moved to stay all claims against the initial defendants and to allow only the claims asserted against the intervenors to move forward in separate, severed actions.[6] Judge Ware denied the motion without prejudice, but agreed to stay all discovery unrelated to claim construction indefinitely until the court could determine the proper structure for the litigation.[7] USEI's claims of infringement were effectively stayed for more than two years, until January 17, 2013, when the district court again denied Defendants' motion to restructure this litigation and re-opened discovery.[8]

Less than four months later, the district court allowed Defendants to submit a motion for summary judgment addressing the defense of failure to comply with the marking statute and whether USEI's recovery should be limited to nominal damages.

---

[5] The Acer Action and the AT&T Action will be referred to collectively herein as the "Acer Action."

[6] *U.S. Ethernet Innovations, LLC v. Acer, Inc., et al.*, No. 4:10-cv-3724, Dkt. No. 449 (N.D. Cal.).

[7] *U.S. Ethernet Innovations, LLC v. Acer, Inc., et al.*, No. 4:10-cv-3724, Dkt. Nos. 475, 516, 535 (N.D. Cal.).

[8] *U.S. Ethernet Innovations, LLC v. Acer, Inc., et al.*, No. 4:10-cv-3724, Dkt. No. 680 (N.D. Cal.).

See A7459-A7462; *see also* A6554-A6581; A5988-A5990. Despite the undeveloped record, the district court granted the motion on the issue of marking and denied without prejudice the motion with regard to nominal damages. A34-A61.

During the last months of discovery in this case, USEI was engaged in a week-long trial on the issue of validity of the Parallel Tasking Patents with Intel's joint defense partner, Texas Instruments Incorporated ("TI"), in the TI Action.[9] Shortly before trial, the district court denied TI's motion for summary judgment, which sought a finding that the Parallel Tasking Patents were anticipated by the SONIC DP83932 prior art reference ("SONIC") as a matter of law. A18527-A18533. On April 11, 2014, the jury in the TI Action returned a verdict for USEI on all issues presented to them, including a finding that the asserted claims of the Parallel Tasking Patents were valid over the SONIC. A18535-A18537.

Three months later, Defendants in this case submitted a motion for summary judgment that raised the identical validity argument previously rejected by both judge and jury in the TI Action. A9374-A9378. Defendants presented the same facts and

[9] Both the district court in this case and the district court in the TI Action have recognized that Intel and TI are joint defense partners that have coordinated a single invalidity defense against USEI. A8336-A8338; A18517-A18524. Indeed, lead counsel for Intel was present in the courtroom throughout the invalidity trial in the TI Action, conferring regularly with counsel for TI. *U.S. Ethernet Innovations, LLC v. Texas Instruments Incorporated*, Nos. 15-1510, 15-1541, Dkt. No. 27 at 6 n.8 (Fed. Cir. July 1, 2015).

the same law in support of the same argument that the SONIC reference anticipated the Parallel Tasking Patents.[10] On November 7, 2014, the district court granted Defendants' motion, finding all asserted claims of the Parallel Tasking Patents anticipated by the SONIC reference as a matter of law. A68-A73. In the court's brief discussion regarding this issue, the court does not mention the prior contradictory summary judgment order or jury verdict in the TI Action on the identical argument. *Id.*[11] The court issued a Final Judgment on December 1, 2014. A94-A95; A96-A97.

IV. STATEMENT OF THE FACTS

A. Background

The patents at issue in this appeal relate to transformative technology developed by 3Com Corporation ("3Com") in the early 1990s that dramatically improved Ethernet networking performance. Bob Metcalfe, commonly recognized as the inventor of Ethernet, founded 3Com. A10559:89:22-A10560:90:19; A14146; A3531. One of 3Com's principal product areas was Ethernet network adapters (also known as "network interface controllers" or "NIC's"). A10560:90:20-A10561:91:3.

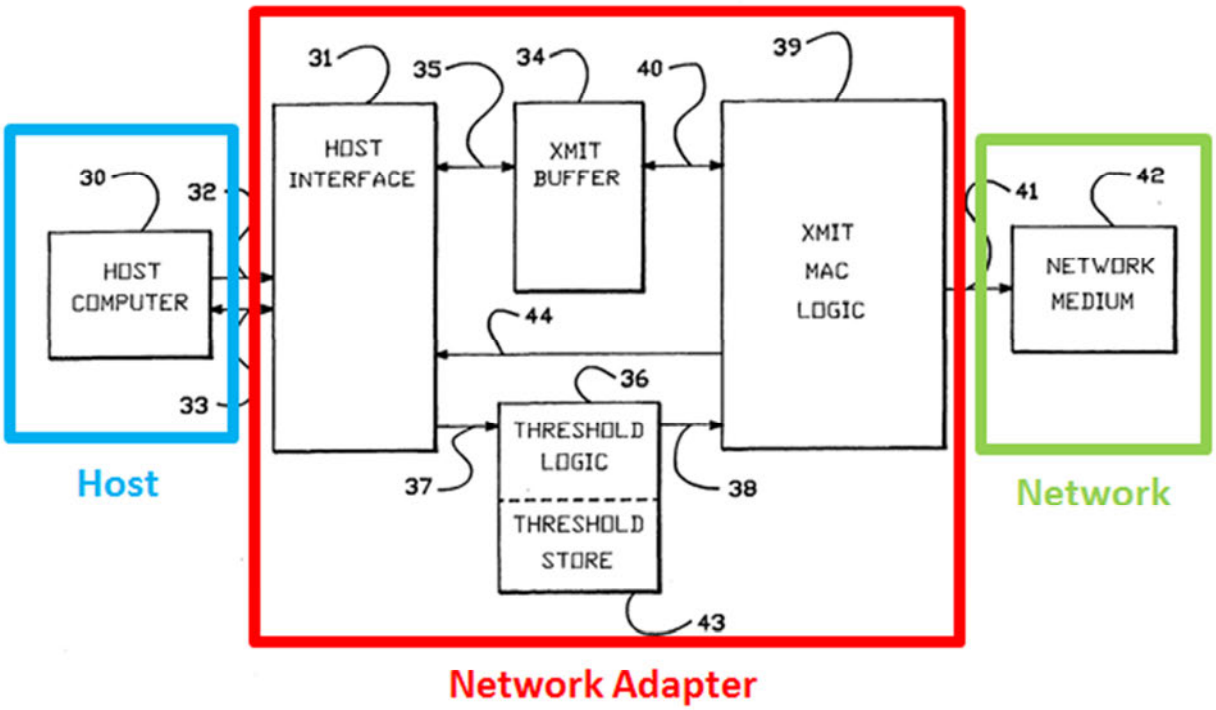
---

[10] *U.S. Ethernet Innovations, LLC v. Texas Instruments Incorporated*, No. 15-1510, 15-1541, Dkt. No. 27 at 26, 54 (Fed. Cir. July 1, 2015).

[11] The Acer Court's order also does not reference the fact that the Parallel Tasking Patents were found by the jury to be valid over a precursor to the SONIC and another similar, FIFO-based reference (Lare) in the Realtek Litigation in the Northern District of California. *See* A6752-A6756; A9149-A9155.

1. Ethernet Technology

Ethernet is a technology used to manage communications between multiple individual computers on a local area network ("LAN"), such as in a home or office environment. A14146. A network adapter is used to serve as an interface between the host computer and the network. *Id.*



A110 (colored descriptions added).

In a typical system, when a host computer sends data out onto a network, that data is first broken down into units known as frames. A10607:9-A10609:6.[12] The

[12] The Ethernet specification allows the network adapters on each computer to communicate with one another, in part, by using a specifically defined range of frame

data frames are sent from the host computer to a buffer memory located on the network adapter and then from the network adapter onto the network. A10607:9-A10609:6.

2. 3Com Revived Ethernet by Developing Technology that Substantially Improved Ethernet Performance

In the early 1990s, the conventional wisdom in the networking industry was that Ethernet networking technology could no longer be improved and would soon be overtaken by other networking protocols.[13] During this time, the market for Ethernet network adapters was highly commoditized, with little to distinguish one brand of Ethernet network adapter from another. A10555:4-A10556:16.

3Com engineers cut against this conventional wisdom and revolutionized Ethernet network adapter technology by developing the famous Parallel Tasking

---

sizes. A10608:19-A10609:6. The Ethernet specification provides for a minimum frame size of 64 bytes. A14275:51:6-20; A10669:4-12; A10608:10-A10609:6; A14147-A14148. A frame less than 64 bytes is not a valid Ethernet frame. A14147-A14149; A14208-A14209; A14275:51:13-20. Thus, an attempt to transmit a frame less than 64 bytes will result in an error. *Id*. Additionally, prior to the development of full-duplex technology, computers on a LAN network had to share the connection, and as a result only one frame could be transmitted at any one time. A10609:22-A10611:15. If two frames are transmitted at the same time, a "collision" of data on the wire may occur, resulting in an error and the need to retransmit the lost data. *Id.*

[13] A10517:19-A10518:16; A10611:18-A10612:16; A6625-A6628.

technology.[14] 3Com's Parallel Tasking technology improved computer performance, including host processor efficiency, when communicating over an Ethernet network. One major aspect of Parallel Tasking technology improved networking performance by reducing the latency (*i.e.*, delay) associated with transmission of data from an adapter capable of storing Ethernet frames by incorporating an early transmit feature.[15] Specifically, this aspect of the Parallel Tasking technology reduced the latency associated with waiting for all of the data of a frame to be downloaded from the host computer to the network adapter before beginning the transmission to the network.[16] 3Com's Buyers Guides explained as follows:

> All Fast EtherLink and EtherLink III adapters include 3Com's exclusive Parallel Tasking technology. To improve your network's data throughput, Parallel Tasking overlaps, or pipelines, processing task. For example, a Fast EtherLink or EtherLink III adapter ***begins writing the first portion of a data frame onto the wire before it has completed reading the frame from PC memory***.

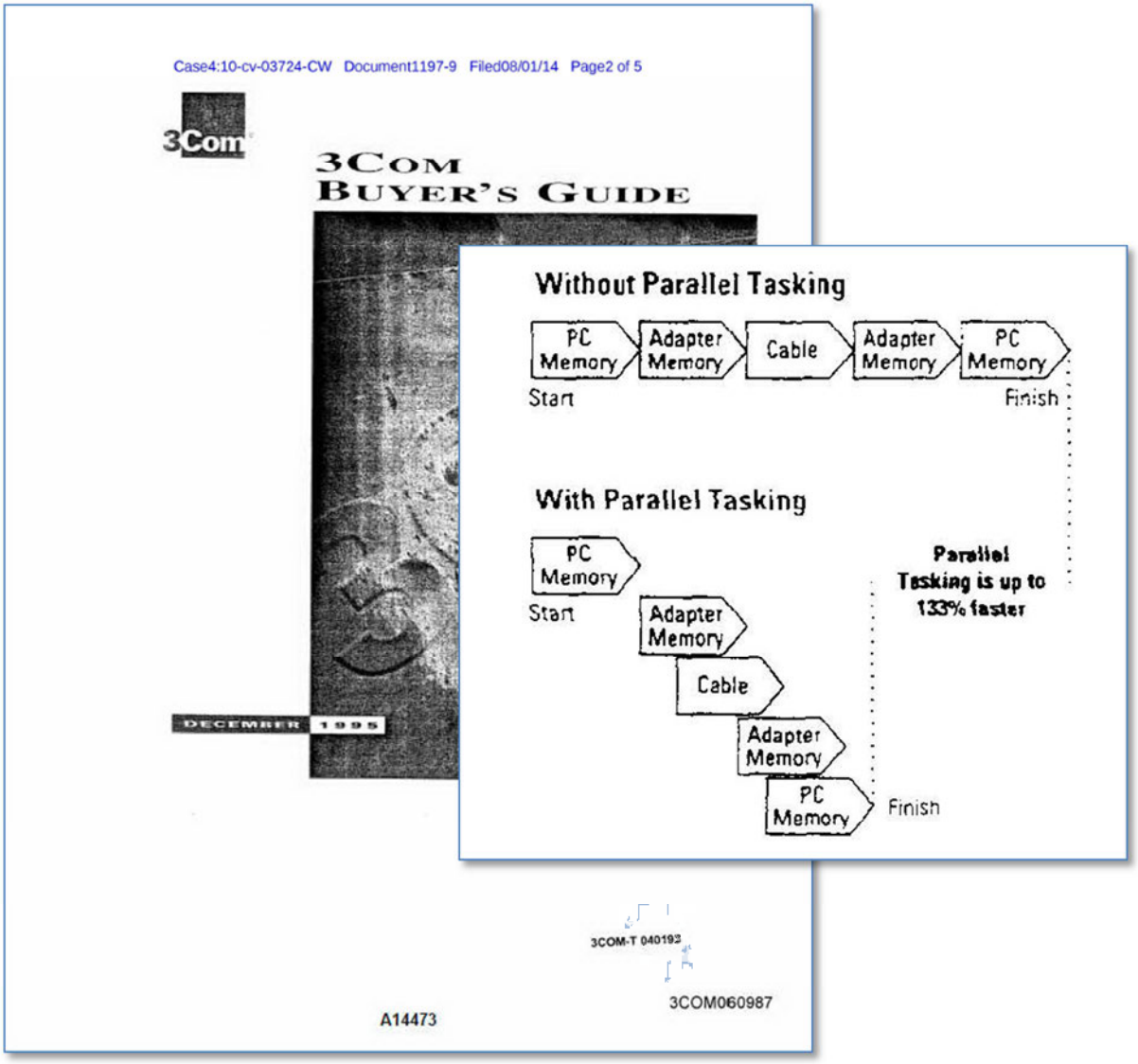
A14476 (emphasis added). 3Com's Buyer's Guide also illustrated the increased efficiency accomplished by Parallel Tasking technology as follows:

---

[14] A10555:4-A10556:16; A15078:114:15-117:6; A14487:3-20; A11477:19-A11481:20; A6625-A6628.

[15] A14198-A14200; A14305-A14306; A10519:7-A10521:10; A14125:157:8-A14126:158:12; A3535-3539.

[16] *Id.*



A14473; A14476. This early transmit aspect of Parallel Tasking technology is found in the '872 and '094 Patents.[17]

[17] The '094 Patent is a continuation of the '872 Patent and includes method claims, covering similar technology to the apparatus claims of the '872 Patent. A108-A138; A139-A170.

It is undisputed that the term "Parallel Tasking" referred to the technology disclosed in the '872 and '094 Patents. Numerous former 3Com employees, like Annette Davis, confirmed this in unrebutted sworn testimony:[18]

> **Q:** And the term "Parallel Tasking," what does that refer to?
>
> **A:** That was a trademark that 3Com used to represent patented technologies.
>
> **Q:** And by patented technologies, what do you mean?
>
> **A:** I mean the '459, the '872, and the '094 patents.

3. 3Com's Novel Parallel Tasking Technology Enjoys Tremendous Success

In the late 1980s and early 1990s, 3Com developed the Parallel Tasking technology in development projects named Slapstick and Vulcan.[19] As the lead inventors explained, 3Com's first commercial network adapter that incorporated this technology was the EtherLink III:

- **Q:** It's your understanding that the EtherLink III product incorporated the inventions from the Parallel Tasking patents; is that right?

[18] A6595:17-21; *see also* A6635-A6638; A6625-A6628; A6630-A6633; A14487:1-16; A14488:16-A14489:24; A11471:19-A11472:21; A14226:20-23; A14251:16-21; A15106:58:17-59:8; A15133:167:18-168:8; A14476; A6758.

[19] A6635-A6638; A6625-A6628; A6630-A6633; A11472:5-21; A14487:1-16; A11574:25-A11575:15; A11471:19-A11472:21; A14226:20-23; A14251:16-21; A15106:58:17-59:8; A15133:167:18-168:8; A14497:20-A14498:4.

A: That's correct.[20]

- **A:** After I left the company and the EtherLink III adapter became popular and sold very well, and I was aware that certain of these technologies, you know, the Parallel Tasking related patents were embodied in that product, I made the assumption that those patents were contributing to that.

  Q: And which patents were those, do you recall?

  A: Well, of the ones I have in front of me, they would be the '094, the '872 and the '459.[21]

The EtherLink III, released in 1992, led 3Com to dominate the network adapter market for the next decade.[22] As described by one of the lead inventors, the Parallel Tasking technology in the EtherLink III "represented substantial improvement over what was available, in terms of the practice at the time." A11584:3-A11585:5. Benchmark testing showed that it improved performance by 25%-100%.[23] As a result, the EtherLink III won multiple industry awards recognizing products considered to be the best of their category.[24]

---

[20] A14489:16-19.

[21] A14487:7-16; *see also* A14251:16-21; A14226:20-23; A14251:16-21.

[22] A14488:16-A14489:12; A14491:16-A14493:11; A11473:20-A11475:15; A11478:2-A11481:20; A11468:4-A11469:16; A11469:22-A11471:17; A5162-A5164.

[23] A11478:16-A11479:12; A14251:2-14; A14476. As described in more detail below, [redacted] A8656.

[24] A10522:12-A10523:12; A10566:20-A10567:23; A15067:7-23.

The efficiencies gained as a result of Parallel Tasking technology helped 3Com become the market leader for network adapters, selling billions of dollars of "Parallel Tasking" products.[25] According to Scott Emery, a 3Com engineer involved in the development of the EtherLink III, the product was "phenomenally successful and sold more than any other Ethernet adapter that 3Com had ever produced, and probably at that point anybody else, ever." A15148:228:3-10. Douglas Spreng, then General Manager of 3Com's entire Network Adapter Division, explained:

> Parallel Tasking gave us a clear, major, measurable differentiation in the marketplace …. So it was really more of – of a question of how quickly and effectively we applied that technology to the product line. And it was in every piece of the product line, both present and future, that we could come up with. It was the heart of every product we built.[26]

Paul Sherer, one of the lead inventors of the Parallel Tasking technology, explained that this technology was the largest contributor to 3Com's profitability and that it helped dramatically increase 3Com's sales and market share. A11477:19-A11481:20.

4. Intel Intentionally Copied 3Com's Parallel Tasking Technology

The success of 3Com's Parallel Tasking products did not go unnoticed by its competitors, particularly Intel. *See, e.g.*, A11439:10-11443:5. Only months after the

---

[25] A6596:2-13; *see also* A11429:5-17; A11439:10-A11441:13; A14488:16-A14489:12; A14491:16-A14493:11; A11473:20-A11475:15; A11478:2-A11481:20; A11468:4-A11469:16; A11469:22-A11471:17.

[26] A11471:8-17; *see also* A6592:19-23.

release of 3Com's EtherLink III, internal Intel e-mails and product documentation show that Intel engineers: [redacted]

In October 1992, Intel engineers confirmed [redacted]:

[27] Intel has repeatedly maintained that its e-mails and product documentation showing the reverse engineering and copying of 3Com's Parallel Tasking technology during the early 1990s contain highly confidential information that must be filed under seal. While USEI does not believe that these 20-year old documents require any confidential treatment, it is nevertheless complying with Intel's designations.



[A8656;] *see also* A8655-A8664. In that same e-mail, Intel's engineers confirmed

[ ]

[redacted]

[A8659; A8656.] The e-mail concludes [redacted]

[redacted]

[redacted]

[redacted]

[A8656; A8658.]

[redacted]

[redacted]

[redacted]

[redacted]

[A8666;] *see also* A8666-A8672. The internal Intel e-mail summarizing [redacted]



[A8668; A8669.]

On April 31, 1993, an internal Intel memorandum states: [

] A6643.

Shortly thereafter, Intel product documentation [

]

1.2 Enhancements to the 82595

The 82595TX is fully backwards compatible to the 82595, both in pinout and software. However, the 82595TX contains several advanced functions from the 82595 which increase performance and ease of use. The following is a list of the major enhancements to the 82595TX:

Concurrent Processing Functionality

32-Bit Local Memory IO Port

Integrated Plug N' Play support

Added EEPROM Interface for Plug N' Play

Flash addressing up to 1 Mbyte (versus 256K for 82595)

For further information on these enhancements and a description of all the differences between the 82595 and 82595TX, please consult the 82595TX User's Manual, available through your local sales representative.

[redacted]

A8678; [A8816;] *see also* A8688.

Dr. Michael Mitzenmacher,[28] an expert witness for USEI, reviewed Intel's internal e-mails and product documentation and determined that Intel "directly reverse

[28] Dr. Mitzenmacher is the Thomas J. Watson Senior Professor (and Former Chair) of Computer Science at Harvard University. A11512:10-A11515:23; A6000-A6021.

engineer[ed] and cop[ied] these technologies from 3Com in an effort to meet and surpass the performance and market success of 3Com products." A14330-A14343. Dr. Thomas Conte,[29] another expert witness for USEI, also reviewed Intel's internal e-mails and product documentation and reached a similar conclusion. A14179-A14189; A14213.

Ultimately, 3Com's competitors, like Intel, were successful at copying 3Com's technology with devastating effect. After growing substantially in the 1990s, widespread infringement had a crippling impact on 3Com's sales of Parallel Tasking products.[30] In 2006, 3Com announced that it would stop selling its network adapter products. A6647:17-A6648:8. On January 31, 2007, 3Com's Network Adapter Division – a division that employed approximately 4,000 people at one time – closed.[31]

Following the closing of 3Com's network adapter division, USEI became the successor-in-interest to 3Com's Parallel Tasking technology.

---

[29] Dr. Conte is a tenured professor at Georgia Institute of Technology and is jointly appointed in the schools of electrical and computer engineering.

[30] A10526:10-A10530:5; A6640-A6641; A6852:1-15.

[31] A6591:23-A6592:16; A6604:13-20 ("Q. When did you eventually leave 3Com? A. January 31st, 2007. Q. What was the state of the connectivity division at the time you left? A. I was the last employee of the connectivity division. Q. Was 3Com still selling products at that time? A. No, I actually announced the end of sale of all products before I left.").

5. The Parallel Tasking Patents Were Found Valid and Infringed by Realtek in the Northern District of California

In 2003, to curb the widespread infringement of its Parallel Tasking technology, 3Com initiated a patent infringement action in the Northern District of California (involving both the '872 and '094 Patents) against Realtek Semiconductor Corporation ("Realtek") and D-Link Systems, Inc. ("D-Link").[32] *3Com Corp. v. Realtek Semiconductor Corp.*, No. 03-02177-VZW (N.D. Cal.). One of the defendant's primary invalidity theories in the Realtek Litigation was based on the NSC 8390, a precursor to the SONIC device asserted here.[33]

Prior to trial, the Honorable Vaughn Walker denied a motion for partial summary judgment virtually identical to the patent marking motion presented by Defendants here:

> [T]he court finds that there are genuine disputes of fact whether 3Com's marking practices amounted to constructive notice of infringement at some point prior to [July 13, 2004]. Partial summary

[32] This suit was part of 3Com's larger licensing program for the Parallel Tasking Patents. A11464:3-A11465:8; A14519:141:1-23; A14970:113:13-22.

[33] Rather than assert the SONIC reference, which was distinguished in the specification of Parallel Tasking Patents, Realtek instead chose to assert its predecessor, the NSC 8390. A9159:37:9-38:4; *see also* A11468:1-A11469:16 (describing commodity market in which most chips were purchased from National Semiconductor). Realtek also asserted a Lare reference similar to the SONIC that also included a 32-byte FIFO. A9149-A9155.

judgment limiting damages is not appropriate for the '459, '872 and '094 patents.[34]

Ultimately, both the district court and the jury rejected Realtek's invalidity argument premised on the NSC 8390 and Lare, which – like the SONIC – had FIFO's that were not large enough to store a minimum-sized Ethernet frame. A6752-A6756; A9149-A9155. The jury found the Parallel Tasking Patents valid and infringed, and the court entered a $45,300,000.00 judgment for 3Com.[35] Shortly thereafter, the Realtek Litigation settled for approximately $70,000,000.00.

B. The Prior Art

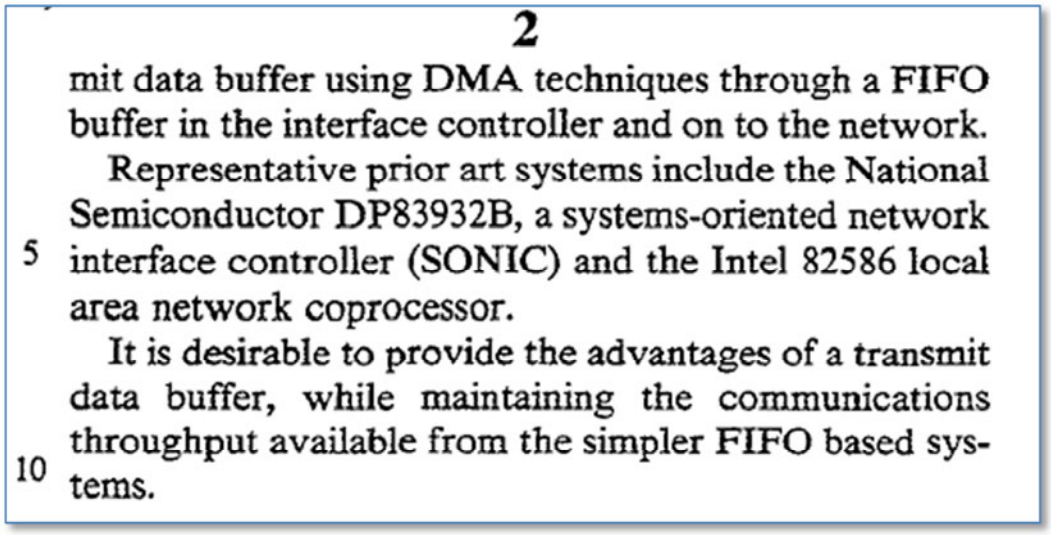
Two basic types of prior art network adapter systems existed at the time 3Com developed the Parallel Tasking technology: (1) those with "dedicated transmit buffers" that were capable of storing the data of a frame "until the [network adapter] ha[s] successfully transmitted the frame"; and (2) those with only simple first-in-first-out ("FIFO") systems, "in which the sending system downloads data of a frame into the FIFO, while the network adapter unloads the FIFO during a transmission."[36] The shared specification of the Parallel Tasking Patents recognizes and discusses these

[34] A6654; *see also* A6653-A6659.

[35] A6752-A6756. The damage award did not include any limitation based on a failure to mark under 35 U.S.C. §287.

[36] A122:1:36-54; *see also* A154:1:35-50. Because the Parallel Tasking Patents have a shared specification, the remaining citations will be solely to the specification of the '872 Patent.

prior art systems in detail,[37] specifically identifying the SONIC as a "[r]epresentative prior art system[]":[38]



2

mit data buffer using DMA techniques through a FIFO buffer in the interface controller and on to the network.

Representative prior art systems include the National Semiconductor DP83932B, a systems-oriented network interface controller (SONIC) and the Intel 82586 local area network coprocessor.

It is desirable to provide the advantages of a transmit data buffer, while maintaining the communications throughput available from the simpler FIFO based systems.

A122:2:3-6.

The specification explains the advantages and disadvantages of both types of prior art systems. For instance, network adapters with a dedicated transmit buffer "enable[d] a sending system to compose and download a frame into the transmit data

---

[37] A122:1:36-54.

[38] A122:2:3-6. The lead inventors were well aware of the SONIC and how it functioned when they described and distinguished it in the specification of the Parallel Tasking Patents. A9163:5-A9165:14; *see also U.S. Ethernet Innovations, LLC v. Texas Instruments Incorporated*, No. 15-1510, 15-1541, Dkt. No. 27 at 17 n.25 (Fed. Cir. July 1, 2015). Moreover, the paid consultant for TI that designed the SONIC confirmed that it is accurately described in the specification. A9159:37:19-38:4; *see also* A14197. Paul Sherer, one of the lead inventors of the Parallel Tasking technology, also testified that he was familiar with the SONIC device and that the SONIC fit the description of the FIFO-based systems in the specification of the Parallel Tasking Patents. A9163:5-A9165:14.

buffer." A122:1:55-57. While the ability to store data of a frame "until the [network adapter] ha[s] successfully transmitted the frame" provided the potential benefit of efficient re-transmission directly from the network adapter in the case of a transmission error (*i.e.*, without having to re-download the data from the host system to the network adapter),[39] prior art systems with a dedicated transmit buffer "suffer[ed] the disadvantage that transmission of a frame is ***delayed until the entire frame has been downloaded into the buffer***." A122:1:58-61 (emphasis added). The delay associated with waiting for the entire frame to be downloaded to the network adapter before initiating transmission is a type of "latency." A11484:5-24; A14305.

In contrast, while systems with prior art FIFOs – like the SONIC – could initiate transmission while still receiving the frame, "[t]he data in FIFOs cannot be retained and reused by the media access control functions, or by the host, like data in transmit data buffers." A122:1:51-54. In particular, systems with a prior art FIFO incapable of storing at least a minimum-sized Ethernet frame – like the SONIC – are forced to "unload" the data from the buffer onto the network to make room in the buffer for the rest of the frame to download.[40] Thus, in the event of an error, the FIFO

[39] A122:1:42-48. The ability to retransmit from the buffer would be a benefit of a system with a buffer that could hold a full Ethernet frame. A14162.

[40] A122:1:36-54; A14163-A14164; A18648:25-A18649:22; A14198-A14199; A14275:49:19-51:12.

system is forced to re-download the frame from the host system before retransmission.[41]

C. The Patented Invention

One of the inventors' primary goals in developing the technology disclosed in the '872 and '094 Patents was to address the latency associated with transmission of data from network adapters with large transmit buffers.[42] Parallel Tasking technology accomplishes this goal by incorporating an early transmit feature into a network adapter that is capable of "storing data of frames … for transmission," as opposed to prior art FIFO-systems that "unloaded" the data during transmission. *See* A122:1:36-54.

As the specification describes, "[t]he present invention provides for the early initiation of transmission of data in a network interface that includes a dedicated transmit buffer." A122:2:13-15. The specification further discloses that a "dedicated transmit buffer" is capable of storing the data of a frame "in the transmit data buffer until the media access control functions associated with transmitting the frame on the

[41] A11570:13-A11571:1; A14163-A14164; A14275:49:19-51:12.

[42] A14198-A14200; A14305-A14306; A10519:7-A10521:10; A14125:157:8-A14126:158:12; A3535-3539.

network have successfully transmitted the frame."[43] Thus, the specification makes clear that the invention would "provide the advantages of a transmit data buffer, while maintaining the communications throughput available from the simpler FIFO-based systems."[44]

The claim language expressly reflects the patented invention's incorporation of the advantages of both prior art systems. For example, Claim 1 of the '872 recites a "buffer memory" that is capable of "storing data of frames … for transmission on the communications medium" [45] and a means for "initiating transmission of the frame prior to transfer of all the data of the frame to the buffer memory from the host computer."[46]

The great weight of testimony from persons of ordinary skill in the art ("POSITAs") that have testified or opined on the Parallel Tasking Patents – including

---

[43] A122:1:39-42 (explaining benefit of transmit data buffer); *see also* A122:2:7-16 (explaining present invention has benefit of transmit data buffer).

[44] A122:2:7-10 ('872 Col. 2:7-10). The district court in the Realtek Litigation also recognized this as the goal of these patents. A2229-A2230. ("The '872 and '094 patents attempt to achieve the advantages of both transmit buffers and FIFO buffers by allowing high throughput as well as efficient retransmission of frames.").

[45] A136:30:7-9.

[46] A136:30:17-22. The same or similar language is also found in the remaining asserted independent claims. A137:32:63-65; A138:33:10-18; A167:28:38-42; A168:29:19-23; A169:31:1-5.

the lead inventors, USEI's experts, and an expert proffered by Intel's joint defense partner – confirms that the patents relate to adapters with transmit buffers that can hold at least a minimum-sized Ethernet frame.[47] Dr. Marwan Hassoun, an expert witness for TI, Intel's joint defense partner, testified specifically that a device cannot meet the buffer memory "for storing data of frames… for transmission" element of the asserted claims unless it can hold a full Ethernet frame:

- **Q.** … Please continue, Dr. Hassoun. You were talking about why the CPMAC can't guarantee holding a full-sized Ethernet frame.

  A. Right. So the CPMAC, as you can see, the blue box here, cannot fit a 1518 full-sized frame or a thousand frame. So fundamentally, if you put this in operation, it will not be able to fit Ethernet frames. The CPMAC will not be able to fit full-sized Ethernet frames.

  Q. And for that reason, in your opinion, does the CPMAC infringe any of the four claims the jury has to consider?

  A. ***No, it does not, because it does not meet the element that has the buffer memory for storing Ethernet frames, that – that element.*** So it doesn't infringe either Claim 1 or Claim 21. And so Claim 5 and Claim 23 also are not infringed, in my opinion.[48]

- "The requirement is that a frame, ***a full-sized frame needs to fit into the buffer*** – into the buffer memory."[49]

---

[47] *See U.S. Ethernet Innovations, LLC v. Texas Instruments Incorporated*, Nos. 15-1510, 15-1541, Dkt. No. 27 at 36-40 (Fed. Cir. July 1, 2015).

[48] A9133:9:21-10:10 (emphasis added). The CPMAC, also known as Communications Processor Ethernet MAC, was a type of Ethernet module incorporated into a group of TI's infringing products.

[49] A9132:7:6-8 (emphasis added).

- "Well, if – if you look at the – if you look at the patent, if you look at the claim, ***you need a buffer memory that can fit – that can store frames, that can store Ethernet frames***."[50]

Dr. Hassoun is far from alone. The lead inventors of the Parallel Tasking technology, Paul Sherer and Brian Peterson, and USEI's expert witnesses, Dr. Mitzenmacher and Dr. Conte, all agreed that a network adapter, as described in the claims and clarified in the specification of the Parallel Tasking Patents, would need a buffer large enough to store at least a minimum-sized Ethernet frame – *i.e.*, 64 bytes.[51] To meet the plain and ordinary meaning of the claim element "storing data of frames … for transmission," the buffer must be capable of holding at least a minimum-sized Ethernet frame.

D. The SONIC Is Not Large Enough to Hold Even a Minimum-Sized Ethernet Frame[52]

The district court correctly found that there is "no dispute that the SONIC device contains a thirty-two byte buffer memory, not big enough to hold the smallest ethernet [sic] data frame, which is sixty-four bytes in size." A69. The district court's

---

[50] A9132:8:9-12.

[51] *See, e.g.,* A14247:7-A14250:25; A9173:1-20; A14206-A14208; A14275:51:13-23; A14305-A14306; A18882-A18883; A8556; *see also U.S. Ethernet Innovations, LLC v. Texas Instruments Incorporated*, Nos. 15-1510, 15-1541, Dkt. No. 27 at 36-40 (Fed. Cir. July 1, 2015).

[52] In the Realtek Litigation, two of Realtek's primary invalidity references were simple FIFO-based systems with 16-byte and 32-byte FIFOs, respectively. A9149-A9155.

finding is confirmed by Wesley Lee,[53] the designer of the SONIC. A18645:18-20; A14196-A14199. Specifically, Mr. Lee conceded that the SONIC, by design, did not include a transmit buffer, but rather transmitted frames from a transmit buffer in the host.[54] The SONIC had only a simple, 32-byte FIFO that is too small to hold even the smallest Ethernet frame.[55] Mr. Lee also conceded that, due to its size, the FIFO in the SONIC had no choice but to begin transmission while still receiving the frame and could not store any data of the frame involved in transmission.[56]

---

[53] TI retained Mr. Lee as a paid consultant shortly before his deposition. Mr. Lee prepared for his deposition by meeting and speaking with counsel for Intel and TI on multiple occasions. During these meetings, counsel for Intel and TI provided him with certain documents, directed him to specific portions of those documents, and discussed the defendants' positions and the substance of Mr. Lee's deposition testimony. A18965:177:6-A18967:182:11; A18967:184:15-185:20; A18968:189:24-191:16.

[54] A11371:18-A11372:22; A9159:37:9-38:4; *see also* A11376:21-A11378:12.

[55] A14198-A14199; A14207-A14208; A9132:7:12-20; *U.S. Ethernet Innovations, LLC v. Texas Instruments Incorporated*, Nos. 15-1510, 15-1541, Dkt. No. 27 at 24 (Fed. Cir. July 1, 2015).

[56] A14198-A14199; *see also* A11371:18-A11372:22; A9163:5-A9165:14 (explaining how prior art FIFO-systems only moved data from one part of system to another). Mr. Lee testified that he believed at the time that including a larger buffer on the SONIC would have been "cost prohibitive" without adding any tangible benefit to performance. *See* A14196-A14199; *see also U.S. Ethernet Innovations, LLC v. Texas Instruments Incorporated*, Nos. 15-1510, 15-1541, Dkt. No. 27 at 25 (Fed. Cir. July 1, 2015). In fact, Mr. Lee conceded on the stand that he was of the opinion that it would not have made sense to include a larger FIFO. A14198-A14199.

E. 3Com and Its Licensees Specifically Authorized to Practice the '872 Patent Marked Their Products

The undisputed evidence demonstrates that 3Com consistently and continuously marked substantially all of its products practicing the '872 Patent. The evidence of 3Com's marking practices includes: (1) Annette Davis' sworn testimony that it was 3Com's practice to mark its products practicing the '872 Patent; (2) Richard Baker's chart and related testimony showing products for which 3Com located a specific exemplar of marking in 2006; (3) product samples, labels, and software license agreements for 3Com products with '872 Patent markings; and (4) verified interrogatory responses listing products marked by year.[57]

The undisputed evidence also demonstrates that the only three licensees specifically authorized to practice the '872 Patent – Realtek, VIA, and ADMtek[58] – were required to mark their products[59] and that 3Com took reasonable steps to ensure their compliance.[60]

---

[57] *See, e.g.*, A6623; A5058:19-A5059:10; A5065:19-22; A6749-A6750; A6684:34:17-A6686:42:4; A6912-A6969; A6974-A6975.

[58] A6819-A6837; A6798-A6817; A6758-A6767.

[59] A6829; A6758-A6759; A6802-A6803. *See also* A6769; A6771-A6772; A6873-A6884; A6862:8-19.

[60] A6769; A6771-A6772; A6862:8-19; A6873-A6884.

F. The Broad, Portfolio-Wide Agreements Involving IBM and Intel Did Not Directly Relate to the Parallel Tasking Patents

In 1999, 3Com entered into a cross-license agreement with IBM. A6979; A6996. Unlike the Realtek, VIA, and ADMtek license agreements, the broad, portfolio-wide cross-license between 3Com and IBM did not resolve a dispute involving the Parallel Tasking Patents or otherwise directly relate to those patents. *See generally* A6979-A6998. Rather, the goal of the agreement was simply to prevent patent disputes between the parties in the future. A7073.

The IBM Agreement does not specifically reference any products or patents.[61] *See generally* A6979-A6998. As stated by IBM, "IBM was selling tens of thousands of unique products spanning various fields of technology. It would have been impractical to perform an appropriate analysis to determine which particular patents of 3Com, if any, were practiced by any of IBM's products." A7073-A7074. Moreover, there is no evidence indicating that either IBM or 3Com believed that IBM was practicing any of the Parallel Tasking Patents at the time of the IBM Agreement.

Similarly, 3Com entered into a broad, two-year, portfolio-wide mutual covenant not to sue with Intel on November 8, 2002. A7394-A7411. The Intel CNS was executed in connection with a larger agreement that resolved litigation between 3Com

[61] The parties never discussed any IBM products as licensed or authorized to practice any claims of the Parallel Tasking Patents. A7074.

and Xircom, Inc., a company acquired by Intel almost a year after the litigation was filed. *See* A7079-A7096. The 3Com/Xircom litigation did not involve any of the Parallel Tasking Patents, any Intel products, or even Ethernet technology.[62]

The Intel CNS does not specifically reference any products or patents. (*See generally* Intel CNS.) During the negotiation of the Intel CNS, the parties never discussed the Parallel Tasking Patents. A7309:14-A7312:10. The goal of the Intel CNS (like the IBM Agreement) was simply patent "peace," not the specific authorization to sell products under certain patents. A10874:17-A10875:3; A7315:17-23.

V. SUMMARY OF THE ARGUMENT

The SONIC reference, a prior art system expressly disclosed, described, and distinguished in the specification of the Parallel Tasking Patents, cannot anticipate the Parallel Tasking Patents as a matter of law. As evidenced by numerous sworn statements and opinions in the record, POSITAs would understand that for a network adapter to "stor[e] data of frames … for transmission," as described in the claims and clarified in the specification of the Parallel Tasking Patents, it would need a buffer memory large enough to store at least a minimum-sized Ethernet frame – *i.e.*, 64 bytes. It is undisputed that the SONIC's 32-byte FIFO is, by design, not large enough

[62] *See generally* A7079-A7096; *see also* A7301:14-A7303:6; A7304:15-A7305:23; A7307:22-A7308:6.

to store even a minimum-sized Ethernet frame, so it cannot anticipate the asserted claims as a matter of law. Accordingly, the district court erred in denying USEI's motion for summary judgment that the Parallel Tasking Patents are not invalid as a matter of law in view of the SONIC prior art reference and in granting Defendants-Appellees' motion for summary judgment of invalidity based on this same reference.

The district court also committed error in denying USEI's motion for summary judgment that Intel's unclean hands, demonstrated through the intentional copying of the Parallel Tasking technology, prevents the application of any of Intel's asserted equitable defenses, including laches. In rejecting USEI's motion, the district court determined that the overwhelming evidence of intentional copying was "irrelevant" based upon its flawed finding that USEI had "fail[ed] to provide ***any*** evidence" that the 3Com product copied by Intel, the EtherLink III, embodied the asserted claims. This finding was in error, as the record contains substantial evidence that the EtherLink III embodied the Parallel Tasking Patents, and the Court should therefore reverse the lower court's order denying USEI's motion.

The district court further erred in granting Defendants-Appellees' motion for partial summary judgment that the patentee failed to comply with the marking statute. The district court's order is premised entirely on two broad, portfolio-wide agreements not directly related to the Patents-in-Suit and aimed only at achieving "patent peace" for the parties. The order is flawed for several reasons: (1) the undisputed evidence

shows that 3Com and its licensees specifically authorized to practice the '872 Patent marked their products throughout the relevant period; (2) 3Com made reasonable efforts to ensure compliance with the marking provision by its licensees; (3) viewing the evidence in the light most favorable to USEI, one cannot conclude that patented articles were sold unmarked by a 3Com licensee after April 30, 2005; and (4) IBM and Intel were not acting within their licensing rights. As a result, the district court's sweeping relief is fatally flawed.

VI. ARGUMENT

A. Standard of Review

This Court reviews a district court's grant of summary judgment without deference. *Dynacore Holdings Corp. v. U.S. Philips Corp.*, 363 F.3d 1263, 1273 (Fed. Cir. 2004). "[S]ummary judgment will not lie if the dispute about a material fact is 'genuine,' that is, if the evidence is such that a reasonable jury could return a verdict for the nonmoving party." *Anderson v. Liberty Lobby*, 477 U.S. 242, 248 (1986). The burden of establishing the absence of a genuine issue of material fact lies with the moving party. *Celotex Corp. v. Catrett*, 477 U.S. 317, 322-23 (1986). Moreover, "[t]he evidence of the nonmovant is to be believed, and all justifiable inferences are to be drawn in favor of the nonmovant." *Cybiotronics, Ltd. v. Golden Source Elecs., Ltd.*, 130 F. Supp. 2d 1152, 1156 (C.D. Cal. 2001).

B. The Parallel Tasking Patents Are Valid over the SONIC as a Matter of Law

Any argument that the SONIC anticipates the Parallel Tasking Patents must fail, as it rests on a flawed view of the patents that is directly contradicted by the overwhelming evidence in the record. A POSITA cannot read the asserted claims of the Parallel Tasking Patents in light of the specification and determine that they provide for nothing more than the simple, FIFO-based SONIC device specifically distinguished in the specification. Accordingly, the Court should find that the Parallel Tasking Patents are valid over the SONIC as a matter of law.

The record in this case demonstrates that the district court's order is flawed for several reasons: (1) the order fails to give any deference to the United States Patent and Trademark Office's ("PTO") decision to issue the patents over the SONIC, a reference that was disclosed, described accurately and distinguished in the specification; (2) the order fails to consider that Intel admitted that the technology was "novel"; (3) the order directly contradicts the great weight of testimony from POSITAs; (4) the order fails to consider critical claim language; (5) the order is premised on an unreasonable reading of the specification; and (6) the order ignores the validity determinations made by two previous courts and juries.[63]

[63] Notably, the district court's invalidity determination directly contradicts a previous decision on summary judgment in the TI Action, as well as the jury verdict in that case, based upon the exact same reference, facts, and arguments presented by Intel's

1. Defendants Cannot Meet Their High Burden of Proof of Establishing Invalidity by Clear and Convincing Evidence

A patent and each of its claims are presumed valid. 35 U.S.C. §282. The party asserting an invalidity defense bears the high burden of presenting clear and convincing evidence to overcome a patent's presumption of validity. *Microsoft Corp. v. i4i Ltd. P'ship*, __ U.S. __, 131 S. Ct. 2238, 2245 (2011); *Canon Computer Sys. v. Nu-Kote Int'l*, 134 F.3d 1085, 1088 (Fed. Cir. 1998) ("where the challenger fails to identify any persuasive evidence of invalidity, the very existence of the patent satisfies the patentee's burden on the validity issue").

When a prior art reference is disclosed in the specification, deference is given to the PTO's decision to issue the patent notwithstanding that prior art. *Sciele Pharma, Inc. v. Lupin Ltd.*, 684 F.3d 1253, 1260 (Fed. Cir. 2012) ("there is a 'high burden of proof created by the necessary deference to the PTO'") (citation omitted); *SciMed Life Sys. v. Advanced Cardiovascular Sys.*, 242 F.3d 1337, 1343 (Fed. Cir. 2001) ("discussion [distinguishing the prior art] in the written description supports the district court's conclusion that the claims should not be read so broadly as to

---

joint defense partners, TI. *Compare* A68-A73 *with* A18527-A18533; *see also* A18535-A18537; *U.S. Ethernet Innovations, LLC v. Texas Instruments Incorporated*, 15-1510, 15-1541, Dkt. No. 27 at 26-27 (Fed. Cir. July 1, 2015).

encompass the distinguished prior art structure"); *Minn. Mining & Mfg. Co. v. Johnson & Johnson Orthopaedics, Inc.*, 976 F.2d 1559, 1572 (Fed. Cir. 1992).

In *Endo Pharm.*, the United States District Court for the District of Delaware addressed whether a prior art reference disclosed in the patent (and thus considered by the PTO) anticipated claims of the patent at issue. *Endo Pharm. Inc. v. Mylan Pharm. Inc.*, No. 11-CV-00717 (RMB/KW), 2014 U.S. Dist. LEXIS 10037, at *33-*34 (D. Del. Jan. 28), *vacated on other grounds*, 2014 U.S. Dist. LEXIS 76251 (D. Del. Apr. 8, 2014). The court held that the defendants had not established anticipation by clear and convincing evidence, finding the testimony of plaintiff's expert persuasive, and giving deference to the PTO's decision to issue the patent over the reference that had been disclosed to it. *Id.* at *41-*43 (citing with approval *Minn. Mining*, 976 F.2d at 1572).

Here, the patent examiner's determination that the Parallel Tasking Patents are valid over the SONIC is entitled to deference.[64] The description of the SONIC in the specification is accurate. A9159:37:19-38:4; *see also* A14197. The patent examiner, a POSITA, thus had an accurate description of the SONIC when considering whether to issue the patents, and allowed them to issue. *See V-Formation, Inc. v. Benetton Grp. SpA*, 401 F.3d 1307, 1311 (Fed. Cir. 2005). This decision is entitled to

[64] *See supra* §IV.B-C.

deference. *Endo Pharm.*, 2014 U.S. Dist. LEXIS 10037, at *41-*41, *vacated on other grounds*, 2014 U.S. Dist. LEXIS 76251 (citing with approval *Minn. Mining*, 976 F.2d at 1572).

In light of all of the evidence in this case, as well as the deference that must be given to the PTO's decision to issue the Parallel Tasking Patents over the SONIC, Defendants cannot meet their high burden of proving that the SONIC invalidates the Parallel Tasking Patents by clear and convincing evidence as a matter of law.

2. The Overwhelming Evidence in the Record Demonstrates that the SONIC Cannot Anticipate the Parallel Tasking Patents

a. Intel Admitted that Parallel Tasking Technology Was Novel in 1992

As an initial matter, the district court's order fails entirely to consider Intel's admission that Parallel Tasking technology was "novel." A8656. Internal Intel e-mails and product documentation demonstrate conclusively that 3Com's Parallel Tasking technology represented a significant step forward, [ ] A8659. Intel's internal documents also show that it intentionally copied the technology, including the early transmission feature, and implemented it into its products.[65] Intel's admission directly contradicts the district court's determination that the Parallel Tasking Patents represented nothing more than

[65] *See supra* §IV.A.4.

the simple, FIFO-based SONIC device widely available at the time. *Cf. Checkpoint Sys. v. All-Tag Sec. S.A.*, 711 F.3d 1341, 1347 (Fed. Cir. 2013) (holding party may rely on admission by party opponent), *vacated on other grounds*, *Kobe Props. Sarl v. Checkpoint Sys.*, __ U.S. __, 134 S. Ct. 2134 (2014).

b. The District Court's Order Fails to Consider the Great Weight of Evidence from Persons of Ordinary Skill in the Art

When considering a question of validity, the appropriate test is what a POSITA would conclude in comparing the language of the claims to the prior art. *See AstraZeneca LP v. Apotex, Inc.*, 633 F.3d 1042, 1055 (Fed. Cir. 2010) (citing *In re Baxter Travenol Labs*, 952 F.2d 388, 390 (Fed. Cir. 1991)); *Sundance, Inc. v. DeMonte Fabricating Ltd.*, 550 F.3d 1356, 1361-63 (Fed. Cir. 2008).

The evidence in the record demonstrates that a POSITA, when comparing the SONIC to the claim elements, would conclude that the SONIC does not contain the claim element "storing data of frames … for transmission." A136:30:7-8 (Claim 1); *see also* A137:32:63-64 (Claim 21). The FIFO in the SONIC is incapable of "storing data of frames … for transmission" because, unlike a transmit data buffer, simple FIFO devices – like the SONIC – that are not large enough to store at least a minimum-sized Ethernet frame are forced to "unload" the data during transmission.[66]

[66] *See supra* §IV.D.

This understanding of the asserted claims is remarkably consistent among POSITAs that have considered the issue.[67]

Dr. Marwan Hassoun, proffered as a POSITA by Intel's joint defense partner, TI, conceded that he understood the asserted claims to require a buffer memory large enough to store an Ethernet frame.[68] Dr. Hassoun's understanding of this requirement was based upon his review of the claim language in light of the specification. A9132:8:7-12.

Dr. Hassoun's conclusions are not unique. The great weight of testimony by POSITAs has confirmed that "storing data of frames … for transmission" requires a buffer memory large enough to store at least one minimum-sized Ethernet frame.[69] Indeed, POSITAs proffered by other alleged infringers have agreed with his conclusions.[70] USEI's expert witnesses also confirmed that the asserted claims of the

---

[67] *U.S. Ethernet Innovations, LLC v. Texas Instruments Incorporated*, Nos. 15-1510, 15-1541, Dkt. No. 27 at 36-40 (Fed. Cir. July 1, 2015).

[68] *See supra* §IV.C at pp. 27-28.

[69] *See supra* §IV.C.

[70] A18882-A18883 ("Therefore, it is my opinion that one of ordinary skill in the art would understand that the claimed ***buffer memory*** must be able to store an entire frame."). The validity of the Parallel Tasking Patents over the SONIC reference is also at issue before this Court in the TI appeal. *U.S. Ethernet Innovations, LLC v. Texas Instruments Incorporated*, Nos. 15-1510, 15-1541, Dkt. No. 27 (Fed. Cir. July 1, 2015). For a discussion of the near consensus opinion of POSITAs in that case, see

invention require a buffer memory large enough to store at least one minimum-sized Ethernet frame. Dr. Mitzenmacher opined as follows: "Adapters practicing this technology contain a buffer that is capable of storing at least a minimum-sized Ethernet frame …." A14305-A14306. Dr. Thomas Conte opined as follows: "[T]he buffer of the invention must be capable of holding a full Ethernet frame." A14206-A14208.

Moreover, the lead inventors of the Parallel Tasking technology confirmed their invention requires a buffer memory large enough to store at least one minimum-sized Ethernet frame.[71] *Phillips v. AWH Corp*., 415 F.3d 1303, 1316 (Fed. Cir. 2005) (en banc) ("'Ultimately, the interpretation to be given a term can only be determined and confirmed with a full understanding of what the inventors actually invented and intended to envelop with the claim.'") (quoting *Renishaw PLC v. Marposs Societa Per Azioni*, 158 F.3d 1243, 1250 (Fed. Cir. 1998)). For example, Paul Sherer testified:

> **Q:** … And so – and with respect to the '872 patent – excuse me – the buffer would have to be big enough to hold at least one Ethernet frame; is that right?

U.S. Ethernet Innovations, LLC v. Texas Instruments Incorporated, Nos. 15-1510, 15-1541, Dkt. No. 27 at 36-40 (Fed. Cir. July 1, 2015).

[71] *See Power Integrations, Inc. v. Fairchild Semiconductor Int'l, Inc.*, 711 F.3d 1348, 1360 (Fed. Cir. 2013) (confirming relevance of inventor testimony as extrinsic evidence in claim construction), *cert. denied*, __ U.S. __, 134 S. Ct. 900 (2014).

> **THE WITNESS:** Yes.[72]

Brian Petersen confirmed as follows:

> **Q:** … So to your recollection, the – the – the transmit buffer, as described in the '872 patent, would have to at least be able to hold one Ethernet frame?
>
> **THE WITNESS:** Yes.[73]

In addition, the inventors[74] and experts for both sides[75] have discussed the benefits that flow from the ability to store an Ethernet frame. The district court's order fails to consider any of this critical testimony.

c. This Understanding of Persons of Ordinary Skill in the Art Is Supported by Overwhelming Evidence

Given the evidence in the record, it is not surprising that the POSITAs would conclude that the buffer memory must be large enough to store at least a minimum-sized Ethernet frame to "stor[e] data of frames … for transmission."

---

[72] A14247:7-A14250:18 (objections omitted); *see also* A9163:5-A9165:14 (noting that prior art FIFO's simply shuttle data from one part of a system to another with no notion of the larger construct, while the inventions of the '872 and '094 Patents were more focused on managing data transfers as part of the transmission process); A14198-A14199 (explaining that prior art FIFOs served the purpose of matching speed differences between the host and network).

[73] A9173:10-20 (objection omitted); *see also* A9175:12-A9176:5.

[74] *E.g.*, A14247:7-A14250:25; A9173:10-20.

[75] *E.g.*, A18882-A18883; *see also* A14206-A14208; A14305-A14306.

One of the primary problems that the inventors set out to solve in developing the Parallel Tasking technology was to reduce the delay or "latency" associated with data transmission from a transmit buffer.[76] The Parallel Tasking Patents accomplished this by allowing initiation of transmission of data of frames prior to the transfer of all of the data of the frame to the buffer.[77] Importantly, because the SONIC has a small FIFO by design, it does not suffer from the latency issues of a transmit data buffer and does not address the problems solved by the Parallel Tasking technology.[78] The district court's failure to consider the underlying purpose of the patented invention contributes to its flawed analysis.

Furthermore, each of the asserted claims require a buffer memory "for storing data of frames … for transmission."[79] The specification for the Parallel Tasking

---

[76] *See supra* n.42.

[77] *Id.*

[78] A14198-A14200; A14207-A14208; A14125:157:8-A14126:158:12; A9132:7:12-20; A11371:18-A11372:22; A11376:21-A11378:12; A9159:37:9-38:4.

[79] Each of the asserted claims includes the same or similar language. *See supra* §IV.C at n.46. The district court's order does not mention this claim language in its analysis. A68-A73. Noting only that the term "'buffer memory'" has been construed as "'memory for the temporary storage of data,'" the lower court fails to consider the rest of the claim element. A70-A71.

Patents states that "[t]he present invention … includes a dedicated transmit buffer."[80] The specification further explains that one of the ***advantages*** of a "dedicated transmit buffer" is its capability of storing the data of a frame "in the transmit data buffer until the media access control functions associated with transmitting the frame on the network have successfully transmitted the frame."[81] The specification further provides as follows:

> It is desirable to provide the advantages of a transmit data buffer, while maintaining the communications throughput available from the simpler FIFO based systems.[82]

A POSITA would understand that in order to store the data of a frame until it has been "successfully transmitted," the buffer memory must be large enough to store at least a minimum-sized Ethernet frame – *i.e.*, 64 bytes.[83]

---

[80] A122:2:13-15. The specification defines a "dedicated transmit buffer" as having the capability of downloading a full frame of data from the sending system to allow the sending system to attend to other tasks while the network adapter attempts to transmit the frame. A122:1:55-60.

[81] A122:1:39-42 (explaining advantage of transmit data buffer as being able to store a frame until successful transmission); *see also* A122:2:7-16 (explaining the present invention has the advantage of a transmit data buffer and the early initiation throughput of a FIFO).

[82] A122:2:7-10.

[83] *See supra* §IV.C, §VI.B.1.b. The district court's determination that the specification "criticized the use of full frame capacity" and considers it to be a "disadvantage[]" of transmit data buffers is further evidence that it misunderstood the patented technology. A70-71.

The inventors expressly disclosed, accurately described, and distinguished the SONIC and its simple FIFO-based system from their invention in the specification. There are also numerous other references in the claims and specification supporting a POSITAs understanding that the buffer memory must be large enough to store at least a minimum-sized Ethernet frame to "stor[e] data of frames … for transmission."[84] For example, Claim 1 of the '872 Patent recites a "means, responsive to the threshold determination of the means for monitoring, for initiating transmission of the frame prior to transfer of ***all of the data of the frame*** to the buffer memory from the host computer."[85] If the buffer memory were smaller than a minimum-sized Ethernet frame, the claim language "prior to transfer of ***all the data of the frame*** to the buffer memory" would be superfluous. *See Bicon, Inc. v. Straumann Co.*, 441 F.3d 945, 950 (Fed. Cir. 2006) ("claims are interpreted with an eye toward giving effect to all terms in the claim"). If the buffer memory were smaller than a minimum-sized Ethernet frame, there would be no need for claim language stating that transmission can occur

[84] Significantly, Mr. Petersen later confirmed his understanding that a buffer of less than 64 bytes – or the size of a minimum Ethernet frame – would not be capable of providing the benefit of retransmission from the buffer in the event of a transmission error. A9175:12-A9176:5.

[85] A136:30:17-22 (emphasis added). The same or similar language is found in each of the asserted independent claims, all indicating that the buffer must have the capability of storing at least a minimum-sized Ethernet frame. A138:33:10-18; A167:28:37-42; A168:29:19-23; A169:31:1-5.

before the buffer memory receives all of the frame because, like the SONIC, the network adapter would be forced to unload the data from the buffer memory to make room for the rest of the frame.[86] The specification also repeatedly confirms that transmission is triggered when either (1) the early transmit threshold has been reached or (2) the entire frame has been transferred to the adapter.[87]

Because it is undisputed that the SONIC's buffer memory – a 32-byte FIFO – is, by design,[88] too small to store at least a minimum-sized Ethernet frame, the SONIC cannot "stor[e] data of frames … for transmission." A69; A9132:7:12-20. Accordingly, the SONIC cannot anticipate the asserted claims of the Parallel Tasking Patents as a matter of law.

C. The District Court's Order Regarding Intel's Intentional Copying Should Be Reversed

Intel's unclean hands, demonstrated through the intentional copying of 3Com's patented technology, prevent the application of any of its asserted equitable defenses, including laches. *A.C. Aukerman Co. v. R.L. Chaides Constr. Co.*, 960 F.2d 1020, 1033 (Fed. Cir. 1992). As an equitable defense, laches is inapplicable if the infringer "'has engaged in particularly egregious conduct which would change the equities

---

[86] *See supra* n.40.

[87] A130:18:31-37; A136:29:28-38; A132:21:31-36; A134:25:3-10; A124:5:5-13.

[88] *See supra* §IV.D.

significantly in plaintiff's favor.'" *Id.* (citation omitted). "Conscious copying may be such a factor weighing against the defendant …." *Id.*; *see also Gasser Chair Co. v. Infanti Chair Mfg. Corp.*, 60 F.3d 770 (Fed. Cir. 1995) (reversible error not to preclude a laches defense in case of willful copying); *Bott v. Four Star Corp.*, 807 F.2d 1567, 1576 (Fed. Cir. 1986) (finding copying is egregious conduct defeating defense of laches); *Carnegie Mellon Univ. v. Marvell Tech. Group, Ltd.*, No. 09-290, 2014 U.S. Dist. LEXIS 4624, at *131 (W.D. Pa. Jan. 14, 2014). Even assuming that Intel can make out the elements of any equitable defense – which it cannot – its intentional copying of the Parallel Tasking technology places the equities in USEI's favor.

1. USEI Presented Overwhelming Evidence that Intel Intentionally Copied the Patented Technology

The evidence that Intel intentionally copied 3Com's Parallel Tasking technology and incorporated it into its products is overwhelming. Unrebutted evidence shows that "Parallel Tasking" referred to the technology in the '872 and '094 Patents, that Parallel Tasking technology was first embodied in the EtherLink III product, and that the EtherLink III was released in 1992. Internal Intel e-mails and product documentation from late 1992 and 1993 further show:[89]

[89] Evidence of copying of a claimed invention may be found in internal documents, such as Defendants' emails. *See Akamai Techs. v. Cable & Wireless Internet Servs.*, 344 F.3d 1186, 1196-97 (Fed. Cir. 2003) ("[The defendant's] redesign process was

- [redacted]
- [redacted]]
- [redacted]]
- [redacted]]

See supra §IV.A.4. Intel datasheets demonstrate:

- Intel's inclusion of the copied functionality into new products; and
- Intel describing the copied functionality as "Concurrent Processing."

See supra §IV.A.4. Moreover, experts reviewed the Intel e-mails and product documentation, ultimately concluding that Intel "directly reverse engineer[ed] and cop[ied] these technologies from 3Com in an effort to meet and surpass the performance and market success of 3Com products." A14330-A14343.

The district court, however, determined that the overwhelming evidence of Intel's intentional copying was "irrelevant" because it found that "USEI fail[ed] to provide ***any evidence*** to show that the 'Parallel Tasking' EtherLink III adapter embodies each and every element of the asserted claims." A74 (emphasis added).

---

documented in the record in internal emails from [the defendant's] engineers discussing [the plaintiff's] approach, identifying weaknesses in [the defendant's] approach, and ultimately deciding to switch to the [plaintiff's] system.")

The court points to the absence of expert testimony to support this finding. The Court's conclusion, however, is flawed both legally and factually.[90]

2. There Is Substantial Evidence that 3Com's Etherlink III Adapter Embodied the Parallel Tasking Patents

The two lead inventors, Brian Petersen and Paul Sherer, testified unequivocally that the EtherLink III practiced the asserted patents.[91] Scott Emery, a co-inventor on another 3Com patent with Mr. Petersen and Mr. Sherer, who also worked on the EtherLink III and its predecessor,[92] also testified that the same early transmit technology embodied in the '872 and '094 Patents was included in the EtherLink III:

> **Q:** Did you ever hear of the phrase that was used by 3Com at one point, "parallel tasking"?
>
> **A:** Yes.
>
> **Q:** And what's your understanding of that phrase?
>
> **A:** That's a marketing term used to describe some technologies that were included in the EtherLink III adapters.

---

[90] To the extent that the district court determined that USEI was required to provide expert testimony that the EtherLink III adapter practiced the asserted claims, it misunderstands the law. *See Morpho Detection, Inc. v. Smiths Detection, Inc.*, 957 F. Supp. 2d 655, 674 n.22 (E.D. Va. 2013) (rejecting the argument that the patentee through its expert, was required by law to provide the equivalent of a claim by claim infringement analysis to show that the patentee's product was an embodiment of its own patent).

[91] A14487:1-16; A14489:16-19; A11479:13-19.

[92] A15106:58:17-59:8.

> **Q:** And what kind of technologies were – in your understanding – does the marketing term refer to?
>
> **A:** So to me, the fundamental technology that that refers to is the ability for the adapter, the network interface card, to receive a frame and simultaneously transfer it across the host bus, or to transmit a frame by transferring it across from the CPU – from the host memory to the adapter memory, and simultaneously begin transmitting it on the network.[93]

Numerous former 3Com employees provided unrebutted testimony confirming that "Parallel Tasking" referred to, *inter alia*, the '872 and '094 Patents.[94] 3Com 10-Ks from the period state, "All 3Com EtherLink and Fast EtherLink adapters feature 3Com's ***patented*** Parallel Tasking® architecture." A14513 (emphasis added). A product brief from 3Com for the EtherLink III adapter explained that "Parallel Tasking" technology allowed the EtherLink III to "begin[] writing the first portion of a data frame onto the wire before it has completed reading the frame from PC memory." A14476.

The lower court was also presented with unrebutted evidence that 3Com's development project codenamed "Vulcan"[95] ultimately resulted in the EtherLink III adapter and that the Vulcan project embodied at least the Parallel Tasking Patents:

---

[93] A15133:167:18-168:8.

[94] *See supra* n.18.

[95] *See, e.g.*, A6630-A6633; A6635-A6638; A6625-A6628; A15106:58:17-59:8; A11472:2:21; A14226:20-23; A14251:16-21.

> **Q:** … And I believe you testified before that following these continuous and diligent efforts to develop this technology, that the – the Vulcan project actually produced a product which was called the EtherLink III; is that right?
>
> **A:** Yes.

A14251:16-21; *see also supra* §IV.A.2-3.

Because the lower court's order is premised upon a flawed understanding of the facts and the law, the court should reverse the lower court's order and remand for re-consideration of USEI's motion.

D. The District Court Erred in Granting Partial Summary Judgment on the Basis of Marking

The district court erred in granting Defendants' motion for summary judgment regarding marking because a reasonable jury could (and should) return a verdict in USEI's favor on this issue.[96] Whether a patentee has complied with 35 U.S.C. §287 is a question of fact for the jury. *Maxwell v. J. Baker, Inc.*, 86 F.3d 1098, 1111 (Fed. Cir. 1996). In the Realtek Litigation (involving the '872 Patent), Judge Walker denied a motion for summary judgment on the issue of marking because a reasonable jury ***could find*** that 3Com complied with its marking obligations (A6653-A6654), and

[96] *See, e.g.*, *Sentry Prot. Prods. v. Eagle Mfg. Co.*, 400 F.3d 910, 918 (Fed. Cir. 2005) (reversing a district court's grant of summary judgment where the patentee "came forth with an affidavit stating that its products were marked, together with documents showing sales in that period," and finding that evidence was "not so 'conclusory' or 'lacking in factual support' as to allow us to affirm the grant of summary judgment") (citation omitted).

later, the jury ***found*** that 3Com complied with its marking obligations (A6752-A6656). Likewise, the jury in the TI Action awarded damages in a manner consistent with a finding that 3Com complied with its marking obligations. A18539-A18542.

This Court has explained that "[i]n cases where a patent owner sells a product embodying the invention, a patent owner may seek damages for infringement occurring during a period in which the accused infringer had actual or constructive notice of the asserted patent." *Sentry*, 400 F.3d at 918. "Constructive notice is provided 'when the patentee consistently marks substantially all of its patented products.'" *Id.* (quoting *Am. Med. Sys. v. Med. Eng'g Corp.*, 6 F.3d 1523, 1538 (Fed. Cir. 1993)); *see also Maxwell*, 86 F.3d at 1111-12.

Here, there is no genuine dispute that 3Com properly marked its products with the '872 Patent.[97] USEI presented undisputed evidence showing that 3Com consistently and continuously marked substantially all of its ***billions*** of dollars of Parallel Tasking products practicing the '872 Patent, beginning in 1997 and continuing through the filing of this litigation and that 3Com and USEI required the only three licensees specifically authorized to practice the '872 Patent – Realtek, VIA,

[97] 35 U.S.C. §287 does not apply to the '094 Patent because it contains only method claims. *See, e.g.*, *State Contracting & Eng'g Corp. v. Condotte Am., Inc.*, 346 F.3d 1057, 1073 (Fed. Cir. 2003) ("It is 'settled in the case law that the notice requirement of this statute does not apply where the patent is directed to a process or a method.'").

and ADMtek – to do the same.[98] Defendants failed to produce any evidence disputing these facts. Indeed, the district court's order relied exclusively on the alleged sale of "patented articles" by IBM and Intel under broad, portfolio-wide cross "licenses" that were not directly related to the Patents-in-Suit and that were aimed only at buying patent "peace" for the parties' agreements.

The district court's order is flawed for several reasons.[99]

1. 3Com Made Reasonable Efforts to Ensure Compliance with the Marking Provision by Its Licensees

When the alleged failure to mark is caused by someone other than the patentee, the question becomes "whether the patentee made ***reasonable efforts*** to ensure compliance with the marking requirements." *Maxwell*, 86 F.3d at 1111-12 (emphasis added); *see also Acticon Techs. LLC v. Creative Labs Inc.*, No. 2:08-cv-194-RSP, 2012 WL 708059, at *3 (E.D. Tex. Mar. 5, 2012). The fact-intensive "rule of reason" test is "'consistent with the purpose of the constructive notice provision – to encourage patentees to mark their products in order to provide notice to the public of the existence of the patent and to prevent innocent infringement.'" *Funai Elec. Co.,*

---

[98] *See supra* §IV.E.

[99] Contrary to Defendants' assertion below, the parties to the Realtek Litigation were aware of the IBM license agreement and the Intel covenant not to sue. *See, e.g.*, A6704:114:8-A6705:118:2.

Ltd., v. Daewoo Elecs. Corp., 616 F.3d 1357, 1375 (Fed. Cir. 2010) (citing *Maxwell*, 86 F.3d at 1111-12).

This appeal appears to present an issue of first impression. Specifically, whether the "rule of reason" test under the marking provision requires a patentee that has entered into a broad, portfolio-wide cross-license to undertake the impossible task of analyzing tens of thousands of products to determine whether they practice any of the thousands of patents in its portfolio. At the district court level, Defendants failed to identify a single opinion in which the sale of "patented articles" under a broad, portfolio-wide cross-license (or covenant not to sue) was imputed to the patentee for purposes of limiting the recovery of damages under §287. Moreover, sound policy counsels against placing such an unreasonable burden on prospective cross-licensees.

a. The District Court's Order Applies an Unreasonable Standard

A test based upon "reasonableness" cannot set an impossible bar. Prospective cross-licensees cannot comply with a rule that requires them to analyze tens of thousands of products against each of the thousands of patents in their respective portfolios – an impossible task.[100] The view of the "rule of reason" test adopted by the district court here effectively requires a patentee that desires to enter into a broad,

[100] It would have been impossible for 3Com to analyze tens of thousands of IBM or Intel products against each patent in 3Com's portfolio, as acknowledged by IBM. A7073-A7075.

portfolio-wide cross-license with another company to forego pre-suit damages for every patent in its portfolio in any future litigation. Adopting such a rule will discourage the negotiated resolution of patent disputes at great cost to litigants and to judicial resources. Moreover, because it sets an impossible bar in these situations, it will not encourage marking to provide notice to the public – the very purpose of the constructive notice provision. *Funai*, 616 F.3d at 1375.

Here, the evidence demonstrates that 3Com and USEI made ***reasonable efforts*** to ensure compliance with the marking provision by their licensees. It is undisputed that 3Com and USEI made reasonable efforts to ensure that its licensees specifically authorized to practice the '872 Patent – Realtek, VIA, and ADMtek –complied with the marking provision.[101]

Unlike the Realtek, VIA, and ADMtek license agreements, the broad, portfolio-wide agreements involving IBM and Intel did not directly relate to the Parallel Tasking Patents or to disputes involving these patents.[102] Instead of specifically authorizing the sale of products under certain patents, the goal of the agreements was simply to prevent patent disputes between the parties in the future.[103]

---

[101] *See supra* §IV.E.

[102] *See supra* §IV.F.

[103] *See supra* §IV.F.

Under these circumstances, the district court erred when it determined that 3Com and USEI failed as a matter of law to make "reasonable efforts" – a "fact intensive" inquiry – to ensure compliance with the marking provision by its licensees. At the very least, there is a fact issue as to whether it would be reasonable to impute to 3Com any sales of products under these agreements.[104]

b. The District's Ruling Is Unprecedented in Case Law

In its order, the district court relies exclusively upon *WiAV Sols. LLC v. Motorola, Inc.*, 732 F. Supp. 2d 634 (E.D. Va. 2010), to support its application of the "rule of reason" test to the facts presented here. The court's reliance upon this case is misplaced.[105]

WiAV is easily distinguished from this case. The only reference to a "portfolio" license in *WiAV* describes a one-way license granted to LG Electronics, Ltd. by WiAV that specifically authorized LG to practice a limited subset of patents that WiAV

[104] Defendants' reference to 3Com's SEC filings, in which 3Com notes that it derives "a significant portion of [***3Com's***] personal connectivity product sales from PC OEMs such as … IBM" and that Intel is "well positioned in the Ethernet connectivity market" are not a reasonable basis for 3Com to believe that either IBM or Intel were infringing the Patents-in-Suit. *See* A7467-A7468.

[105] The district court's misplaced reliance on *WiAV* might be explained by counsel for Intel's mischaracterization of the case during oral argument. A6537 ("We did, in fact, cite the case, *WiAV* case in our reply that specifically did rely on a broad cross-license. So the idea that we haven't cited any cases that suggest that a broad cross-license could be the basis for the kind of marking motion we brought is not true.").

purchased from Skyworks Solutions. 732 F. Supp. 2d at 636. Despite this specific authorization, the license did not include a marking provision. *Id.* This is a far cry from a broad, portfolio-wide cross license between large technology companies, each with tens of thousands of unidentified products and thousands of unidentified patents.

These distinctions are important for two reasons. First, a one-way license does not present the same motivations to the parties. In a one-way license, the prospective licensee is specifically seeking to practice the technology of the licensor. In a portfolio-wide cross license between major technology companies, the primary motivation is most often the certainty of patent "peace," as reflected in the sworn testimony in this case.[106] Second, a significant factor in the "reasonableness" analysis should be the burden placed on the parties. In short, there is nothing in the *WiAV* case to suggest that it would have been unreasonable – let alone impossible – for WiAV to require LG to mark its products with the specific subset of patents that LG was specifically authorized to practice under the license. That is not the case here. For at least these reasons, *WiAV* is inapplicable here.

Because 3Com and USEI made reasonable efforts to ensure that the licensees to the '872 Patent complied with the marking provision, the district court's order precluding USEI from recovering for pre-suit damages should be reversed.

[106] A7073; A10874:17-A10875:3; A7315:17-23.

2. One Cannot Conclude that Patented Articles Were Sold Unmarked by a 3Com Licensee After April 30, 2005

Defendants failed to present any evidence that a patented article was sold unmarked after April 30, 2005. It is undisputed that the Intel CNS terminated on November 8, 2004. A7283; A7292. Thus, any products practicing the asserted claims sold by Intel after November 8, 2004, were merely infringing products that cannot implicate the marking requirement of 35 U.S.C. §287.[107] Moreover, Defendants have not presented ***any*** evidence that any IBM product practices the asserted claims. Accordingly, whether a patented article was sold unmarked after April 30, 2005, turns entirely on IBM's alleged re-sale of the Intel products accused of infringement.

The only evidence of products sold by IBM is a declaration from an IBM employee, providing as follows:

- Every year since 2003, IBM has purchased components from Intel for incorporation into IBM products, such as those products identified in IBM000001-IBM000002.[108]
- The products identified in IBM000001-IBM000002 are a very small sample of IBM products sold during the years 2004-2009.

[107] *Mass. Institute of Tech. v. Abacus Software, Inc.*, No. 5:01-cv-344, 2004 WL 5268123, at *40 (E.D. Tex. Aug. 4, 2004).

[108] Fifteen of the twenty-seven products identified in IBM000001-IBM000002 are IBM laptop products. Lenovo purchased IBM's personal computing division, including the ThinkPad and ThinkCentre brands in April 2005. Lenovo is not licensed under the '872 Patent. Accordingly, there were no licensed sales of fifteen of the twenty-seven products identified in IBM000001-IBM000002 after April 30, 2005.

- During the period from 2004-2009, IBM sold millions of products, including many that incorporate Intel components.
- The vast majority of components that IBM has purchased from Intel have been incorporated into IBM products that IBM subsequently sold to its customers.

A7073-A7075.

The evidence presented by Defendants begs the questions: (1) what Intel products were incorporated into the IBM products; (2) did the IBM products, as sold, practice the asserted claims; (3) how many IBM products that incorporate accused Intel products were sold before 2009; and (4) how many products that incorporate accused Intel products were sold before 2009 in the United States. Viewing the evidence in the light most favorable to USEI, the IBM declaration establishes nothing more than the fact that from 2004-2009 IBM sold "many" products that incorporate some unidentified Intel components. Indeed, the only evidence of actual IBM sales between April 30, 2005 and 2009 are sales of approximately 4,000 units of IBM servers to unidentified purchasers in unidentified locations that incorporate some unidentified Intel component. The evidence presented cannot support, as a matter of law, the district court's conclusion that IBM re-sold in the United States, between

2004 and 2009, the "vast majority" of ***twenty-five million infringing products*** purchased from Intel.[109] A48-A49 (emphasis added).

For this reason alone, the district court's order on marking should be reversed.

3. IBM and Intel Were Not Acting Within Their License Rights

a. IBM Was Not Acting Within Its License Rights When It Purchased the Infringing Intel Products and Resold Them

Defendants' argument that IBM's sale of accused Intel products somehow implicates the marking statute cannot survive close scrutiny. The plain language of the IBM Agreement demonstrates that IBM would not have been acting within its license rights if it resold accused Intel products. As a result, those sales cannot be imputed to the patentee, and the marking statute is not implicated.

The IBM Agreement defines the license granted by 3Com to IBM:

> (a) to make (including the right to use any apparatus and practice any method in making), use, import, offer for sale, lease, sell and/or otherwise transfer or distribute IBM's Licensed Products[; and]
>
> (b) to have IBM's Licensed Products made by another manufacturer for the use, importation, offer for sale, lease, sale and/or other transfer or distribution by IBM only when the conditions set forth in Section 2.2 are met.

A6985 (in relevant part).

[109] The district court failed to address whether any of the IBM sales occurred within the United States.

There is no evidence that any product "made" by IBM practices the asserted claims of the Patents-in-Suit. Defendants' argument is premised entirely on Intel products purchased by IBM and resold to customers. Section 2.2 places significant restrictions on the license granted to IBM for products "made by another," providing that IBM's right to sell products extended only to situations in which IBM created the specifications for the product and provided them to the manufacturer, Intel here. A6985 (in relevant part).

There is no evidence in the record that IBM created the specifications for the Intel products that form the basis of Defendants' argument under the IBM Agreement.[110] To the contrary, the evidence suggests that the Intel products at issue are "off-the-shelf" products sold to a wide variety of Intel's customers.[111]

The District of Delaware's decision in *Tulip Computs. Int'l B.V. v. Dell Computs. Corp.*, 262 F. Supp. 2d 358 (D. Del. 2003) confirms this conclusion. In that case, the court was faced with the very same question in the context of an IBM license agreement that is nearly word-for-word identical to the one at issue here, and the court refused to impute IBM's sales of third-party infringing products to the patentee.

---

[110] There is no evidence in the record that any IBM product infringes the Patents-in-Suit. USEI has not accused any IBM product of infringement.

[111] *See* A7000-7021 (showing the Intel ICH6 family of chips, which includes Intel's 82801AB, in various Dell products) (relevant excerpts attached as Ex. Y to Jones Decl.).

In *Tulip*, the court determined that IBM's re-selling of the alleged infringer's products could not be imputed to the patentee because IBM was not acting in accordance with its licensing obligations. *Id.* at 365-66. The district judge concluded:

> Accordingly, since IBM did not have [the alleged infringer] make those products in accordance with the specific restrictions in the IBM-Tulip license, IBM was not acting within its license rights when it purchased the allegedly infringing product from [the alleged infringer] and resold them to its customers. To read the "make, use, or sell" language to mean IBM was entitled to sell infringing product acquired from a third party makes no sense ….

Id. at 366. For the same reasons expressed by the *Tulip* court above, any unmarked sales of infringing Intel products by IBM should not be imputed to USEI because IBM was not acting within its license rights when it purchased the Intel products and resold them.

b. Intel Was Not Acting Within Its License Rights When It Sold the Infringing Intel Products

As discussed above, the Intel CNS was part of a larger settlement agreement that included an express cross-license.[112] While the CNS appears to be unlimited in terms of the relevant patents, the cross-license is restrictive. Under the cross-license, Intel is only licensed to five 3Com patents – none of which are at issue here. A7380-A7381; A7384.

[112] *See supra* §VI.D.3.

At least one district court has directly considered "the scope to be accorded the 'for or under' language of §287(a) when there is a restrictive express license that contains an apparently unlimited covenant not to sue."[113] In *Adams*, the court concluded that sales of products that fall outside the bounds of the restrictive license "cannot be imputed to Plaintiff for purposes of the marking statute" under an apparently unrestricted covenant not to sue.[114]

Accordingly, any sales by Intel should not be imputed to 3Com under the Intel CNS because they do not fall within the licensing rights granted Intel in the related Cross-License Agreement between the parties.

4. At a Minimum, the District Court's Sweeping Relief Is Premature

The district court's order precluding USEI from recovering for nearly any act of infringement before the date of suit is based upon a flawed and wholly unsupported assumption that all of USEI's allegations of infringement against the wide variety of Defendants' products will rise or fall together. At a minimum, the district court's order is premature with regard to any product other than those Intel products specifically mentioned in the Defendants' motion.

---

[113] *Phillip M. Adams & Assocs., LLC v. Winbond Elecs. Corp.*, No. 1:05-cv-64-TS, 2010 WL 3522097, at *4 (D. Utah Sept. 8, 2010).

[114] *Id.* at *5.

VII. CONCLUSION AND STATEMENT OF RELIEF SOUGHT

For the reasons above, the Court should reverse the district court's orders and remand for further proceedings.

Respectfully submitted, this 14th day of July, 2015.

ROBBINS GELLER RUDMAN
& DOWD LLP

/s/ John C. Herman

John C. Herman
Ryan K. Walsh
Peter M. Jones

3424 Peachtree Road, N.E.
Suite 1650
Atlanta, GA 30326
Telephone: (404) 504-6500
Facsimile: (404) 504-6501
jherman@rgrdlaw.com
rwalsh@rgrdlaw.com
pjones@rgrdlaw.com

Attorneys for Plaintiff-Appellant
U.S. Ethernet Innovations, LLC

ADDENDUM TO THE BRIEF OF PLAINTIFF-APPELLANT

Pursuant to Fed. Cir. R. 28(a)(12), Plaintiff-Appellant presents the following Addendum to the Brief of Plaintiff-Appellant. Enclosed please find the following:

Exhibit 1: Order Granting in Part and Denying in Part Motion for Partial Summary Judgment Addressing Marking and Nominal Damages, Granting USEI's Rule 56(D) Motion and Granting in Part and Denying in Part USEI's Motion for Leave to File A Notice of Newly Discovered Evidence and A Surreply

Exhibit 2: Order on Motion to Supplement the Record

Exhibit 3: Order on Summary Judgment Motions

Exhibit 4: Clerk's Judgment

Exhibit 5: Amended Clerk's Judgment

Exhibit 6: Order denying Motion to Alter or Amend Summary Judgment Order

Exhibit 7: U.S. Patent No. 5,434,872

Exhibit 8: U.S. Patent No. 5,732,094

Exhibit 1

United States District Court
For the Northern District of California

IN THE UNITED STATES DISTRICT COURT

FOR THE NORTHERN DISTRICT OF CALIFORNIA

U.S. ETHERNET INNOVATIONS, LLC, No. C 10-3724 CW

Plaintiff,

v.

ACER, INC.; ACER AMERICA CORPORATION; APPLE, INC.; ASUS COMPUTER INTERNATIONAL; ASUSTEK COMPUTER, INC.; DELL, INC.; FUJITSU, LTD.; FUJITSU AMERICA, INC.; GATEWAY, INC.; HEWLETT PACKARD CO.; SONY CORPORATION; SONY CORPORATION OF AMERICA; SONY ELECTRONICS INC.; TOSHIBA CORPORATION; TOSHIBA AMERICA, INC.; and TOSHIBA AMERICA INFORMATION SYSTEMS, INC.,

Defendants.

INTEL CORPORATION; NVIDIA CORPORATION; MARVELL SEMICONDUCTOR, INC.; ATHEROS COMMUNICATIONS, INC.; and BROADCOM CORPORATION,

Intervenors.

_____/

U.S. ETHERNET INNOVATIONS, LLC,,

Plaintiff,

v.

AT&T MOBILITY LLC; BARNES & NOBLE, INC.; CLAIRE'S BOUTIQUES, INC.; J. C. PENNEY COMPANY, INC.; SALLY BEAUTY HOLDINGS, INC.; ANN TAYLOR STORES CORPORATION; ANN TAYLOR RETAIL, INC.; HARLEY-DAVIDSON, INC.; HARLEY-DAVIDSON MOTOR COMPANY, INC.; KIRKLAND'S INC.; KIRKLAND'S STORES, INC.; MACY'S, INC.; MACY'S RETAIL HOLDINGS, INC.; MACY'S WEST STORES, INC.; NEW YORK & COMPANY, INC.; LERNER NEW YORK, INC.; RADIOSHACK CORPORATION; RENT-A-CENTER, INC.; and THE DRESS BARN, INC.,

Defendants.

_____/

AND ALL RELATED CLAIMS AND COUNTERCLAIMS

_____/

No. C 10-5254 CW

ORDER GRANTING IN PART AND DENYING IN PART MOTION FOR PARTIAL SUMMARY JUDGMENT ADDRESSING MARKING AND NOMINAL DAMAGES (Docket Nos. 766 in 10-3724, 450 in 10-5254), GRANTING USEI'S RULE 56(D) MOTION (Docket No. 793 in 10-3724) AND GRANTING IN PART AND DENYING IN PART USEI'S MOTION FOR LEAVE TO FILE A NOTICE OF NEWLY DISCOVERED EVIDENCE AND A SURREPLY (Docket No. 823 in 10-3724)

Intervenors Intel Corporation, Broadcom Corporation, Marvell Semiconductor Inc., Nvidia Corporation, Atheros Communications Inc. and Sigma Designs, Inc., and Defendants Acer, Inc., Acer America Corporation, Apple, Inc., Asustek Computer Inc., AT&T Services, Inc., Dell Inc., Fujitsu Ltd., Fujitsu America, Inc., Gateway, Inc., Hewlett Packard Co., Sony Corporation, Sony Corporation of America, Sony Electronics Inc., Toshiba Corporation, Toshiba America, Inc., and Toshiba America Information Systems, Inc. (collectively, with Defendant ASUS Computer International, Movants) move for partial summary judgment against Plaintiff U.S. Ethernet Innovations, LLC (USEI), seeking a determination that USEI cannot recover pre-suit damages for

infringement of three of the patents-in-suit, U.S. Patent Nos. 5,434,872, issued on July 18, 1995 (the '872 Patent), 5,307,459, issued on April 26, 1994 (the '459 Patent) and 5,299,313, issued on March 29, 1994 (the '313 Patent) based on its failure to mark under 35 U.S.C. § 287(a).[1] ASUS Computer International also seeks a determination that USEI cannot recover pre-suit damages for infringement of the '313 Patent based on its failure to mark. Intel and Nvidia further move for partial summary judgment precluding USEI from recovering more than nominal damages against them. USEI opposes the motion, moves to continue resolution of Intel and Nvidia's request under Federal Rule of Civil Procedure 56(d) and moves for leave to file a notice of additional evidence and a surreply. Having considered the papers filed by the parties and their arguments at the hearing on this motion, the Court GRANTS in part the motion for partial summary judgment, DENIES it in part, GRANTS the Rule 56(d) motion and GRANTS IN PART AND DENIES IN PART the motion for leave to file a notice of additional evidence and surreply.

BACKGROUND

On June 1, 1999, 3Com Corporation, the original owner of the patents-in-suit, entered into a license agreement with International Business Machines Corporation (IBM). Constant Decl. ¶ 2, Ex. A.1. Among other things, the license agreement addressed IBM's use of "3COM Licensed Patents," which are defined to "mean

---

[1] Movants do not seek summary judgment regarding U.S. Patent No. 5,732,094 (the '094 Patent) on this ground. Accordingly, USEI's argument that the '094 Patent is not subject to the requirements of 35 U.S.C. § 287(a) is not on point.

all patents," with certain limitations, "issued or issuing on patent applications entitled to an effective filing date prior to June 1, 2005" and for which 3Com had the right to grant licenses. Id. at § 1.4. The license agreement provided in relevant part that 3Com

> grants to IBM, as Grantee, a nonexclusive and worldwide license under 3COM's Licensed Patents:
>
> (a) to make (including the right to use any apparatus and practice any method in making), use, import, offer for sale, lease, sell, and/or otherwise transfer or distribute IBM's Licensed Products, and/or practice any businesses related to such IBM Licensed Products;
>
> (b) to have IBM's Licensed Products made by another manufacturer for the use, importation, offer for sale, lease, sale and/or other transfer or distribution by IBM only when the conditions set forth in Section 2.2 are met.
>
> . . .

Id. at § 2.1.2. The phrase "any apparatus" was not defined in the license agreement. However, the agreement did provide definitions for several types of apparatuses, including "Manufacturing Apparatus," "Data Processing Apparatus" and "Semiconductor Apparatus." Id. at §§ 1.8-1.11.

Section 2.2 provided,

> The License granted in [Section 2.1.2(b)] to the respective Grantee to have products made by another manufacturer:
>
> . . .
>
> (b) shall only apply when the specifications for such Grantee's Licensed Products were created by or for the Grantee (either solely or jointly with one or more third parties; and
>
> (c) shall not apply to (i) any methods used; or (ii) any products in the form manufactured or marketed by said other manufacturer prior to the Grantee's furnishing of said specifications.

. . .

Id. at § 2.2.

The agreement defined "IBM Licensed Products" to mean "IHS Products and Peripheral Devices." Id. at § 1.5. In turn, "IHS Products" was defined to mean,

> an Information Handling System or any instrumentality or aggregate of instrumentalities (including without limitation, any component, subassembly, computer program or supply) designed for incorporation in an Information Handling System. . . .

Id. at § 1.2. An "Information Handling System" referred to

> any instrumentality or aggregate of instrumentalities primarily designed to compute, classify, process, transmit, receive, retrieve, originate, switch, store, display, manifest, measure, detect, record, reproduce, handle or utilize any form of information, intelligence or data for business, scientific, control, entertainment, or other purposes.

Id. at § 1.1. A "Peripheral Device" meant "an ancillary device which is designed to operate in combination with a Grantee's Licensed Product." Id. at § 1.19. The agreement did not require that IBM Licensed Products be marked with the relevant patents utilized, including the patents-in-suit.

On November 8, 2002, Intel and 3Com entered into three agreements to resolve litigation taking place between 3Com and Xircom, Inc., a subsidiary of Intel. Jones Decl., Ex. AA.[2] These included a settlement agreement, a covenant not to sue and a cross license. The latter two documents were attached to the settlement agreement as exhibits. In the covenant not to sue, Intel and 3Com agreed not to assert "any Patent" that they owned against one another for a two-year time period. Id. at Ex. I

[2] Although USEI refers to this as the "IBM Settlement Agreement," IBM was not a party to any of the three agreements.

(Covenant Not to Sue) §§ 3.2.1, 3.2.2. The covenant not to sue did not require Intel to mark any products it made during this time period to qualify for the protection under it. In the cross-license agreement, 3Com granted Xircom and Intel a license to practice five of 3Com's patents, none of which are at issue in this case. Id. at Ex. H (Cross License Agreement) §§ 1.6, 3.3, 3.4.

Between 2003 and 2006, 3Com's patented product sales totaled about $200 million. Constant Decl., Ex. A.5, 475:16-23; Constant Decl., Ex. A.6. Between November 2003 and November 2004, Intel sold more than 140 million components that USEI now accuses of infringing the patents-in-suit, for almost $1.9 billion. Insley Decl. ¶ 3, Ex. B. Between 2004 and 2009, IBM purchased more than twenty-five million Intel components, for more than $350 million. Insley Decl. ¶ 2, Ex. A. The "vast majority" of the components that IBM purchased from Intel were incorporated into IBM products that it subsequently sold to customers. Constant Decl., Ex. A.3, ¶ 10.[3] In the present lawsuit, USEI's infringement contentions against Intel accuse many of the components that IBM purchased for inclusion in its products. Constant Decl., Ex. A.2, 2-5. Neither IBM nor Intel marked their products with the '872, '459 or '313 Patent numbers. Schiffhauer Decl. ¶ 2; Constant Decl., Ex. A.3 ¶ 11.

[3] In their reply brief, Movants cite "Exhibit C" to the Constant declaration. However, there is no such exhibit attached to this declaration, and it appears that Movants intended to refer to Exhibit A.3 to the declaration instead.

Starting in 2003, 3Com was involved in a separate lawsuit in this district with Realtek Semiconductor System and D-Link Systems, Inc. In that case, 3Com alleged that Realtek and D-Link infringed of the '872, '459 and '094 Patents. The '313 Patent was not at issue in that case. In 2008, the court denied Realtek's motion for partial summary judgment based on marking. Jones Decl., Ex. I. In opposing the summary judgment motion, 3Com offered "a spreadsheet showing the sales of products practicing the patents at issue for which marked samples have been located and products for which marked samples could not be found during the period roughly from 1997 until [July 13, 2004] . . . According to this spreadsheet, 3Com sold 32,196,864 units practicing the patents at issue during this time, of which 4,803,508 units, or 14.9 percent were of product types for which 3Com could not locate a marked sample." Id. at 1, 7. Although the court noted that "[m]arking 85 percent of products sold is almost certainly not marking 'substantially all' products," it held that a jury could infer, based on other evidence in the record, that, although 3Com had lost some documentary evidence, it had a practice of marking its products since at least 1997 and thus that the remaining products for which documentary evidence did not exist were in fact marked. Id. at 7-8. USEI has offered here the evidence that 3Com submitted in the Realtek case.

At the deposition taken in this matter, USEI's Rule 30(b)(6) witness testified that he did not know if there were ever any products marked with the '313 Patent produced by 3Com or anyone else. Constant Decl., Ex. A.7, 21:6-14. USEI contends that 3Com did practice this patent. Constant Decl., Ex. A.2, 18.

United States District Court
For the Northern District of California

USEI also offers evidence that it has entered into license agreements with Realtek, ADMtek and VIA, to authorize them to use some or all of the '459, '872 and '094 Patents, that these licenses included marking requirements, and that the licenses had complied with their marking obligations or that 3Com took steps to ensure that they did so.

LEGAL STANDARD

Summary judgment is properly granted when no genuine and disputed issues of material fact remain, and when, viewing the evidence most favorably to the non-moving party, the movant is clearly entitled to prevail as a matter of law. Fed. R. Civ. P. 56; Celotex Corp. v. Catrett, 477 U.S. 317, 322-23 (1986); Eisenberg v. Ins. Co. of N. Am., 815 F.2d 1285, 1288-89 (9th Cir. 1987).

The moving party bears the burden of showing that there is no material factual dispute. Therefore, the court must regard as true the opposing party's evidence, if supported by affidavits or other evidentiary material. Celotex, 477 U.S. at 324; Eisenberg, 815 F.2d at 1289. The court must draw all reasonable inferences in favor of the party against whom summary judgment is sought. Matsushita Elec. Indus. Co. v. Zenith Radio Corp., 475 U.S. 574, 587 (1986); Intel Corp. v. Hartford Accident & Indem. Co., 952 F.2d 1551, 1558 (9th Cir. 1991).

Material facts which would preclude entry of summary judgment are those which, under applicable substantive law, may affect the outcome of the case. The substantive law will identify which facts are material. Anderson v. Liberty Lobby, Inc., 477 U.S. 242, 248 (1986).

Where the moving party does not bear the burden of proof on an issue at trial, the moving party may discharge its burden of production by either of two methods:

> The moving party may produce evidence negating an essential element of the nonmoving party's case, or, after suitable discovery, the moving party may show that the nonmoving party does not have enough evidence of an essential element of its claim or defense to carry its ultimate burden of persuasion at trial.

Nissan Fire & Marine Ins. Co., Ltd., v. Fritz Cos., Inc., 210 F.3d 1099, 1106 (9th Cir. 2000).

If the moving party discharges its burden by showing an absence of evidence to support an essential element of a claim or defense, it is not required to produce evidence showing the absence of a material fact on such issues, or to support its motion with evidence negating the non-moving party's claim. Id.; see also Lujan v. Nat'l Wildlife Fed'n, 497 U.S. 871, 885 (1990); Bhan v. NME Hosps., Inc., 929 F.2d 1404, 1409 (9th Cir. 1991). If the moving party shows an absence of evidence to support the non-moving party's case, the burden then shifts to the non-moving party to produce "specific evidence, through affidavits or admissible discovery material, to show that the dispute exists." Bhan, 929 F.2d at 1409.

If the moving party discharges its burden by negating an essential element of the non-moving party's claim or defense, it must produce affirmative evidence of such negation. Nissan, 210 F.3d at 1105. If the moving party produces such evidence, the burden then shifts to the non-moving party to produce specific evidence to show that a dispute of material fact exists. Id.

If the moving party does not meet its initial burden of production by either method, the non-moving party is under no obligation to offer any evidence in support of its opposition. Id. This is true even though the non-moving party bears the ultimate burden of persuasion at trial. Id. at 1107.

DISCUSSION

I. Failure to mark

Movants contend that they are entitled to a summary adjudication that USEI cannot collect pre-suit damages for infringement of the '459, '872 and '313 Patents pursuant to 35 U.S.C. § 287(a) because it cannot show that "substantially all" of the licensed products were marked during the damages period[4] or that it provided any or adequate actual notice of alleged infringement to Movants.[5] They argue that, even assuming that 3Com marked substantially all of the products that it sold itself, this is not sufficient to meet the statutory requirement because 3Com's sales were dwarfed by the sales by Intel and IBM, as licensees, of purportedly infringing products.

---

[4] USEI filed the original complaint in the earlier case on October 9, 2009. Title 35 U.S.C. § 286 provides that "no recovery shall be had for any infringement committed more than six years prior to the filing of the complaint or counterclaim for infringement in the action." Accordingly, USEI may not recover damages for infringement that occurred prior to October 3, 2003.

[5] ASUS Computer International joins this motion as to the '313 Patent but not as to the '459 and '872 Patents. It acknowledges that it received a letter from 3Com dated December 8, 2003, asserting that certain ASUS products infringed the '459 and '872 Patents.

USEI responds that 3Com and its authorized licensees did mark "substantially all" patented products that they produced, and it disputes that Intel and IBM were actually licensed.[6]

Title 35 U.S.C. § 287(a) provides,

> Patentees, and persons making, offering for sale, or selling within the United States any patented article for or under them, . . . may give notice to the public that the same is patented, either by fixing thereon the word "patent" or the abbreviation "pat.", together with the number of the patent, or by fixing thereon the word "patent" or the abbreviation "pat." together with an address of a posting on the Internet, accessible to the public without charge for accessing the address, that associates the patented article with the number of the patent, or when, from the character of the article, this can not be done, by fixing to it, or to the package wherein one or more of them is contained, a label containing a like notice. In the event of failure so to mark, no damages shall be recovered by the patentee in any action for infringement, except on proof that the infringer was notified of the infringement and continued to infringe thereafter, in which event damages may be recovered only for infringement occurring after such notice. Filing of an action for infringement shall constitute such notice.

"Thus, the statute defines that '[a patentee] is entitled to damages from the time when it either began marking its product in compliance with section 287(a)[, constructive notice,] or when it actually notified [the accused infringer] of its infringement, whichever was earlier.'" Maxwell v. J. Baker, Inc., 86 F.3d 1098, 1111 (Fed. Cir. 1996) (quoting American Medical Sys., Inc. v. Medical Eng'g Corp., 6 F.3d 1523, 1537 (Fed. Cir. 1993)) (formatting in original). The Federal Circuit has "construed

---

[6] USEI has not presented any argument to dispute Movants' contentions regarding the issue of actual notice. Although it avers in a footnote that "[a]ny decision on actual notice is premature" in light of the status of discovery, Opp. at 19 n.51, it has not made a showing of any essential facts relevant to this determination that it may obtain through discovery.

section 287(a) to require that 'once marking has begun, it must be substantially consistent and continuous in order for the party to avail itself of the constructive notice provisions of the statute.'" Id. (quoting American Medical Sys., 6 F.3d at 1537).

"In addition to governing patent owners, the provisions of 35 U.S.C. § 287(a) extend to licensees." Tulip Computers Int'l B.V. v. Dell Computer Corp., 262 F. Supp. 2d 358, 362 (D. Del. 2003) (citing 35 U.S.C. § 287(a); Devices for Medicine, Inc. v. Boehl, 822 F.2d 1062 (Fed. Cir. 1987)); see also Amsted Indus. v. Buckeye Steel Castings Co., 24 F.3d 178, 185 (Fed. Cir. 1994) ("A licensee who makes or sells a patented article does so 'for or under' the patentee, thereby limiting the patentee's damage recovery when the patented article is not marked."); Maxwell, 86 F.3d at 1111 ("licensees, such as Target, and other authorized parties, such as Target's manufacturers, must also comply" with the marking requirement). "Thus, while an infringer's failure to mark does not implicate Section 287(a), . . . a licensee's failure to mark does bear consequences for a patent owner seeking damages for infringement." Id. (internal citation omitted). "A patent owner bears responsibility to ensure that 'licensees . . . and other authorized parties . . . comply' with Section 287(a)." Id. (quoting Maxwell, 86 F.3d at 1111-12). This requirement applies to both express licensees and implied licensees. Amsted Indus., 24 F.3d at 185.

"However, with third parties unrelated to the patentee, it is often more difficult for a patentee to ensure compliance with the marking provisions." Maxwell, 86 F.3d at 1111. "A 'rule of reason' approach is justified in such a case and substantial

compliance may be found to satisfy the statute." Id. "When the failure to mark is caused by someone other than the patentee, the court may consider whether the patentee made reasonable efforts to ensure compliance with the marking requirements." Id. at 1111-12.

As the patentee, USEI bears the burden of pleading and proving compliance with the marking statute. Id. at 1111.

As previously noted, the parties dispute whether Intel or IBM was either explicitly or implicitly licensed to practice the patents at issue. If Intel and IBM were not acting in accordance with a license or with their obligations under a license when they sold the products in question, then the sales may not be imputed to 3Com or its successor, USEI. See Maxwell, 86 F.3d at 1111-12; Tulip, 262 F. Supp. 2d at 362.

Regarding IBM, USEI argues that the 1999 license agreement between 3Com and IBM did not permit IBM to purchase infringing Intel components and then resell them, absent compliance with the requirements of section 2.2 of the agreement that IBM create the specifications for the products and not purchase premade items. Movants contend that the evidence shows that IBM did not simply sell the premade components but instead used them to make its Licensed Products and then sold the completed item. Movants argue that this was authorized by section 2.1.2(a) of the license agreement, which permits IBM to "use any apparatus . . . in making" its Licensed Products.

USEI's interpretation of the 3Com-IBM license agreement is not persuasive. The sections on which it relies govern the terms under which IBM can use 3Com's patents "to have IBM's Licensed Products made by another manufacturer." Constant Decl., Ex. A.1,

§§ 2.1.2(b), 2.2. Thus, these sections do not address the terms under which IBM can obtain parts for use when it makes its own Licensed Products. Here, there is no dispute that IBM made its own products; the issue is instead whether it was permitted to obtain purportedly infringing, premade components and to use those components when it made its products.

As Movants contend, pursuant to section 2.1.2(a) of the license agreement, the answer is that IBM was allowed to do this. In that provision, 3Com gave IBM the right to make its own Licensed Products and to "use any apparatus" in doing so. The agreement placed no restriction that IBM was permitted to use only particular types of components--for example, those obtained from licensed chip suppliers--and instead unambiguously provided that "any apparatus" could be used. USEI provides no argument that the relevant components accused here do not qualify as "any apparatus" under the terms of section 2.1.2(a). Although USEI argues that the reference in section 2.1.2(a) to "any apparatus" should be construed to mean only "manufacturing apparatus," it provides no reason that this phrase should be so construed. Notably, the parties to the agreement specifically defined and used the term "manufacturing apparatus" elsewhere in the agreement where they intended to do so and distinguished it from other apparatus types. Thus, to define apparatus to mean "manufacturing apparatus" here would render other parts of the agreement surplusage or contradictory. Further, the Court notes that the term "apparatus" is frequently used in the patents at issue themselves to refer to the claimed inventions.

The sole case upon which USEI relies is not to the contrary. In that case, Tulip Computers, IBM had entered into a license agreement with Tulip very similar to the one at issue here. In that case, the court held that IBM's purchase and sale of the allegedly infringing products from Dell was not imputable to Tulip because IBM had not complied with the specific restrictions on its "have made" rights under section 2.2 of the agreement and was thus not acting within its license granted by Tulip. Tulip Computers, 262 F. Supp. 2d at 366. However, there, IBM had entered into a remarketing agreement with Dell Computer Corporation, pursuant to which "IBM made off-the-shelf purchases of products manufactured by Dell and resold them to its customers." Id. at 360. Thus, there, IBM did not make the products at issue and rather purchased them premade. In contrast, here, IBM made its own licensed products, using chips purchased from Intel. USEI's argument that Movants' interpretation would render the restrictions in section 2.2 of the agreement "virtually meaningless" is also unavailing. IBM was not allowed to have its own licensed products made by another manufacturer practicing 3Com's patents, except in compliance with those restrictions.

Thus, because the license agreement allowed IBM to make and sell its own licensed products, using any apparatus to do so, it acted pursuant to its license rights when it made and sold products that included purportedly infringing components that it purchased from Intel.[7] Accordingly, those products--the "vast

[7] USEI has moved for leave to file a surreply to address this issue. It contends that this argument was raised for the first time in Movants' reply brief.

majority" of twenty-five million between 2004 and 2009--can be imputed to 3Com and USEI. There is no dispute that neither Intel nor IBM marked these products or components with the patent numbers. USEI has not offered evidence that the products sold by IBM constitute a de minimis number of the licensed products or that the marked products sold by 3Com or the licensees whose agreement required marking constituted substantially all of the relevant products sold.

USEI contends that summary judgment is nevertheless inappropriate because "3Com made reasonable efforts to ensure compliance with the marking requirements." Opp. at 13-16. It argues that, because the IBM-3Com cross-licensing agreement was "broad" and encompassed the entirety of each company's portfolio of patents, it would have been unreasonable for 3Com to monitor IBM's large number of products to determine which infringed the patents-in-suit. It contends that IBM-3Com agreement did not have "any relationship to the patents at issue" here and that there is

---

In their opening brief, Movants made clear that they based their arguments in part on IBM's sale of its own licensed computers that were built with accused Intel components, not simply on IBM's "sale of accused Intel chips," as USEI argues. Mot. for Leave to File a Notice and Surreply, Docket No. 823, 2 n.1; see, e.g., Mot. at 3 (arguing that "IBM purchased more than 25 million Intel components now accused of infringement for inclusion in IBM's licensed products"). Thus, USEI could have responded to this argument in its opposition brief. Further, even if USEI misunderstood the argument at the time that it filed its opposition, it had ample opportunity to present any further response on this issue at the hearing.

Nonetheless, because the Court has considered the arguments raised by USEI in its proposed surreply and has not found them meritorious, the Court GRANTS it leave to file the surreply.

no evidence that 3Com suspected that IBM was selling products that infringed any of these patents.

However, here, the evidence does not support that 3Com made reasonable efforts to ensure that IBM marked any licensed products with the patents-in-suit. In fact, the evidence offered shows that 3Com took no steps whatsoever to do so. Even if it would have been burdensome for 3Com to monitor whether IBM was in fact marking all products practicing the patents, 3Com failed even to include in the licensing agreement an obligation that IBM itself make efforts to mark any products that it reasonably believed practiced 3Com's patents.[8] It instead authorized IBM to practice the patents-in-suit without any requirement that IBM even make a minimal effort to mark them. 3Com could reasonably have taken such a step without first knowing itself that IBM was making products that practiced its patents, given that 3Com was authorizing IBM to do so.[9] USEI's argument that the licensing

---

[8] The Court also notes that, although USEI contends that it would have been unreasonable to expect 3Com to investigate fully which IBM products practiced each of 3Com's patents in order to monitor compliance and to know which products had to be marked with the patents-in-suit, 3Com would simply have had to conduct the same type of investigation that USEI was required to conduct for the accused products in this suit.

[9] Movants have offered some evidence that 3Com did in fact have such knowledge. Notably, in an August 1999 Form 10-K filed with the Securities Exchange Commission, 3Com wrote that "we derive a significant portion of our personal connectivity product sales from PC OEMs, such as . . . IBM, who incorporate our [Network Interface Cards], analog modems, or chipsets into their products." Constant Decl., Ex. A.16.

agreement had no relationship to the patents-in-suit is unavailing. Although the agreement did not specifically identify certain patents by number, and the parties did not discuss in their negotiations particular products that would be licensed under the agreement, the license encompassed 3Com's patents with an effective filing date prior to June 5, 2005, which includes all of the patents at issue here.

In addition, USEI provides no authority that supports its contention that the sale of products under a broad portfolio license cannot be imputed to the patent holder.[10] The terms of § 287(a) do not contain any such express exemption and instead apply to any "persons making, offering for sale or selling within the United States any patented article" under a patentee. 35 U.S.C. § 287(a); see also In re Yarn Processing Patent Validity Litig., 602 F. Supp. 159, 169 (W.D.N.C. 1984) (this section "applies to authorizations by patentee of other persons to make

---

In response, USEI asserts that the Form 10-K is not a "reasonable basis for 3Com to believe that" IBM was infringing the patents-in-suit. Opp. at 16 n.42. However, USEI does not explain why not. Further, Movants did not offer this document as evidence from which 3Com should have learned that IBM was using technology covered by the patents-in-suit in its products; instead, they offered it to show that 3Com knew that IBM was doing so.

[10] USEI suggests in its opposition that the court in Clancy Sys. Int'l v. Symbol Techs., 953 F. Supp. 1170 (D. Colo. 1997), found that, for a licensee to sell a patented article "for or under" a patentee, the license must "specifically" permit the licensee to sell the infringing product, and thus that a "broad portfolio license" is insufficient. However, there was no such discussion or holding in Clancy. The portion of that opinion that USEI cites is simply a description of the license agreement at issue in that case and a recitation of the fact that the patent owner there admitted that the licensed product infringed the patent, thereby qualifying as a "patented article" under § 287(a). Id. at 1173.

United States District Court
For the Northern District of California

and sell patented articles regardless of the particular form these authorizations may take"), cited with approval by Amsted Indus. v. Buckeye Steel Castings Co., 24 F.3d 178, 185 n.2 (Fed. Cir. 1994). Here, 3Com, as the patentee, granted IBM the right to make and sell products that practice its patents. Other courts have applied marking requirements in the context of a portfolio license. See, e.g., WiAV Solutions LLC v. Motorola, Inc., 732 F. Supp. 2d 634, 636, 641-43 (E.D. Va. 2010). Further, to hold otherwise would be inconsistent with the "three related purposes" of the marking statute: "1) helping to avoid innocent infringement . . .; 2) encouraging patentees to give notice to the public that the article is patented . . . ; and 3) aiding the public to identify whether an article is patented . . ." Nike Inc. v. Wal-Mart Stores, 138 F.3d 1437, 1443 (Fed. Cir. 1998). Section 287(a) penalizes patentees who put into the marketplace--either directly or indirectly, through their failure to take reasonable efforts to police their licensees--unmarked products that do not inform the public that the intellectual property embodied in those products is protected.

The parties also dispute whether the products sold by Intel were subject to the marking requirement due to the 2002 covenant not to sue between Intel and 3Com. Although USEI contends that the purpose of the covenant not to sue was to obtain "patent peace" and not to grant a license or authorization to make products, courts have previously found that § 287's marking requirement applies to a patentee's authorization of other persons to make and sell patented items in whatever form the authorization was given, "regardless of whether the authorizations are

'settlement agreements,' 'covenants not to sue' or 'licenses.'" In re Yarn Processing Patent Validity Litig., 602 F. Supp. at 169; see also De Forest Radio Tel. Co. v. United States, 273 U.S. 236 (1927) ("As a license passes no interest in the monopoly, it has been described as a mere waiver of the right to sue by the patentee") (internal quotation marks and citation omitted); TransCore, LP v. Elec. Transaction Consultants Corp., 563 F.3d 1271, 1275 (Fed. Cir. 2009) (stating that "this court and its predecessors have on numerous occasions explained that a non-exclusive patent license is equivalent to a covenant not to sue" and collecting cases).

USEI also argues that the covenant not to sue should not be read as such an authorization because 3Com did not grant Intel the rights to practice the patents-in-suit under the restrictive cross-license that they entered at the same time as the covenant not to sue. In support of its argument, USEI relies on Phillip M. Adams & Assocs., LLC v. Winbond Elecs. Corp., 2010 U.S. Dist. LEXIS 93799 (D. Utah), in which the district court considered a licensing agreement that contained a covenant not to sue and addressed whether it authorized the sale of particular products, which were not encompassed in the express license, as "for or under the" patentee pursuant to § 287. The court found that, to read the covenant not to sue as granting an implied license as to all matters covered in that covenant would render all of the limitations in the express license "surplusage, an impermissible result under California law." Id. at 29-30. Thus, the court concluded that the products sales could not be imputed to the patentee for the purposes of the marking statute. Id. at 30.

United States District Court
For the Northern District of California

However, USEI fails to demonstrate that the separate cross license and covenant not to sue at issue here are the equivalent of those analyzed by in the Adams case or that reading of the covenant not to sue as an implied license in this case would render any part of the cross license as surplusage. Unlike in Adams, the documents here are separate contracts that involve different parties. As explained above, to resolve the Xircom litigation, Xircom, 3Com and Intel contemporaneously entered into a series of contracts. The settlement agreement was signed by all three and had the other two contracts attached to it. The cross license was also executed by all three and involved only Xircom and 3Com patents, among other things, granting Xircom and Intel the right to use five specific 3Com patents that are not at issue in this litigation. Cross License Agreement §§ 1.6, 3.3, 3.4. In contrast, the covenant not to sue was executed only by Intel and 3Com. In that document, each of these companies agreed not to assert any of their patents against each other alleging infringement of the subject matter covered by the agreement. Covenant Not to Sue § 3.2. The covered subject matter is defined broadly and encompasses all products made by the respective companies, with limited exclusions. Id. at §§ 1.9, 1.10, 1.20, 1.21. Particularly in light of the differing time periods, scope and parties to the agreements, it is not clear that interpreting the covenant not to sue as an authorization to practice the patented technology would have any effect whatsoever on the separate cross license, and USEI has not identified any.

Further, reading the document as a whole, the covenant not to sue does unambiguously authorize Intel's manufacture and sale of

products under any of 3Com's covered patents, even if the parties did not specifically discuss or know that Intel was practicing the technology of the patents-in-suit at the time of contracting.[11] In the covenant, 3Com agrees that, for the effective term of the agreement, it "will not Assert any Patent owned by 3Com or as to which 3Com has the right to Assert infringement against Intel or Permitted Assignees, alleging direct or indirect infringement with respect to Covered Intel Subject Matter." Id. at § 3.2.2. "This term, without apparent restriction or limitation, thus authorizes all acts that would otherwise be infringements: making, using, offering for sale, selling, or importing." TransCore, 563 F.3d at 1276. This is particularly clear because "Covered Intel Subject Matter" encompasses "all products manufactured, imported, sold or offered for sale by or for Intel," with limited exceptions that USEI has not invoked here. Cross License Agreement §§ 1.9, 1.21; see also id. at § 3.1 (stating the intent of the parties to avoid "patent disputes between themselves as a result of sales of their products").

Accordingly, Intel's sales of purportedly infringing products during the time period covered by the covenant not to sue were carried out under the authority of 3Com and can thus be imputed to 3Com for the purposes of § 287's marking requirement. As with the IBM agreement, 3Com did not take reasonable efforts--or any efforts whatsoever--to ensure that any sale of products by Intel under this authorization complied with this requirement. Further,

---

[11] Movants have offered evidence that shows that 3Com was aware that Intel was a "well-positioned competitor" in the relevant technology at this time. Constant Decl., Ex. A.17, 9.

the number of purportedly infringing components sold by Intel during a one year time period from November 2003 to November 2004 dwarfs, by a factor of almost five, the number of components sold by 3Com for a seven year period through 2004.

To the extent that USEI relies on the district court's denial of the defendant's motion for partial summary judgment based on marking in the Realtek litigation, this is unavailing. The motion there was brought based on 3Com's purported failure to mark its own products with the patents, not on its failure to make reasonable efforts to ensure that licensees complied with the marking requirements. Even assuming for the purposes of this motion that 3Com did mark substantially all of its own products and that it made reasonable efforts to ensure that its express licensees, Realtek, ADMtek and VIA, did so, this is not sufficient to preclude summary judgment, in light of the failure to take reasonable steps to ensure that Intel and IBM marked products with the patents.

USEI also argues that the motion should be denied because Movants have not offered evidence that Intel's products infringe the patents-in-suit and instead rely on USEI's assertions of infringement. It contends that this motion is premature, because some or all of Intel's products may be found ultimately not to infringe and "there would be no reasonable argument that 3Com failed to comply with the marking requirements of Section 287 based on the sale of non-infringing Intel products." Opp. at 19. However, as explained by another district court presented with the same argument, this argument is insufficient to raise a genuine issue of material fact:

> Loral simply argues that a genuine factual issue remains as to whether the CCDs sold by Toshiba, NEC and the other defendants infringe claims of the '485 patent. Loral contends that this court cannot rule on this motion until the finder of fact rules on infringement. These legal contentions, however, do not satisfy Loral's burden to present factual evidence showing a genuine issue of material fact.
>
> Loral has consistently asserted in this case that the CCDs sold by Matsushita, Toshiba, and NEC in the United States infringe claims of the '485 patent. Defendants have conclusively shown these specific allegedly infringing CCDs were sold, unmarked, during the terms of license agreements with Fairchild. Loral has not withdrawn its claim of infringement of any of these devices. In sum, Loral has provided no evidence to show a genuine issue of material fact. See Fed. R. Civ. P. 56(e).
>
> Rather than supply evidence creating a genuine factual issue, Loral has merely argued the possibility that the trier of fact could find that these CCDs do not infringe -- a position inconsistent with Loral's pleadings, answers to interrogatories, statement of undisputed facts, and its interpretation of the scope of its patent, either literally or by equivalents. Loral cannot create a genuine issue of fact as to whether these licensed CCDs are covered by the '485 patent simply by arguing the possibility of such a result.

Loral Fairchild Corp. v. Victor Co. of Japan, 906 F. Supp. 813, 818 (E.D.N.Y. 1995). Similarly, here, USEI has consistently taken the position that the relevant products sold by Intel infringe its patents, and it has not offered any evidence here that they do not--instead, it just speculates that the factfinder ultimately may find that Intel's products do not infringe. With their motion, Movants offered evidence that the marking requirement was not met and thus the burden shifted to USEI to raise an issue of fact to preclude summary judgment. It cannot meet its burden to do so simply through argument unsupported by evidence.

Further, USEI has raised no genuine issue of material fact that 3Com failed to mark its products with the '313 Patent. With their motion, Movants offered evidence that 3Com did practice the

'313 Patent. In its infringement contentions, USEI asserted that "its predecessor in interest, 3Com Corporation, practiced [the Asserted Claims] as 3Com was a world-wide leader in Ethernet products until it was forced out of business . . ." Constant Decl., Ex. A.2, 18. USEI responds that this statement "falls short of demonstrating that 3Com sold an unmarked products that practiced the '313 Patent." Opp. at 18, n.49. However, Movants offered this to show that 3Com did make products that practiced the '313 Patent, and a jury could conclude from this statement that 3Com did this. Movants also offered testimony from USEI's Rule 30(b)(6) witness that USEI did "not know" if 3Com or anyone else ever marked products with the '313 Patent. Constant Decl., Ex. A.7, 21:6-14. Thus, Movants produced evidence that showed that USEI would not able to meet its burden of persuasion at trial to show that the products that 3Com made practicing the '313 Patent were marked. USEI did not respond with any evidence to demonstrate that there is a genuine issue of fact as to this.[12]

Finally, USEI argues that, even if the Court holds--as it has--that patented products sold by Intel during the term of the covenant not to sue should be imputed to 3Com, "[a]ny limitation on USEI's ability to recover damages for infringement of the Patents-in-Suit under 35 U.S.C. §287 should end on November 8, 2004," the expiration of that covenant. Opp. at 17-18. However,

---

[12] In addition, Movants offered evidence that USEI has identified Broadcom's BCM5700 chip of infringing the '313 Patent, see Constant Decl., Ex. A.2, 7; Bremer Reply Decl., Ex. A.2, 7, and that 3Com itself made unmarked products that utilized this chip, see Bremer Reply Decl., Exs. A.3, A.4. USEI did not address or respond to this evidence at the hearing.

this argument fails to take into account that the license agreement with IBM remained valid past that date and that IBM continued to sell unmarked products that included accused Intel components through 2009.

Accordingly, the Court GRANTS Movants' motion for partial summary judgment on the basis of marking. USEI cannot recover damages for any acts of infringement of the '872, '459 or '313 Patents against Movants, except ASUS Computer International, that took place before it filed infringement claims against each of them. Further, USEI cannot recover damages for any acts of infringement of the '313 Patents against ASUS Computer International that took place before USEI filed infringement claims against it.

II. Nominal damages

Intel and Nvidia move for summary judgment precluding USEI "from asserting that damages should be calculated by applying a royalty rate or percentage to revenues or profits from the accused products" and limiting USEI "to proffering a reasonable royalty based on the total costs of permanently removing the accused features from the accused products, rather than on any alleged value of the accused features to Intel, Nvidia, or their customers." Mot. at 20. They argue that, since before October 2003, they have disabled the accused features of their Ethernet products in the software drivers that they provide to purchasers. USEI moves to continue the motion for partial summary judgment pursuant to Federal Rule of Civil Procedure 56(d).

Although, if Intel and Nvidia were to prove ultimately that any infringing capabilities of their accused products were in fact

fully disabled for the damages period, and that the accused features were of no benefit to them or their customers during that time, USEI may be foreclosed from recovering damages as they urge, USEI has identified sufficiently specific evidence that it may obtain through discovery that may be relevant to the issue of disablement, including whether the features may have been re-enabled. Accordingly, the Court GRANTS USEI's Rule 56(d) motion. The Court DENIES Intel and Nvidia's motion for partial summary judgment without prejudice to re-filing as part of the parties' dispositive cross-motions that are set to be briefed after the close of fact and expert discovery.[13]

CONCLUSION

For the reasons set forth above, the Court GRANTS in part and DENIES in part Movants' motion for partial summary judgment (Docket Nos. 766 in 10-3724, 450 in 10-5254), GRANTS USEI's Rule 56(d) motion (Docket No. 793), GRANTS USEI leave to file a surreply and DENIES it leaves to file a notice of newly discovered evidence (Docket No. 823). Within three days of the date of this Order, USEI shall file its proposed surreply in the docket.

USEI is precluded from recovering damages for any acts of infringement of the '872, '459 or '313 Patents by Intervenors Intel Corporation, Broadcom Corporation, Marvell Semiconductor Inc., Nvidia Corporation, Atheros Communications Inc. and Sigma Designs, Inc., and Defendants Acer, Inc., Acer America Corporation, Apple, Inc., Asustek Computer Inc., AT&T Services,

---

[13] USEI's motion for leave to file newly discovered evidence regarding disablement is therefore DENIED as moot.

Inc., Dell Inc., Fujitsu Ltd., Fujitsu America, Inc., Gateway, Inc., Hewlett Packard Co., Sony Corporation, Sony Corporation of America, Sony Electronics Inc., Toshiba Corporation, Toshiba America, Inc., or Toshiba America Information Systems, Inc. that took place before it filed infringement claims against such Defendant or Intervenor. Further, USEI is precluded from recovering damages for any acts of infringement of the '313 Patents against ASUS Computer International that took place before USEI filed infringement claims against it.

IT IS SO ORDERED.

Dated: 8/16/2013

CLAUDIA WILKEN
United States District Judge

United States District Court
For the Northern District of California

Exhibit 2

United States District Court
For the Northern District of California

IN THE UNITED STATES DISTRICT COURT

FOR THE NORTHERN DISTRICT OF CALIFORNIA

U.S. ETHERNET INNOVATIONS, LLC,

Plaintiff,

v.

ACER, INC., et al.,

Defendants.

ATHEROS COMMUNICATIONS, INC., et al.,

Intervenors.

/

No. C 10-3724 CW

ORDER ON MOTION TO SUPPLEMENT THE RECORD

(Docket No. 1237)

U.S. ETHERNET INNOVATIONS, LLC,

Plaintiff,

v.

AT&T MOBILITY, LLC, et al.,

Defendants,

/

No. C 10-5254 CW

ORDER ON MOTION TO SUPPLEMENT THE RECORD

(Docket No. 552)

Before the Court is Plaintiff's Motion to Supplement the Record in these related cases. Case No. 10-3724, Docket No. 1237; Case No. 10-5254, Docket No. 552. The Court DENIES the motion in part and GRANTS it in part. Plaintiff may supplement the record only with an amended report by Mr. Bratic, but only to the extent necessary for him to remove improper bases for his calculations,

such as unmarked products or non-accused products. He may not add any additional products or units of product.

IT IS SO ORDERED.

Dated: October 10, 2014

CLAUDIA WILKEN
United States District Judge

United States District Court
For the Northern District of California

Exhibit 3

United States District Court
For the Northern District of California

IN THE UNITED STATES DISTRICT COURT

FOR THE NORTHERN DISTRICT OF CALIFORNIA

U.S. ETHERNET INNOVATIONS, LLC,

Plaintiff,

v.

ACER, INC., et al.,

Defendants.

and

ATHEROS COMMUNICATIONS, INC., et al.,

Intervenors.
_____/

No. C 10-3724 CW

ORDER ON SUMMARY JUDGMENT MOTIONS

(Docket Nos. 1133 and 1167)

U.S. ETHERNET INNOVATIONS, LLC,

Plaintiff,

v.

AT&T MOBILITY, LLC, et al.,

Defendants.
_____/

No. C 10-5254 CW

ORDER ON SUMMARY JUDGMENT MOTIONS

In this consolidated patent infringement case, Plaintiff U.S. Ethernet Innovations, LLC (USEI) moves for summary judgment on a number of discrete issues: infringement of claim 21 of the '872 patent against Intel, Intel's intentional copying of the patented inventions, validity of the '872 and '094 patents in view of the SONIC prior art reference, and Defendants' claims of inequitable

conduct.[1] Defendants and Intervenors (collectively, Defendants) oppose the motion and affirmatively move for summary judgment on the issues of damages, non-infringement of the '313 patent, anticipation of certain claims of the '872 and '094 patents by the Intel 82593 prior art reference, anticipation of all asserted claims of the '872 and '094 patents by the SONIC prior art reference, non-infringement of the '459 patent, invalidity of certain '313 and '459 patent claims under § 112, and issues specific to AT&T Services (ATTS), Marvell (MSI), Apple, and Atheros/Sigma/ATTS. The motions were heard on August 14, 2014. Having considered the parties' submissions and the arguments of counsel, the Court grants some motions and denies others, and grants some motions in part.

BACKGROUND

3Com Corporation, USEI's predecessor-in-interest, developed ethernet technology in the 1980s and 1990s. In the early 1990s, 3Com obtained the four patents-in-suit: U.S. Patent Nos. 5,434,872 (the '872 patent) (Apparatus for automatic initiation of data transmission), 5,732,094 (the '094 patent) (Method for automatic initiation of data transmission), 5,307,459 (the '459 patent) (Network adapter with host indication optimization), and 5,299,313 (the '313 patent) (Network interface with host independent buffer management).

[1] Defendants have withdrawn their inequitable conduct claims. Thus, the Court need not address this issue.

The patents-in-suit relate to the field of network interface controllers/adapters for managing the transmission and reception of data between a host computer system and a communication network. '313 patent, Field of the Invention. Typically, these controllers manage the transfer of data between the host computer system and the communication network, relieving the host computer to perform other tasks. Id., Description of Related Art. Using such a network controller, a sender may transmit "packets" or "frames" of data across a communication network. '872 patent, Description of Related Art. The frames of data must be organized according to the network "protocol" before transmission. Id. Some network adapters include dedicated transmit buffers, which can download data of a frame before they are transmitted. Id. A dedicated transmit buffer allows the host computer system to attend to other tasks while the data frame is being processed and transmitted. Id. If frame transmission is cancelled, the data may be retained in the transmit data buffer until the sending system tries to transmit the frame again. Id.

On October 9, 2009, USEI filed suit in the Eastern District of Texas against sixteen computer maker defendants,[2] alleging that they were manufacturing and selling desktop and laptop computers

[2] Acer, Inc., Acer America Corporation, Apple, Inc., ASUS Computer International, Asustek Computer, Inc., AT&T Services, Inc., Dell, Inc., Fujitsu Ltd., Fujitsu America, Inc., Gateway, Inc., Hewlett Packard Co., Sony Corporation, Sony Corporation of America, Sony Electronics, Inc., Toshiba Corporation, Toshiba America, Inc., and Toshiba America Information Systems, Inc.

which incorporated chips supplied by others that practice certain ethernet technology, thereby infringing the four patents-in-suit. That case was subsequently transferred to this district and given Case No. 10-3724 (the Acer case). On March 10, 2010, USEI filed a separate suit in the Eastern District of Texas against the retailer Defendants,[3] alleging infringement of the same four patents-in-suit. That case was subsequently transferred to this district and given Case No. C 10-5254 (the AT&T case). Chip suppliers, including Intel, Marvell, Atheros, and Sigma, who designed and provided chips to Defendants, successfully moved to intervene in both cases. The cases against the retailer Defendants and ATTS were stayed.

The Court has issued two claim construction orders. See Case No. C 10-3724, Docket Nos. 586 and 634; Case No. C 10-5254, Docket Nos. 331 and 379.

LEGAL STANDARDS

Summary judgment is appropriate only where the moving party demonstrates there is no genuine dispute as to any material fact such that judgment as a matter of law is warranted. Fed. R. Civ. P. 56(a); Celotex Corp. v. Catrett, 477 U.S. 317, 323 (1986). Material facts are those that might affect the outcome of the case, as defined by the framework of the underlying substantive

[3] AT&T, Inc., Barnes & Noble, Inc., Claire's Stores, Inc., J.C. Penney Company, Inc., Sally Beauty Holdings, Inc., and Home Depot U.S.A., Inc.

law. Anderson v. Liberty Lobby, Inc., 477 U.S. 242, 248 (1986). A dispute is genuine if the evidence is such that a reasonable jury could return a verdict for either party. Id.

The moving party bears the initial burden of informing the district court of the basis for its motion and identifying those portions of the pleadings, discovery, and affidavits that demonstrate the absence of a disputed issue of material fact. Celotex, 477 U.S. at 323. In opposing the motion, the non-moving party may not rely merely on the allegations or denials in its pleadings, but must set forth "specific facts showing that there is a genuine issue for trial." Anderson, 477 U.S. at 248 (citing Fed. R. Civ. P. 56(e)). The court must construe the evidence in the light most favorable to the non-moving party, making all reasonable inferences that can be drawn. Matsushita Elec. Indus. Co., Ltd. v. Zenith Radio Corp., 475 U.S. 574, 587 (1986); Intel Corp. v. Hartford Accident & Indem. Co., 952 F.2d 1551, 1558 (9th Cir. 1991); Eisenberg v. Ins. Co. of N. Am., 815 F.2d 1285, 1289 (9th Cir. 1987).

DISCUSSION

I. Anticipation of the asserted claims of the '872 and '094 patent by the SONIC prior art reference

A. USEI's motion

USEI moves for summary adjudication that the SONIC reference cannot anticipate the '872 and '094 patents (the "Early Transmit" patents), because it lacks a buffer memory of a certain minimum

size as required by the balance of the claim language. Defendants disagree, and cross-move for summary judgment that the asserted claims are anticipated by the SONIC prior art.

Patents are presumed valid, and the presumption may only be overcome by clear and convincing evidence. Microsoft Corp. v. i4i Ltd. P'ship, 131 S. Ct. 2238, 2252 (2011). "The burden of establishing invalidity of a patent . . . shall rest on the party asserting such invalidity." 35 U.S.C. § 282(a). Title 35 U.S.C. § 102 establishes the various grounds for invalidation of patents based on anticipation by prior art. "Anticipation requires the presence in a single prior art disclosure of all elements of a claimed invention arranged as in the claim." Therasense, Inc. v. Becton, Dickinson and Co., 593 F.3d 1325, 1333 (Fed. Cir. 2010) (quoting Connell v. Sears, Roebuck & Co., 722 F.2d 1542, 1548 (Fed. Cir. 1983)).

There is no dispute that SONIC is prior art pursuant to 35 U.S.C. § 102. SONIC is referred to in the Description of Related Art sections of the '872 and '094 patents as "representative prior art." There is also no dispute that the SONIC device contains a thirty-two byte buffer memory, not big enough to hold the smallest ethernet data frame, which is sixty-four bytes in size.

USEI contends that the plain language of the '872 and '094 patents demonstrates that the buffer memory must be big enough to hold an entire ethernet frame. For example, USEI points to claim 1 of the '872 patent, which calls for: "initiating transmission of

the frame prior to transfer of all the data of the frame to the buffer memory from the host computer." USEI then points to claim 21, which requires "logic which initiates transmission of the frame from the buffer memory to the medium access controller prior to transfer of all of the data of the frame from the buffer memory, including logic which initiates transmission of the frame when no complete frame of data is present in the buffer memory." USEI finally refers to the Court's construction of "buffer memory" as "memory for the temporary storage of data" to argue that, taken as a whole, the buffer memory of the patented invention must be capable of holding a complete minimum-sized ethernet data frame from the host computer "all at one time."

USEI's argument is unpersuasive. First, the plain language of the claims says nothing about the buffer memory's ability to hold a complete frame of data. Instead, it states more or less the opposite -- that a complete frame need not be downloaded into buffer memory before transmission can occur. The claim language USEI points out supports only the notion that a complete download of a frame need not occur. In other words, the language does not require that the buffer memory have the capacity to store all of the data of a frame at once, only that the transmission of the frame away from the buffer memory is initiated before all of the data of the frame is fully transferred to the buffer memory. USEI's own evidence of the specification, noting the disadvantages of prior art using a dedicated transmit buffer that accommodated a

full data frame, actually undermines USEI's contention; if the specification criticized the use of full frame capacity, that would indicate that the invention itself did not require full frame capacity.

Moreover, the Court construed the term "buffer memory" to mean "a memory for temporary storage of data" and did not impose any extraneous storage requirements on buffer memory. USEI previously made a similar argument that "the buffer memory is able to retain a frame of data that has been transmitted." The Court rejected this argument, finding that such additional language would add a limitation that is not required by the specification. Docket No. 586 at 21. Similarly, USEI's present attempt to infer a size requirement is simply not justified by the claim language, the specification, or any other evidence.

USEI does not present any evidence to support a finding that SONIC does not anticipate the '872 and '094 patents. Therefore, summary judgment for USEI on this point is not warranted. Instead, as will be discussed below, summary adjudication of the issue is granted for Defendants.

B. Defendants' cross motion

Because Defendants also move affirmatively that SONIC anticipates the asserted claims of the '872 and '094 patents, the Court turns to the other arguments they advance, all of which they must prove by clear and convincing evidence in order to prevail. Microsoft Corp., 131 S. Ct. at 2252-53.

United States District Court
For the Northern District of California

To show anticipation, Defendants refer to Dr. Wicker's report, which provides invalidity analysis and invalidity claim charts that detail element-by-element how SONIC satisfies each and every asserted claim limitation in the '872 and '094 patents. See Constant Decl. Ex. 14, Wicker Rpt. at ¶¶ 554-675, Exhs 9, 11. This evidence is largely unrefuted by USEI. USEI instead pins its substantive defense on its contention that the balance of the claim language imposes a lower limit on the buffer memory capacity -- that the buffer memory, as claimed, must be large enough to hold a full-size ethernet data frame, which has a minimum size of sixty-four bytes; SONIC's thirty-two byte buffer memory cannot satisfy the "buffer memory" limitation as claimed. In essence, USEI does not dispute any relevant facts regarding the alleged anticipating SONIC prior art, but only disagrees over an interpretation of the claim language. This renders the anticipation issue one of claim construction, which is a question of law. Markman v. Westview Instruments, Inc., 517 U.S. 370, 384 (1996).

After reviewing the claim language and the specification, the Court finds no justification to require that the "buffer memory" of the "Early Transmit" patents be capable of holding of a complete minimum-sized ethernet data frame all at one time. As the Court detailed above, the plain language of the claims says nothing about the buffer memory's ability to hold a complete frame of data. Instead, it states that a complete frame need not be

downloaded into buffer memory before transmission can occur. Nothing in the specification contradicts this understanding. Accordingly, the Court grants Defendants' motion for summary adjudication that SONIC anticipates the asserted claims of the '872 and '094 patents. Because the Court finds that the asserted claims of the '872 patent are invalid due to anticipation by the SONIC prior art, the Court need not decide USEI's motion for summary judgment of infringement of claim 21 of the '872 patent.

II. Whether Intel intentionally copied the patented invention, precluding any equitable defense

USEI contends that ample evidence exists to show that Intel copied the patented technology, which should prevent Intel from asserting any equitable defense. See, e.g., A.C. Aukerman Co. v. R.L. Chaides Constr. Co., 960 F.2d 1020, 1033 (Fed. Cir. 1992) ("A patentee may also defeat a laches defense if the infringer 'has engaged in particularly egregious conduct which would change the equities significantly in plaintiff's favor.' Conscious copying may be such a factor weighing against the defendant.") USEI presents evidence that Intel began testing and discussing 3Com's ethernet technology in 1992. For example, USEI has proffered internal Intel emails discussing the value of these accused infringing technologies and how they might be implemented in upcoming products. See Nation Decl. Exs. E., F.

Intel disagrees, arguing that USEI's allegations concern the copying of a 3Com product (the Etherlink III) that USEI's own

experts could not even identify or describe, much less compare to the asserted claims. Further, Intel contends that USEI's experts never compared the asserted claims to the Intel product (the 82595TX) that was alleged to be the result of Intel's intentional copying.

In order to establish that Intel intentionally copied the patented inventions, USEI must first show that the product allegedly copied by Intel embodied the asserted claims. Without such a showing, the allegations of copying are irrelevant. See, e.g., Amazon.com, Inc. v. Barnesandnoble.com, Inc., 239 F.3d 1343, 1366 (Fed. Cir. 2001) ("[E]vidence of copying Amazon's '1-Click' feature is legally irrelevant unless the '1-click' feature is shown to be an embodiment of the claims.").

The proffered internal Intel emails refer to "Parallel Tasking," a 3Com marketing term describing a commercial product (the "EtherLink III" adapter) that 3Com released in 1992. USEI fails to provide any evidence to show that the "Parallel Tasking" EtherLink III adapter embodies each and every element of the asserted claims or practices them. USEI's experts admit that they have not looked into whether the EtherLink III practiced any of the asserted claims. See Constant Decl. Ex. 6, Mitzenmacher Depo. at 290:16-293:19; Ex. 10, Conte Depo. at 79:1-24. Without such a showing, to the extent that USEI's motion asserts intentional copying, it must be denied. USEI also contends, in this motion, that Defendants, including Intel, engaged in litigation misconduct

by paying critical fact witnesses to prevent them from speaking to USEI. The Court has already denied sanctions regarding these actions and addressed accessibility concerns regarding this witness. Docket Nos. 866, 900. Thus USEI's request to bar Intel from asserting any equitable defense is denied.

III. Non-infringement of the '313 patent by any of the accused products

Claims 1 and 13 are the only asserted independent claims of the '313 patent. Defendants move for summary adjudication that no accused product infringes claim 1 of the '313 patent because (1) none of the accused products practices "network interface means"; (2) none of the accused products satisfies the requirement of "buffer memory outside of the host address space;" and (3) none of the accused products practices "host interface means." Defendants further contend that no accused product infringes claim 13 because USEI failed to identify any accused product that includes the structures identified by the Court as corresponding to the "host interface means" or the "network interface means" limitation.

A. "Network interface means" in claim 1

Claim 1 of the '313 patent reads:

> An apparatus for controlling communication between a host system and a network transceiver coupled with a network, wherein the host system includes a host address space, comprising:
>
> a buffer memory outside of the host address space;

> host interface means, sharing the host address space with the host, for managing data transfers between the host address space and the buffer memory in operations transparent to the host system; and
>
> network interface means, coupled with the network transceiver, for managing data transfers between the buffer memory and the network transceiver.

Defendants contend that none of the accused products practices "network interface means" as construed by the Court.

Claim 1 of the '313 patent recites "network interface means, coupled with the network transceiver, for managing data transfers between the buffer memory and the network transceiver." The Court found that "network interface means" is a means-plus-function term governed by § 112 ¶ 6. First Claim Construction Order, Docket No. 586 at 12. The Court construed "network interface means" to perform the function of "managing data transfers between the buffer memory and the network transceiver" and the corresponding structures to be "in Figure 3, Network interface logic 104, and equivalents." Second Claim Construction Order, Docket No. 634 at 17. Figure 3 of the '313 patent shows that the "network interface logic 104" is made up of "XMIT DMA LOGIC 109" and "RECEIVE DMA LOGIC 110." The Court elaborated on its construction by pointing to the specification, which states the "network interface logic 104 includes the transmit DMA logic 109 responsive to descriptors stored in the adaptor memory 103, for moving data out of the adapter memory to the network transceiver 105." Id.; '313 Patent, 10:3-11.

Defendants contend that the accused infringing products do not "transmit DMA logic responsive to descriptors stored in the adaptor memory" because all accused products move data from the buffer memory onto the network irresponsive to transmit descriptors stored in adapter memory. Kunz Decl. at ¶¶ 19-23; Carkin Decl. at ¶ 7; McCauley Decl. Ex. 1, Lin Rpt. at ¶¶ 156-169; Hu Decl. Ex. 1, Lin Rpt. for Sigma at ¶¶ 81-91, 152-162. USEI responds that, to prove infringement under the Court's construction, all it needs is to identify a "network interface logic 104" that performs the function of "managing transfers of data from buffers in the independent memory 103 and the network transceiver 105"; it does not need to identify further the corresponding structure to "transmit DMA logic 109 responsive to descriptors stored in adapter memory for moving data out of the adapter memory to the network transceiver."

Thus, the debate hinges on what is required to show infringement of "network interface means," a term the Court found to be governed by § 112 ¶ 6. Literal infringement of a claim limitation governed by § 112 ¶ 6 requires that the relevant structure in the accused device (1) perform the identical function recited in the claim, and (2) be identical or equivalent to the corresponding structure in the specification. See Applied Med. Resources Corp. v. United States Surgical Corp., 448 F.3d 1324, 1333 (Fed. Cir. 2006). Defendants do not dispute that the identified structures in the accused products perform the

identical function of "managing data transfers between the buffer memory and the network transceiver." Defendants also do not dispute that the corresponding structure in the specification is the network interface logic 104 shown in Fig. 3, which includes transmit DMA logic 109 that is responsive to descriptors stored in the adapter memory for moving data out of the adapter memory to the network transceiver. In fact, Dr. Mitzenmacher confirmed that the Court's construction requires just such logic. Constant Decl. Ex. 6, Mitzenmacher Depo. at 163:21-164:13. Instead, the parties dispute whether the identified structures in the accused products are identical or equivalent to the corresponding structure in the specification. In this regard, Dr. Mitzenmacher fails to proffer any evidence that the identified structures in the accused products are identical or equivalent to the corresponding structure identified by the Court. For example, in his initial Intel Report, Dr. Mitzenmacher identifies the MAC core component in Intel Gigabit products as the "network interface means" but fails to identify any DMA logic that is "responsive to descriptors stored in the adapter memory for moving data out of the adapter memory to the network transceiver." Constant Decl. Ex. 7, Mitzenmacher Intel Rpt. at ¶ 239. In fact, Dr. Mitzenmacher concedes that he did not specifically identify any such DMA logic in the MAC core component. Nation Decl. Ex. 41, Mitzenmacher 5/30 Depo. at 377:20-381:3. Likewise, Dr. Mitzenmacher's report for MSI makes no mention of any structure in the accused products that

is "responsive to descriptors stored in adapter memory for moving data out of adapter memory to a network transceiver." Flynn-O'Brien Decl. Ex. 6, Mitzenmacher Marvell Rpt. at ¶ 69, Ex. 8, Mitzenmacher Depo. at 626:12-22. As an alternative, USEI refers to Dr. Mitzenmacher's Supplemental Reports on how the structures identified in his initial reports are equivalent to the "transmit DMA logic 109 that is responsive to descriptors stored in the adapter memory for moving data out of the adapter memory to the network transceiver." However, the Court has not allowed these supplemental reports. As a result, USEI is foreclosed from arguing that the structures identified in Dr. Mitzenmacher's initial reports are equivalent to the "transmit DMA logic 109." Accordingly, because no structure in any accused product infringes the "network interface means" limitation of claim 1 of the '313 patent, summary judgment of non-infringement by any of the accused products of claim 1 of the '313 patent is appropriate.

B. "Buffer memory outside of host address space" in claim 1

Defendants contend that USEI fails to make a showing that the limitation "a buffer memory outside of the host address space" is met because the identified buffer memory in the accused Intel products can be accessed by the host under some circumstances, which implies that the buffer memory is not "outside of the host address space."

In his infringement report against Intel, Dr. Mitzenmacher identifies first-in-first-out buffers (FIFOs) in the accused Intel

products as the alleged "buffer memory outside of the host address space" recited in claim 1. Constant Decl. Ex. 7, Mitzenmacher Intel Rpt. at ¶ 230. However, Dr. Mitzenmacher concedes that "the Intel technical documentation for at least some of these products allows for direct addressing to the data FIFO buffers by the host specifically for diagnostic purposes." Id. at ¶ 231. These FIFOs in the accused Intel products do not meet the "buffer memory outside of the host address space" requirement because they are accessible by the host system and, thus, lie within the host address space. USEI contends that these FIFOs are only accessible by the host during diagnosis but are inaccessible to the host during normal data transmission, thereby rendering them "outside of the host address space" during the relevant time. Id. USEI's contention fails as a literal infringement argument; the buffer memory can either be inside or outside the host address space, but not both. To the extent that USEI argues that the FIFOs in the accused Intel products are equivalent to "buffer memory outside of the host address space," that is a doctrine of equivalents argument, one for which USEI has failed to proffer any evidence. Therefore, summary judgment of non-infringement of claim 1 of the '313 patent by the accused Intel products is appropriate for this reason as well.

Defendants further contend that Dr. Mitzenmacher improperly identifies two distinctly different "buffer memories" in each accused product for different limitations in claim 1, even though

the claim only mentions one "buffer memory." For example, Dr. Mitzenmacher identifies "packet buffer data FIFO buffers" in the accused Intel products as the "buffer memory outside the host address," but identifies "transmit and receive descriptor ring buffers" as the "buffer memory" of the "host interface means." Id. at 230, 233. Similarly, for each accused MSI product, Dr. Mitzenmacher identifies the "MAC Rx and Tx FIFOs" as the "buffer memory outside of the host address space," but the combination of the "MAC Rx and Tx FIFOs" and the "PFU Rx and Tx FIFOs" as the "buffer memory" of the "host interface means." Flynn-O'Brien Decl. Ex. 6, Mitzenmacher Marvell Rpt. at ¶¶ 61-67. USEI does not dispute this showing, but instead argues that this is covered by claim 1.

Claim 1 only recites one buffer memory in the "host interface means," which is for "managing data transfers between the host address space and the buffer memory," a reference to the "buffer memory outside of the host address space." Therefore, to show literal infringement, USEI must identify one "buffer memory" that performs the claim function. To the extent that USEI argues that the two identified buffer memories are equivalent to "the buffer memory" recited in the claim, that requires a showing of equivalency, for which USEI has failed to proffer any evidence. Therefore, summary judgment of non-infringement of claim 1 of the '313 patent by the accused Intel and MSI products is appropriate for this reason as well.

C. "Host interface means" in claim 1

Claim 1 contains the limitation of "host interface means," which the Court construed as having the function of "managing data transfer between address spaces on the host system bus and the buffer memory in operations performed independently of management by the host system." First Claim Construction Order, Docket No. 586 at 10-11. Defendants contend that the Court's construction that the data transfer must be "performed independently of management by the host system" dictates that the host system cannot be involved in data transfers. As a result, Defendants argue no infringement can be shown because the host systems in all accused products are involved in managing the data transfers between the host system and the network adapter by writing descriptors to a queue in the host system's memory (i.e., the "host address space.") Defendants' Motion and Opposition, Docket No. 1167-3 at 17.

Because, as discussed above, the Court finds that claim 1 of the '313 patent is not infringed for other reasons, the Court need not address this issue.

D. "Host interface means" in claim 13

Defendants contend that USEI fails to apply the Court's construction of the "host interface means" in claim 13 of the '313 patent to identify the relevant structures in the accused products and, instead, improperly applies the construction from claim 1.

Claim 13 reads:

> An apparatus for controlling communication between a host system and a network transceiver coupled with a network, wherein the host system includes a host address space, comprising:
>
> a buffer memory outside of the host address space, including a transmit buffer and a receive buffer;
>
> host interface means, sharing host address space including a prespecified block of host addresses of limited size defining a first area and a second area, and coupled with the buffer memory, for mapping data addressed to the first area into the transmit buffer, mapping data in the receive buffer into the second area, and uploading data from the receive buffer to the host; and
>
> network interface means, coupled with the network transceiver and the buffer memory, for transferring data from the transmit buffer to the network transceiver and mapping data into the receive buffer from the network transceiver.

Unlike claim 1, which recites a "host interface means" that performs a single function of "managing data transfers between the host address space and the buffer memory in operations transparent to the host system," claim 13 recites three functions for the "host interface means," namely, "mapping data addressed to the first area into the transmit buffer," "mapping data in the receive buffer into the second area," and "uploading data from the receive buffer to the host." Second Claim Construction Order, Docket No. 634 at 16. The Court identified separate corresponding structures for each identified function, including an XMIT AREA register, transfer descriptor logic; an XFER AREA register, upload logic; and an upload DMA module, respectively. Id.

USEI fails to base its infringement analysis on the Court's construction for claim 13, but instead applies the construction from claim 1. See Constant Decl. Ex. 53, Mitzenmacher Intel Base Rpt. at ¶ 250. USEI contends that Dr. Mitzenmacher's reports identify structures in the accused products that perform the functions recited by the Court for the "host interface means" element of the '313 patent. Nation Decl. Ex. 35, Mitzenmacher 5/29 Depo. at 168:6-172:14, 175:8-21. However, the record does not support USEI's contention. Claim 13 clearly recites three functions for "host interface means" but USEI fails specifically to identify structures in the accused device that perform these functions. Literal infringement of a claim limitation governed by § 112 ¶ 6 requires that the relevant structure in the accused device (1) perform the identical function recited in the claim, and (2) be identical or equivalent to the corresponding structure in the specification. Applied Med. Resources, 448 F.3d at 1333. USEI fails to show even the first requirement that the relevant structure in the accused device perform the identical function recited in the claim, let alone the second requirement that the relevant structures be identical or equivalent to the corresponding structures. Therefore, USEI fails to show an issue of fact regarding infringement of claim 13 of the '313 patent and, thus, summary judgment of non-infringement of that claim by any of the accused products is warranted.

IV. Anticipation of claim 21 of the '872 patent and claims 9, 28, and 39 of the '094 patent by the Intel 82593 prior art reference

Defendants move for partial summary adjudication of anticipation under 35 U.S.C. § 102(a) and (b), contending that Intel's 82593 chip is prior art that satisfies every limitation of claim 21 of the '872 patent and claims 9, 28, 39 of the '094 patent. However, as discussed above, the Court grants summary adjudication of invalidity of all the asserted claims of the '872 and '094 patents due to the SONIC prior art reference. Thus, the Court need not address this issue.

V. Non-infringement of the '459 patent by any of the accused products

Claims 1 and 44 are the only asserted independent claims of the '459 patent, and they are asserted only against Intel (claim 1) and HP (claims 1 and 44).

A. "Means for comparing"

Defendants contend that USEI's infringement reports regarding Intel and HP fail to establish infringement because they ignore that the "means for comparing" recited in the claims is a means-plus-function element governed by § 112 ¶ 6 and do not address any of the corresponding structures in the '459 patent that the Court identified for such means (e.g., blocks 224 and 318, interrupt controller 60), much less show that such structures are satisfied by the accused Intel or HP products. Constant Decl. Ex. 12, Crayford Rpt. at ¶¶ 859-861.

As noted above, literal infringement of a claim limitation governed by § 112 ¶ 6 requires that the relevant structure in the accused device (1) perform the identical function recited in the claim, and (2) be identical or equivalent to the corresponding structure in the specification. Applied Med., 448 F.3d at 1333.

Claim 1 of the '459 patent reads:

> An apparatus for transferring a data frame between a network transceiver, coupled with a network, and a host system which includes a host processor and host memory, the apparatus generating an indication signal to the host processor responsive to the transfer of the data frame, with the host processor responding to the indication signal after a period of time, comprising:
>
> a buffer memory for storing the data frame;
>
> network interface logic for transferring the data frame between the network transceiver and the buffer memory;
>
> host interface logic for transferring the data frame between the host system and the buffer memory;
>
> threshold logic for allowing the period of time for the host processor to respond to the indication signal to occur during the transferring of the data frame, wherein the threshold logic includes,
>
> a counter, coupled to the buffer memory, for counting the amount of data transferred to or from the buffer memory;
>
> an alterable storage location containing a threshold value; and
>
> means for comparing the counter to the threshold value in the alterable storage location and generating an indication signal to the host processor responsive to a comparison of the counter and the alterable storage location.

Claim 44 contains more or less the same language for "means for comparing."

The Court construed "means for comparing" to have two functions: (1) "comparing the counter to the threshold value in the alterable storage location"; and (2) "generating an indication signal to the host processor." The Court found that the block labeled 224 in Fig. 14 and 318 in Fig. 21 performs the first function. Second Claim Construction Order, Docket No. 634 at 10-11. The Court found that the functional components labeled "Interrupt Controller 60" shown in Fig. 4, together with "Early Rcv Control 225" in Fig. 14, perform the second function. Id.

USEI argues that Dr. Mitzenmacher "identifies structures performing the required functions taking into account the Court's claim construction in his original Infringement Reports," but his report merely identifies a portion of the Intel source code of the accused products performing the comparison function without identifying any structure corresponding to the structures specified in the Court's construction. Constant Decl. Ex. 7, Mitzenmacher Intel Rpt. at ¶ 210; Constant Decl. Ex. 12, Crayford Rpt. at ¶¶ 859-61. Similarly, for the accused HP products, Dr. Mitzenmacher simply opines that the recited function is performed, without regard to the corresponding structure required by the Court's claim construction. Declaration of Cameron A. Zinsli (Zinsli Decl.), Ex. 2, Mitzenmacher HP Rpt. at ¶¶ 119, 120. This is not sufficient; USEI must identify structures in the accused product that are either identical or equivalent to the corresponding structures identified by the Court. Applied Med.,

448 F.3d at 1333. As an alternative, USEI points to Dr. Mitzenmacher's Supplemental Reports for evidence that the proffered source code is at least equivalent to that required by the Court. However, these reports have been excluded. Therefore, USEI is foreclosed from arguing that the identified source code is equivalent to the corresponding structures required by the Court. Accordingly, summary judgment of non-infringement of claims 1 and 44 of the '459 patent by the accused Intel and HP products is granted.

B. "Look-ahead threshold logic"

Intel separately contends that claim 1 of the '459 is not infringed because Intel's accused products do not satisfy "means for comparing" in that they do not contain the "look-ahead threshold logic" required by the Court's construction. However, as discussed above, given that summary adjudication of non-infringement of claims 1 and 44 of the '459 patent by the accused Intel and HP products is granted, the Court need not address whether Intel's products do not infringe claim 1 for this additional reason.

C. "Threshold value"

In addition, Defendants contend that claims 1 and 44 of the '459 patent are not infringed because the accused products do not satisfy "an alterable storage location containing a threshold value" limitation in claim 1 and "an alterable storage location containing a transfer threshold value" limitation in claim 44.

Again, as discussed above, given that summary adjudication of non-infringement of claims 1 and 44 of the '459 patent by the accused Intel and HP products is granted for other reasons, the Court need not decide this issue.

VI. Invalidity under § 112 of claim 13 of the '313 patent and claim 1 of the '459 patent

Defendants contend that nothing in the '313 or '459 patents suggests a single structure for performing the functions recited for the "host interface means" of claim 13 of the '313 patent and the "means for comparing" of claim 1 of the '459 patent, and ask the Court to invalidate the claims as indefinite under 35 U.S.C. § 112. However, as discussed above, the Court grants summary adjudication of non-infringement of claim 13 of the '313 patent and claim 1 of the '459 patent. Thus, the Court need not decide this issue.

VII. Exclusion of the unreliable opinion of USEI's damages expert, Walter Bratic

Defendants contend that USEI cannot prove damages, because its expert, Mr. Bratic, used unreliable methods, unreasonable inferences and speculation to reach his damages conclusions and, thus, his testimony should be excluded.

However, given the findings in this order, this motion is moot, and the Court need not address it.

VIII. ATTS' motion for partial summary judgment of no damages or, alternatively, to limit the AT&T royalty base

ATTS moved for summary judgment to limit the AT&T royalty base to the number of Sigma chips purchased during the relevant

time period. Given the other findings in this order, this motion is moot, and the Court need not address it.

IX. MSI's motion for partial summary judgment to exclude foreign non-party sales from USEI's asserted damages base

MSI contends that the damages USEI claims against it are improper in that they include Yukon chips that are manufactured abroad by third-party foundries or offered for sale and sold abroad by non-party Marvell Asia Pte. Ltd. (MAPL), a Singapore corporation that is a legal entity distinct from MSI. Declaration of Joseph Kuo (Kuo Decl.) at ¶¶ 9, 10. However, given the other findings in this order, the Court need not address this issue.

X. Apple's motion for partial summary judgment of non-infringement of claims 1, 9, 12 and 28 of the '094 patent

USEI accuses certain Apple products of infringing method claims 1, 9, 12, and 28 of the '094 patent based solely on its inclusion of GEM ethernet technology supplied by Sun Microsystems. Declaration of Christopher Cravey (Cravey Decl.) Ex. 1, Mitzenmacher Apple Rpt. at ¶¶ 52-84. Apple contends that it does not infringe these method claims because it has always disabled the accused infringing "Early Transmit" feature in the Sun GEM ethernet technology, thus rendering use of the accused infringing feature impossible. Cravey Decl. Ex. 3, Seifert Rpt. at ¶¶ 48-52, 54-57, and 64-69.

Because the Court grants summary adjudication of invalidity of all the asserted claims of the '094 patent due to the SONIC prior art reference, the Court need not address this issue.

XI. Atheros, Sigma and ATTS's motion for partial summary judgment of non-infringement of the '313 patent

In construing the '313 patent's claim 1 element, "host interface means," the Court found that the function of "managing data transfers between the host address space and the buffer memory in operations transparent to the host system" necessarily includes the function of remapping a section of the host address space in the host system to the buffer memory. Second Claim Construction Order, Docket No. 634 at 13. Atheros, Sigma and ATTS contend that Dr. Mitzenmacher's report does not provide any analysis to indicate that these Defendants' accused products practice the claims under the Court's construction. Sigma and ATTS argue that the same is true for the claims dependent on claim 1 in the '313 patent, as well as claim 13 and its dependent claims.

However, as discussed above, the Court grants summary adjudication of non-infringement of the asserted claims of the '313 patent by any of the accused products. Thus, the Court need not address this issue.

CONCLUSION

As discussed above, the Court has adjudicated the status of each asserted claim of the patents-in-suit as follows:

The Court (1) DENIES USEI's motion for summary adjudication of non-anticipation of the asserted claims of the '872 or '094 patents by the SONIC prior art reference, and instead GRANTS

Defendants' motion for invalidity of the asserted claims of the '872 or '094 patents in view of the SONIC prior art reference; (2) DENIES USEI's motion for summary judgment against Intel, and Defendants using Intel chips, of infringement of claim 21 of the '872 patent as moot; and (3) DENIES USEI's motion for summary judgment that Intel may not present any equitable defense because it intentionally copied USEI's patented invention.

With regard to Defendants' motions for summary judgment, the Court GRANTS the following motions: (1) for summary adjudication that SONIC anticipates the asserted claims of the '872 and '094 patents; (2) for summary adjudication of non-infringement of claims 1 and 13 of the '313 patent by any of the accused products; and (3) for summary adjudication of non-infringement of claims 1 and 44 of the '459 patent by the Intel and HP accused products.

The Court DENIES the following Defendants' motions as moot: (1) to exclude Mr. Bratic's expert testimony regarding all four patents-in-suit as moot; (2) for summary adjudication of anticipation of claim 21 of the '872 patent by the Intel 82593 chip prior art; (3) for summary adjudication of anticipation of claims 9, 28 and 39 of the '094 patent by the Intel 82593 chip prior art; (4) for summary judgment of invalidity, under § 112, of claim 13 of the '313 patent and claim 1 of the '459 patent; (5) MSI's motion for partial summary judgment to exclude foreign non-party sales from USEI's asserted damages base; (6) Apple's motion for summary adjudication of non-infringement of claims 1,

9, 12 and 28 of the '094 patent; and (7) Atheros, Sigma and ATTS's motion for partial summary judgment of non-infringement of the '313 patent.

Accordingly, all of the asserted claims of the patents-in-suit are resolved in favor of Defendants. The Clerk of the Court shall enter judgment in favor of Defendants, who shall recover costs from USEI.

IT IS SO ORDERED.

Dated: November 7, 2014

CLAUDIA WILKEN
United States District Judge

United States District Court
For the Northern District of California

Exhibit 4

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

U.S. ETHERNET INNOVATIONS, LLC,

Plaintiff,

v.

ACER, INC.; ACER AMERICA CORPORATION; APPLE, INC.; ASUS COMPUTER INTERNATIONAL; ASUSTEK COMPUTER INC.; DELL; FUJITSU LTD.; FUJITSU AMERICA, INC.; GATEWAY, INC.; HEWLETT PACKARD CO.; SONY CORPORATION; SONY CORPORATION OF AMERICA; SONY ELECTRONICS INC.; and AT&T MOBILITY, LLC,

Defendants,

INTEL CORPORATION; MARVELL SEMICONDUCTOR, INC.; ATHEROS COMMUNICATIONS, INC., and SIGMA DESIGNS, INC.,

Intervenors.

Case No. 10-cv-03724-CW

JUDGMENT

Re: Dkt. No. 1289

For the reasons set forth in this Court's Order on Summary Judgment Motions,

IT IS ORDERED AND ADJUDGED

That judgment is hereby entered in favor of Defendants Acer, Inc., ACER America Corporation, Apple, Inc., Asus Computer International, Asustek Computer Inc., Dell Inc., Fujitsu Ltd., Fujitsu America, Inc., Gateway, Inc., Hewlett Packard Company, Sony Corporation, Sony Corporation of America, Sony Electronics Inc., and AT&T Mobility, LLC and Intervenors Intel Corporation, Marvell Semiconductor, Inc., Atheros Communications, Inc., and Sigma Designs, Inc. These parties shall recover from Plaintiff U.S. Ethernet Innovations, LLC their costs of action.

Dated at Oakland, California, this 1st day of December, 2014.

Richard W. Wieking
Clerk, United States District Court

By: _____
Nikki D. Riley, Deputy Clerk to the
Honorable CLAUDIA WILKEN

United States District Court
Northern District of California

Exhibit 5

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

U.S. ETHERNET INNOVATIONS, LLC,

Plaintiff,

v.

ACER, INC.; ACER AMERICA CORPORATION; APPLE, INC.; ASUS COMPUTER INTERNATIONAL; ASUSTEK COMPUTER INC.; DELL; FUJITSU LTD.; FUJITSU AMERICA, INC.; GATEWAY, INC.; HEWLETT PACKARD CO.; SONY CORPORATION; SONY CORPORATION OF AMERICA; SONY ELECTRONICS INC.; TOSHIBA CORPORATION; TOSHIBA AMERICA INFORMATION SYSTEMS, INC.; and AT&T MOBILITY, LLC,

Defendants,

INTEL CORPORATION; MARVELL SEMICONDUCTOR, INC.; ATHEROS COMMUNICATIONS, INC., and SIGMA DESIGNS, INC.,

Intervenors.

Case No. 10-cv-03724-CW

AMENDED JUDGMENT

Re: Dkt. No. 1294

For the reasons set forth in this Court's Order on Summary Judgment Motions,

IT IS ORDERED AND ADJUDGED

That judgment is hereby entered in favor of Defendants Acer, Inc., ACER America Corporation, Apple, Inc., Asus Computer International, Asustek Computer Inc., Dell Inc., Fujitsu Ltd., Fujitsu America, Inc., Gateway, Inc., Hewlett Packard Company, Sony Corporation, Sony Corporation of America, Sony Electronics Inc., Toshiba Corporation, Toshiba America Information Systems, Inc., and AT&T Mobility, LLC and Intervenors Intel Corporation, Marvell

Semiconductor, Inc., Atheros Communications, Inc., and Sigma Designs, Inc. These parties shall recover from Plaintiff U.S. Ethernet Innovations, LLC their costs of action.

Dated at Oakland, California, this 2nd day of December, 2014.

Richard W. Wieking
Clerk, United States District Court

By: _____
Nikki D. Riley, Deputy Clerk to the
Honorable CLAUDIA WILKEN

United States District Court
Northern District of California

Exhibit 6

United States District Court
For the Northern District of California

IN THE UNITED STATES DISTRICT COURT

FOR THE NORTHERN DISTRICT OF CALIFORNIA

U.S. ETHERNET INNOVATIONS, LLC,

Plaintiff,

v.

ACER, INC., et al.,

Defendants,

and

ATHEROS COMMUNICATIONS, INC., et al.,

Intervenors.

_____/

No. C 10-3724 CW

ORDER DENYING MOTION TO ALTER OR AMEND SUMMARY JUDGMENT ORDER

(Docket No. 1344)

U.S. ETHERNET INNOVATIONS, LLC,

Plaintiff,

v.

AT&T MOBILITY, LLC, et al.,

Defendants.

_____/

No. C 10-5254 CW

ORDER DENYING MOTION TO ALTER OR AMEND SUMMARY JUDGMENT ORDER

(Docket No. 591)

Plaintiff U.S. Ethernet Innovations, LLC (USEI) moves to alter or amend the Court's November 7, 2014 Summary Judgment Order. (Case No. 10-3724, Docket No. 1344; Case No. 10-5254, Docket No. 591). It argues that (1) Defendants and Intervenors are collaterally estopped from litigating the validity of the asserted claims of the '872 and '094 patents and (2) the Court committed clear error when it found in favor of Defendants and Intervenors with regard to the non-infringement of claims 1 and 13

of the '313 patent. Defendants and Intervenors oppose the motion. For the reasons stated below, the Court DENIES the motion.

BACKGROUND

The facts in this case are summarized in the Court's November 7, 2014 Order on Summary Judgment Motions. The following facts are those that are relevant to this motion.

3Com Corporation, USEI's predecessor-in-interest, developed ethernet technology in the 1980s and 1990s. In the early 1990s, 3Com obtained the four patents-in-suit: U.S. Patent Nos. 5,434,872 (the '872 patent) (Apparatus for automatic initiation of data transmission), 5,732,094 (the '094 patent) (Method for automatic initiation of data transmission), 5,307,459 (the '459 patent) (Network adapter with host indication optimization), and 5,299,313 (the '313 patent) (Network interface with host independent buffer management).

On October 9, 2009, USEI filed suit in the Eastern District of Texas against sixteen computer maker defendants,[1] alleging that they were manufacturing and selling desktop and laptop computers which incorporated chips supplied by others that practice certain ethernet technology, thereby infringing the four patents-in-suit. That case was subsequently transferred to this district and given

[1] Acer, Inc., Acer America Corporation, Apple, Inc., ASUS Computer International, Asustek Computer, Inc., AT&T Services, Inc., Dell, Inc., Fujitsu Ltd., Fujitsu America, Inc., Gateway, Inc., Hewlett Packard Co., Sony Corporation, Sony Corporation of America, Sony Electronics, Inc., Toshiba Corporation, Toshiba America, Inc., and Toshiba America Information Systems, Inc.

Case No. 10-3724. On March 10, 2010, USEI filed a separate suit in the Eastern District of Texas against retailer Defendants, [2] alleging infringement of the same four patents-in-suit. That case was also transferred to this district, and given Case No. 10-5254.

In 2011, USEI filed suit against Texas Instruments Incorporated (TI) in the Eastern District of Texas. Two jury trials were held in that case. On April 11, 2014, the first jury returned a verdict that the asserted claims of the '872 and the '094 patents were not invalid as anticipated by the SONIC prior art. On June 20, 2014, the second jury found that TI infringed the asserted claims of the '872 patent and awarded damages to USEI. On September 19, 2014, judgment was entered in the TI case, "with the exception of the parties' post-verdict briefing." Pl.'s Mot. to Alter or Amend, Docket No. 1344, Ex. E.

On November 7, 2014, this Court found on summary judgment in the above-entitled cases that: (1) the asserted claims of the '872 and '094 patents were invalid in view of the SONIC prior art reference and (2) claims 1 and 13 of the '313 patent were not infringed by any of the accused products.

On November 13, 2014, the Texas court ordered the parties to brief whether USEI was collaterally estopped from arguing that the asserted claims of the '872 and '094 patents were valid in the

[2] AT&T, Inc., Barnes & Noble, Inc., Claire's Stores, Inc., J.C. Penney Company, Inc., Sally Beauty Holdings, Inc., and Home Depot U.S.A., Inc.

United States District Court
For the Northern District of California

light of this Court's November 7, 2014 Order on Summary Judgment Motions. Briefing in the Texas court on the applicability of collateral estoppel ended on December 19, 2014. In the meantime, this Court entered final judgment against USEI on December 1, 2014.

USEI filed the present motion in this Court on December 29, 2014. Briefing on this motion ended on February 2, 2015. On February 19, 2015, the Eastern District of Texas ruled that, in the light of this Court's November 7, 2014 order invalidating the asserted claims of the '872 and '094 patents, USEI was collaterally estopped from litigating the validity of the '872 and '094 patents and from recovering infringement damages in the Texas litigation. See Docket No. 1370. Final judgment in that case was entered on February 24, 2015.

LEGAL STANDARD

"'Rule 59(e) amendments are appropriate if the district court (1) is presented with newly discovered evidence, (2) committed clear error or the initial decision was manifestly unjust, or (3) if there is an intervening change in controlling law.'" In re Syncor ERISA Litig., 516 F.3d 1095, 1100 (9th Cir. 2008)(citing Dixon v. Wallowa County, 336 F.3d 1013, 1022 (9th Cir. 2003)). Motions for reconsideration are not a substitute for appeal or a means of attacking some perceived error of the court. See Twentieth Century-Fox Film Corp. v. Dunnahoo, 637 F.2d 1338, 1341 (9th Cir. 1980).

DISCUSSION

I. Collateral Estoppel

USEI contends that Defendants and Intervenors should be collaterally estopped from litigating the validity of the '872 and '094 patents in this Court.

USEI's motion to alter or amend the judgment based on collateral estoppel is moot. On February 19, 2015, following the completion of briefing on this issue in this Court, the Eastern District of Texas ruled on this question. It held that USEI is collaterally estopped from litigating the validity of the asserted claims of the '872 and '094 patents, and receiving damages, in the light of this Court's December 1, 2014 judgment. See Docket No. 1370. This Court need not decide if collateral estoppel applies to Defendants and Intervenors in this case because the Eastern District of Texas has already decided that collateral estoppel applies to USEI. Accordingly, the Court denies USEI's motion to alter or amend its judgment with regard to the validity of the asserted claims of the '872 and '094 patents.

II. Non-infringement of the '313 Patent

USEI claims that the Court committed clear error when it found in favor of Defendants and Intervenors with regard to certain infringement claims.

The Ninth Circuit has not articulated a standard as to what constitutes "clear error" in connection with a Rule 59(e) motion for reconsideration. It has, however, defined "clear error" in

other contexts, which can provide guidance. For example, the Ninth Circuit stated that it would only find a trial court's factual finding "'clearly erroneous' when, although there is evidence to support it . . . on the entire evidence [it] is left with the definite and firm conviction that a mistake has been committed." United States v. Ruiz-Gaxiola, 623 F.3d 684, 693 (9th Cir. 2010). Thus, the Court will find "clear error" in connection with a Rule 59(e) motion only when it has the "definite and firm conviction" that a mistake has been committed. See also Joe Hand Promotions, Inc. v. Mujadidi, 2012 WL 4901429, at *1 (N.D. Cal.) ("[C]lear error should conform to a very exacting standard" by which "a final judgment must be 'dead wrong' to constitute clear error"); J & J Sports Prods., Inc. v. Juanillo, 2011 WL 335342, at *1 (N.D. Cal.) ("If a court 'got the law right' and 'did not clearly err in its factual determinations,' then clear error was not committed -- even if another reasonable judicial body 'would have arrived at a different result'"); Mitchell v. Asuncion, 2013 WL 2016136, at *1 (N.D. Cal.)("A district court does not commit clear error warranting reconsideration when the question before it is a debatable one"). The Court now turns to each of USEI's clear error contentions.

A. Non-infringement of claim 13 of the '313 patent based on the "host interface means" element

USEI argues,

> In this Court's recent summary judgment order, the Court granted summary judgment for Defendants based on the Court's finding that USEI's infringement expert failed to identify

> required structures for the host interface means element of claim 13 of the '313 Patent. A necessary premise of the Court's order is the prior construction of claim 13 to require certain structures that perform the recited functions of the element. USEI respectfully requests that the Court alter its summary judgment ruling because this Court has never construed the host interface means element of claim 13 to require particular structures. By basing its summary judgment ruling on the erroneous position that the required structures of this claim element had been construed, the Court's judgment for Defendants on this issue is itself clearly erroneous and should be altered in favor of USEI.

Pl.'s Mot. to Alter or Amend, Docket No. 1344-3 at 13. USEI, however, misstates the basis for this Court's summary judgment ruling. Contrary to USEI's argument, the Court did not base its conclusion on "the . . . position that the required structures of [claim 13 of the '313 patent] had been construed."

In its November 7, 2014 Order, the Court stated that finding literal infringement of a means plus function claim limitation governed by 35 U.S.C. § 112 ¶ 6 requires that the accused device perform the identical function recited in the claim, and be identical or equivalent to the corresponding structure in the specification. In the prior claim construction order, the Court identified three functions for the "host interface means" element in claim 13 of the '313 patent. See Second Claim Construction, Docket No. 634 at 16. The Court identified separate structures corresponding to each of the identified functions but found that "there [is] no single structure that is capable of performing all three functions" and noted that the "lack of corresponding structure renders Claim 13 of the '313 patent arguably invalid." Id. at 17.

To prevail on a summary judgment motion, USEI bore the burden to identify structures in the accused devices that performed each of the functions identified in this means plus function element. USEI failed to do so. While USEI contended that the reports of Dr. Mitzenmacher, its infringement expert, "identify structures in the accused products that perform the functions recited by the Court" for this element, the Court found that "the record does not support USEI's contention." November 7, 2014 Order at 21. In its motion to alter or amend, USEI again states that Dr. Mitzenmacher "correctly used his knowledge and skill to identify structures that perform the recited functions of the host interface means of claim 13 in his infringement reports." Docket No. 1344 at 14. However, USEI again fails to show that Dr. Mitzenmacher specifically identified these structures.

In its summary judgment order, the Court did not find it necessary even to reach the question of whether the structure of the accused device was identical or equivalent to any corresponding structure in the patent specification, nor the question of whether a lack of any such structure in the specification rendered the means plus function claim at issue invalid, as the claim construction order had posited.

Accordingly, USEI's argument that the Court committed clear error on this issue is unpersuasive. USEI's motion to alter or amend the Court's November 7, 2014 Summary Judgment Order on the

basis of clear error regarding non-infringement of claim 13 of the '313 patent based on the "host interface means" element is DENIED.

B. Non-infringement of claims 1 and 13 of the '313 patent based on the "network interface means" element

USEI argues,

> The Court also committed clear error in granting summary judgment of non-infringement of claims 1 and 13 of the '313 Patent based on an erroneous reading of the claim construction for the network interface means element of both claims. In construing claims 1 and 13, the Court described the required structure for the network interface means element as follows: "In Figure 3, Network interface logic 104, and its equivalents." Defendants argued during summary judgment that this element required additional structures, "transmit DMA logic, (generally 109) and receive DMA logic (generally 110)," because these DMA logic structures are described in the preferred embodiment of the patent as being included as sub-components of the network interface logic 104. Because these DMA sub-structures were never required by the Court's construction, the Court clearly erred both in agreeing with Defendants' interpretation and subsequently granting summary judgment for Defendants on this issue.

Docket No. 1344-3 at 15.

In its second claim construction order, the Court described the corresponding structures for the "network interface means" element of claim 1 and 13 of the '313 patent:

> The written description of the '313 Patent discloses that the "network interface logic 104" shown in Fig. 3 "manages transfers of data from buffers in the independent memory 103 and the network transceiver 105." ('313 Patent, Col. 9:55-59.) The written description further states: The network interface logic 104 includes transmit DMA logic, (generally 109) and receive DMA logic (generally 110). The transmit DMA logic 109 is responsive to descriptors stored in the adapter memory 103, as described below, for moving data out of the independent adapter memory 103 to the network transceiver 105. Similarly, the receive DMA logic 110 is responsible for moving data from the transceiver 105 into the independent adapter memory 103.

Docket No. 634 at 17. Accordingly, for an accused device to infringe the "network interface means" of claims 1 and 13 of the

United States District Court
For the Northern District of California

'313 patent, it must include "network interface logic 104," which, in turn, includes "transmit DMA logic" and "receive DMA logic." Therefore, USEI's contention that the Court's Second Claim Construction Order did not require structures that included DMA logic is incorrect.

USEI also argues that its infringement expert, Dr. Mitzenmacher, "properly identified the minimum structure that would perform network interface logic 104" as it believes it was construed by the Court. Given the discussion above, this argument is irrelevant because USEI does not claim that Dr. Mitzenmacher identified structures that performed the DMA logic.

Thus, USEI has failed to show that the Court committed clear error when it granted summary judgment on this ground in favor of Defendants and Intervenors. Accordingly, USEI's motion to alter or amend the Court's November 7, 2014 summary judgment order based on the "network interface means" element in claims 1 and 13 of the '313 patent is DENIED.

CONCLUSION

For the reasons stated above, USEI's motion to alter or amend the Court's November 7, 2014 Summary Judgment Order (Case No. 10-3724, Docket No. 1344; Case No. 10-5254, Docket No. 591) is DENIED.

IT IS SO ORDERED.

Dated: March 30, 2015

CLAUDIA WILKEN
United States District Judge

Exhibit 7

US005434872A

United States Patent [19]

Petersen et al.

[11] Patent Number: 5,434,872

[45] Date of Patent: Jul. 18, 1995

[54] **APPARATUS FOR AUTOMATIC INITIATION OF DATA TRANSMISSION**

[75] Inventors: **Brian Petersen**, Los Altos; **David R. Brown**, San Jose; **W. Paul Sherer**, Sunnyvale, all of Calif.

[73] Assignee: **3Com Corporation**, Santa Clara, Calif.

[21] Appl. No.: **920,893**

[22] Filed: **Jul. 28, 1992**

[51] **Int. Cl.**[6] **H04L 12/413**; H04L 25/36; G06F 13/00

[52] **U.S. Cl.** **371/57.1**; 364/DIG. 1; 364/239.51; 370/85.3; 395/200; 395/250

[58] **Field of Search** 371/57.1; 395/250, 200; 364/239.51; 370/85.1, 85.15, 85.3

[56] **References Cited**

U.S. PATENT DOCUMENTS

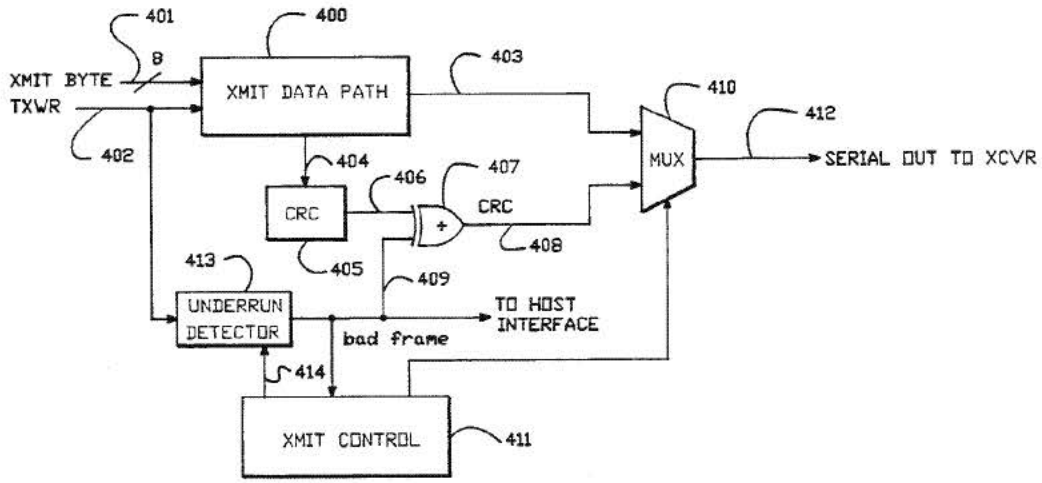
4,258,418	3/1981	Heath	395/250
4,715,030	12/1987	Koch et al.	370/85.13
4,860,193	8/1989	Bentley et al.	395/250
5,043,981	8/1991	Firoozmand et al.	370/85.1
5,195,093	3/1993	Tarrab et al.	371/3
5,210,749	5/1993	Firoozmand	370/85.1
5,278,956	1/1994	Thomsen et al.	395/250

Primary Examiner—Stephen M. Baker
Attorney, Agent, or Firm—Haynes & Davis

[57] **ABSTRACT**

Early initiation of transmission of data in a network interface that includes a dedicated transmit buffer is provided in a system which includes logic for transferring frames of data composed by the host computer into the transmit buffer. The amount of data of a frame which is downloaded by the host to the transmit buffer is monitored to make a threshold determination of an amount of data of the frame resident in the transmit data buffer. The network interface controller includes logic for initiating transmission of the frame when the threshold determination indicates that a sufficient portion of the frame is resident in the transmit buffer, and prior to transfer of all of the data of the frame into the transmit buffer. The monitoring logic includes a threshold store, which is programmable by the host computer for storing a threshold value. Thus, the threshold value may be set by the host system to optimize performance in a given setting.

26 Claims, 13 Drawing Sheets



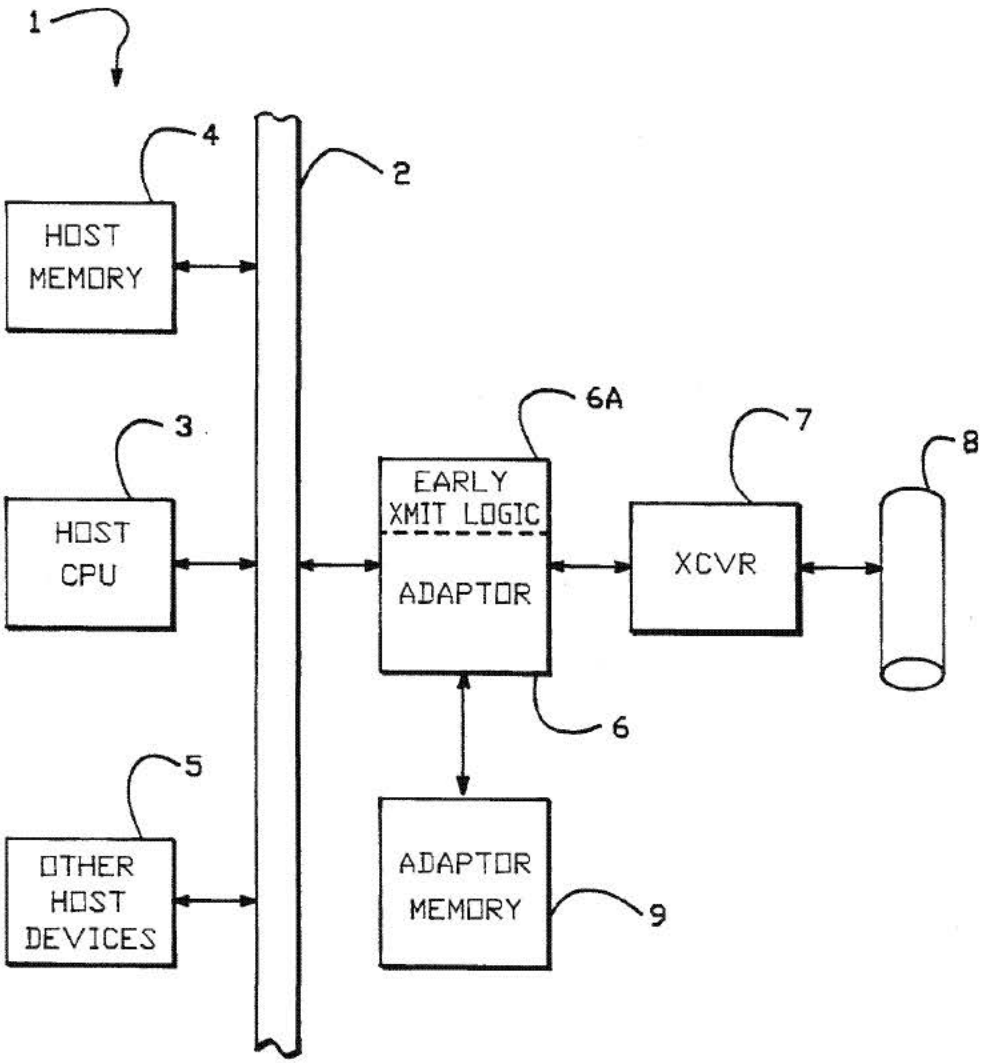

1
4
2
HOST MEMORY
3
HOST CPU
6A
EARLY XMIT LOGIC
ADAPTOR
7
XCVR
8
6
5
OTHER HOST DEVICES
ADAPTOR MEMORY
9

FIG.—1

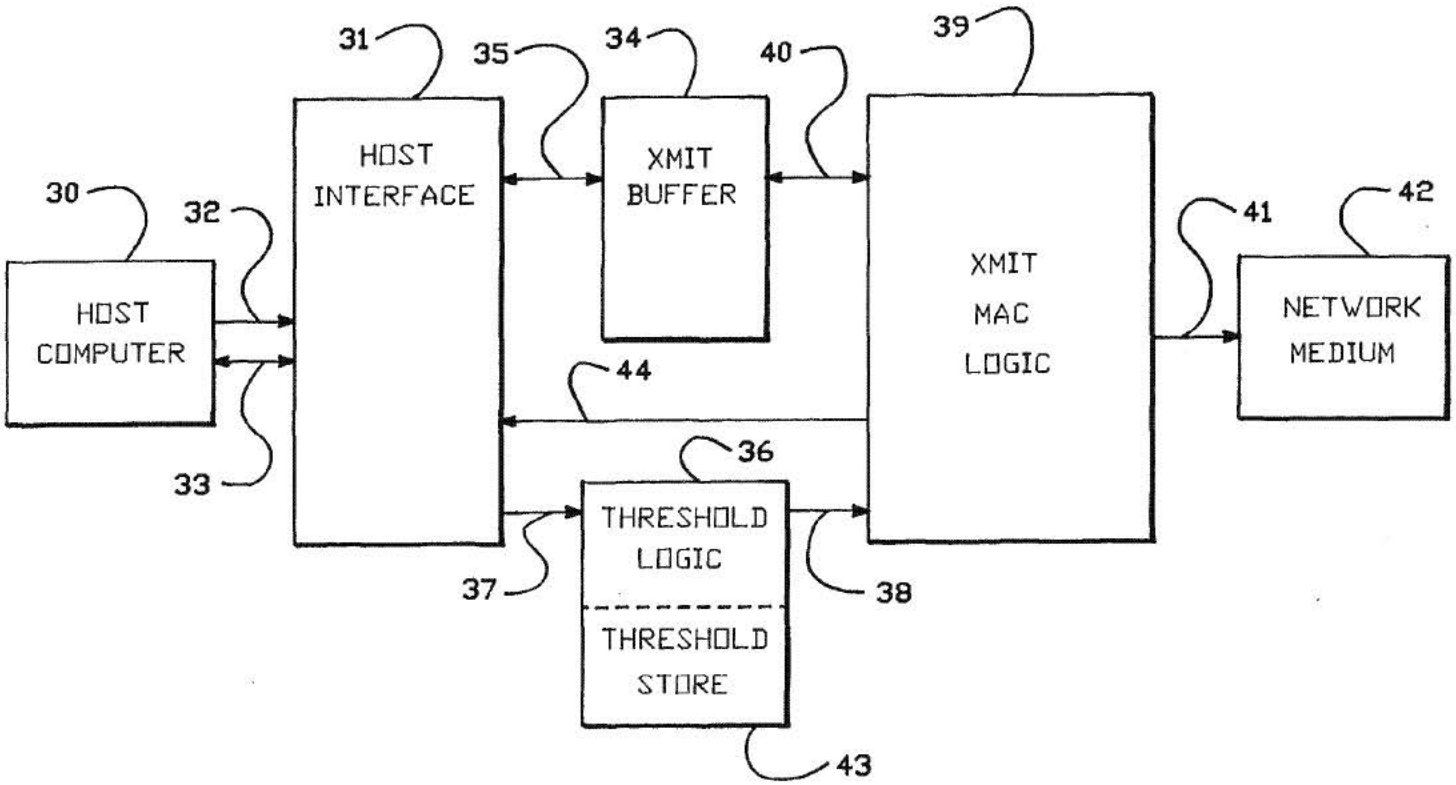

31
35
34
40
39
HOST
INTERFACE
XMIT
BUFFER
XMIT
MAC
LOGIC
30
32
41
42
HOST
COMPUTER
NETWORK
MEDIUM
44
33
36
THRESHOLD
LOGIC
37
38
THRESHOLD
STORE
43

FIG.—2

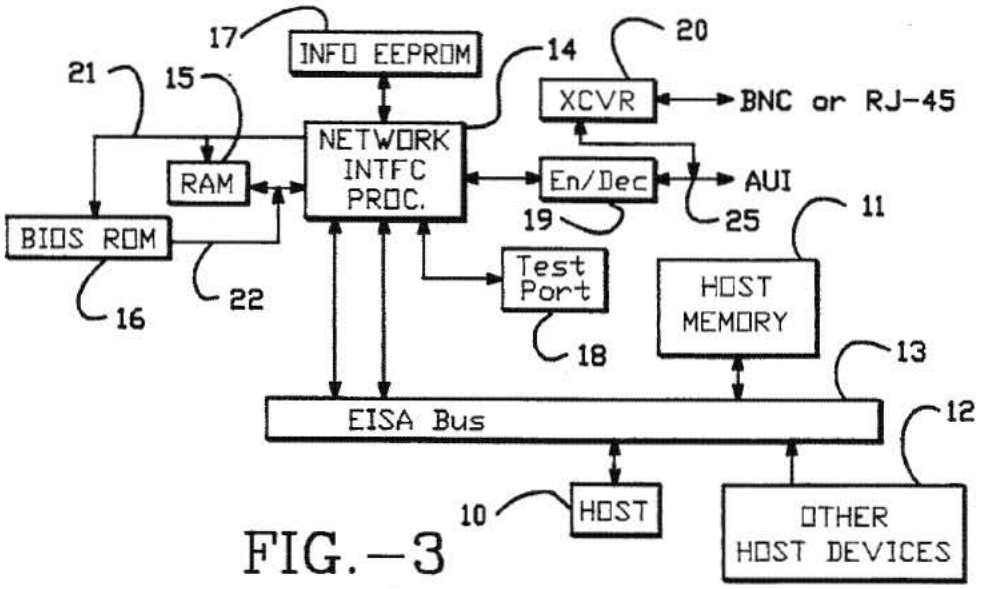

17
INFO EEPROM
14
20
21
15
XCVR
BNC or RJ-45
NETWORK INTFC PROC.
RAM
En/Dec
AUI
19
25
11
BIOS ROM
Test Port
HOST MEMORY
16
22
18
13
EISA Bus
12
10
HOST
OTHER HOST DEVICES

FIG.—3

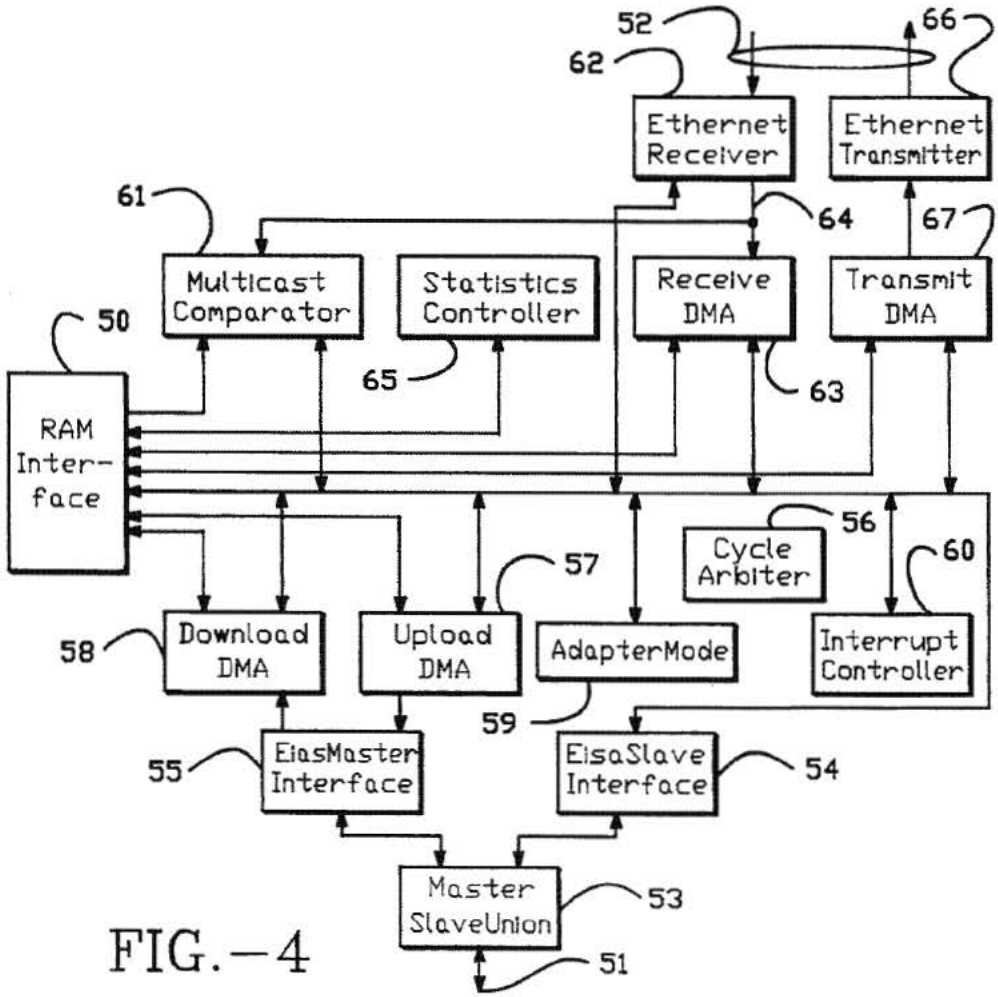

52
66
62
Ethernet Receiver
Ethernet Transmitter
61
64
67
Multicast Comparator
Statistics Controller
Receive DMA
Transmit DMA
50
65
63
RAM Inter-face
56
Cycle Arbiter
60
57
58
Download DMA
Upload DMA
AdapterMode
Interrupt Controller
59
55
EisaMaster Interface
EisaSlave Interface
54
Master SlaveUnion
53
51

FIG.—4

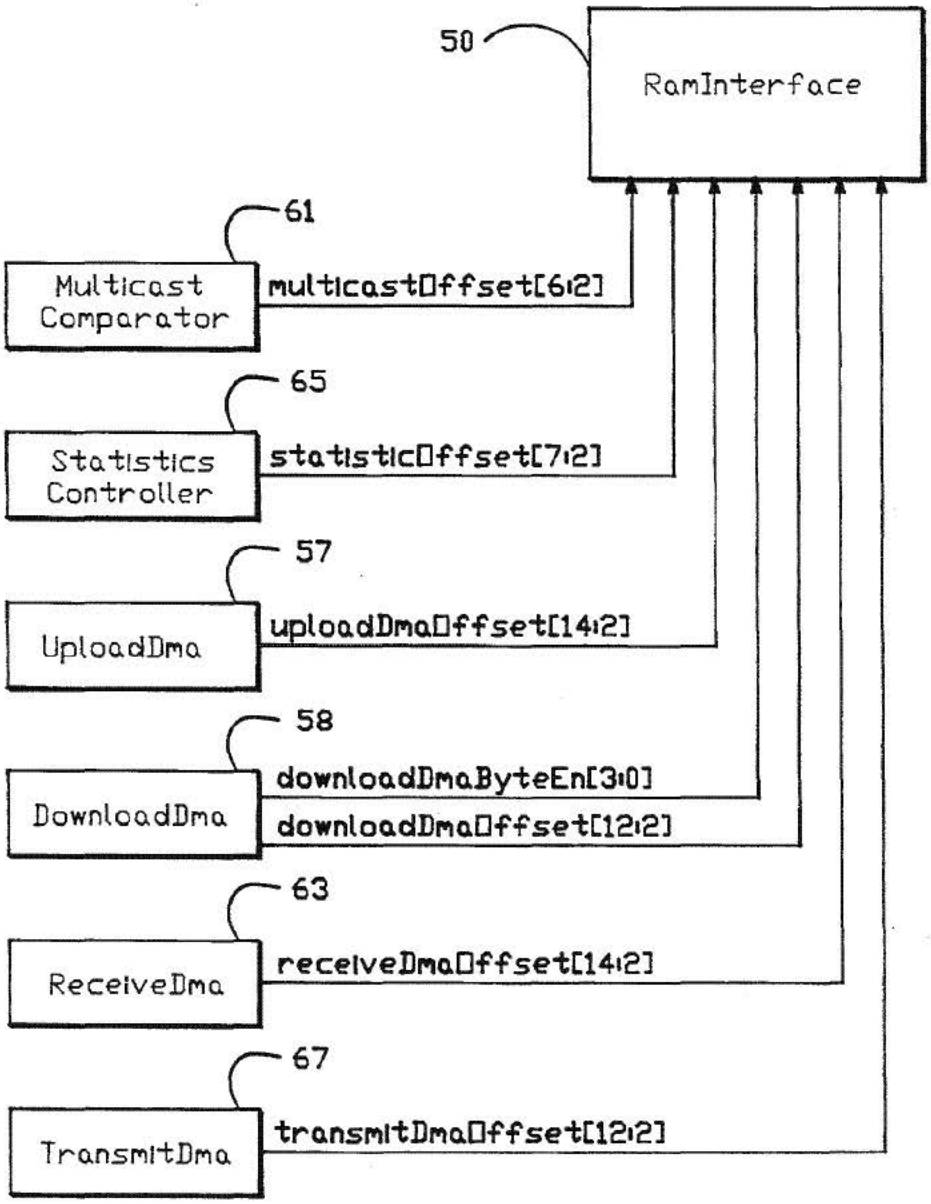

50
RamInterface
61
Multicast Comparator
multicastOffset[6:2]
65
Statistics Controller
statisticOffset[7:2]
57
UploadDma
uploadDmaOffset[14:2]
58
DownloadDma
downloadDmaByteEn[3:0]
downloadDmaOffset[12:2]
63
ReceiveDma
receiveDmaOffset[14:2]
67
TransmitDma
transmitDmaOffset[12:2]
Autonomous Access Address Bus Structure

FIG.—4A

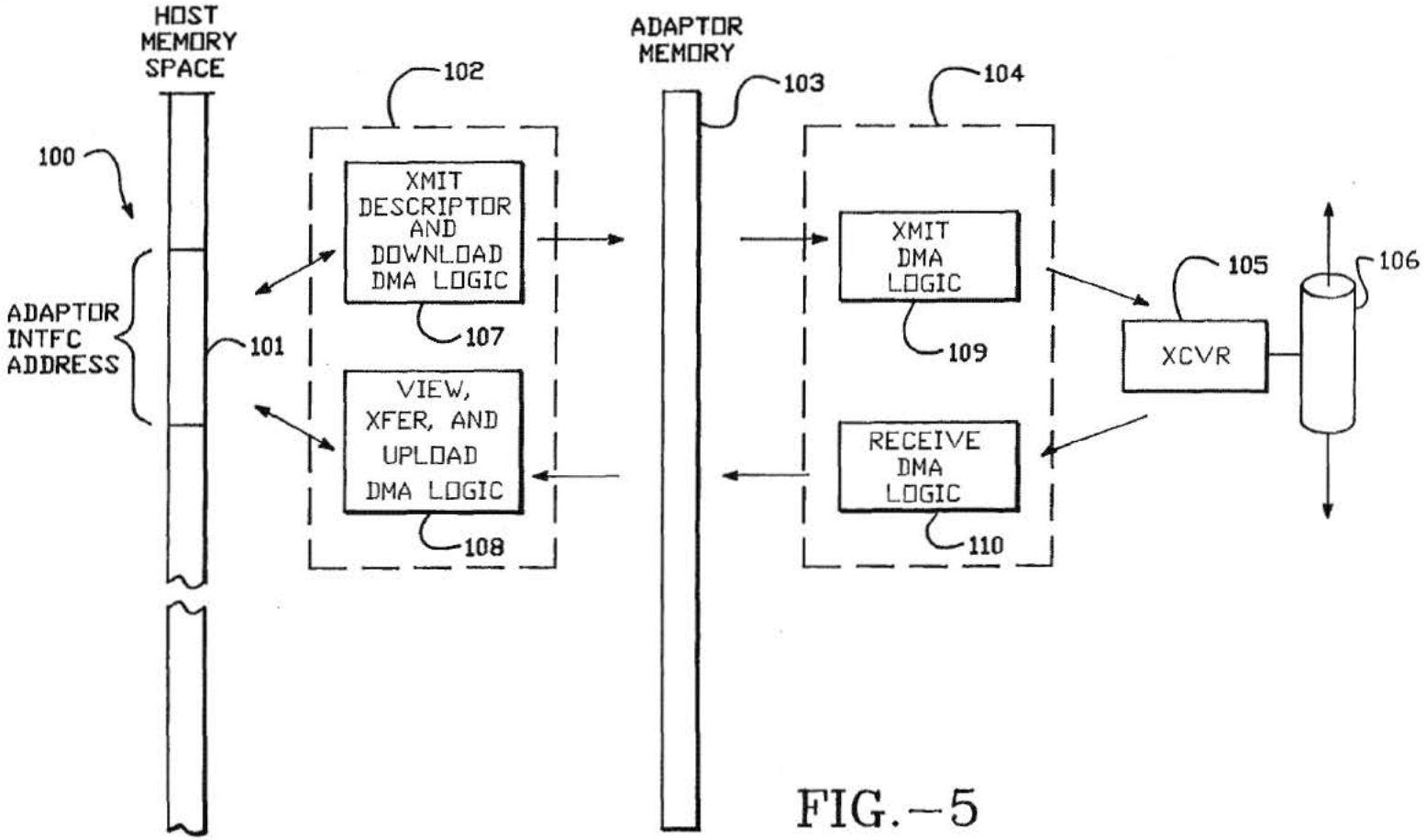

HOST MEMORY SPACE
100
ADAPTOR INTFC ADDRESS
101
102
XMIT DESCRIPTOR AND DOWNLOAD DMA LOGIC
107
VIEW, XFER, AND UPLOAD DMA LOGIC
108
ADAPTOR MEMORY
103
104
XMIT DMA LOGIC
109
RECEIVE DMA LOGIC
110
105
XCVR
106

FIG.—5

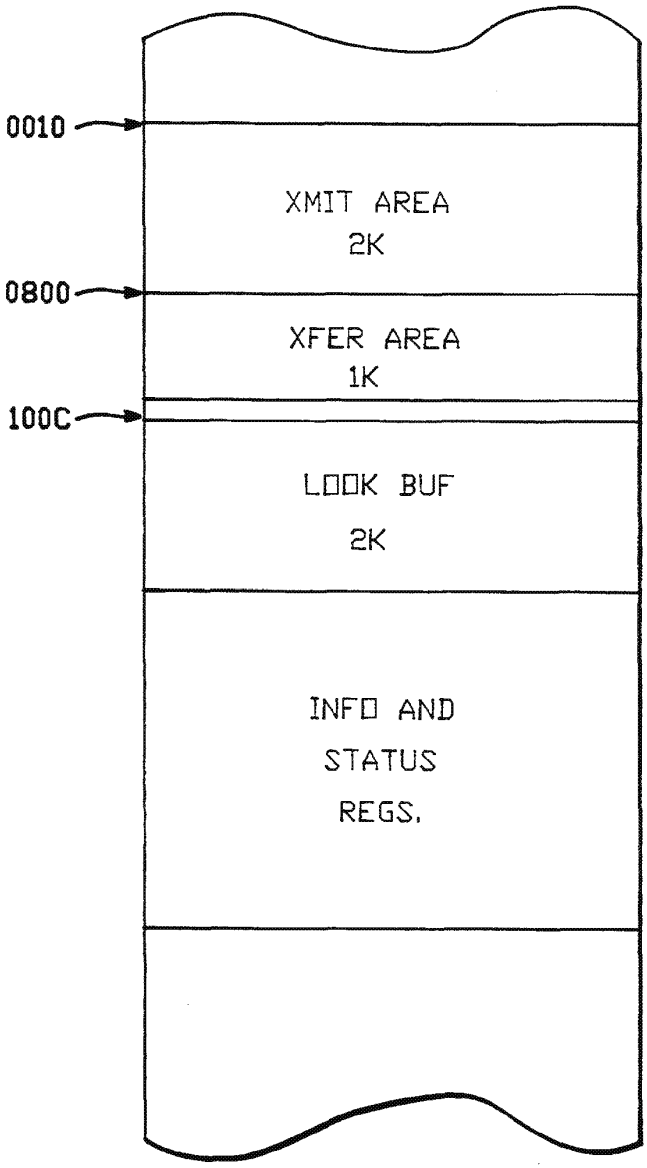
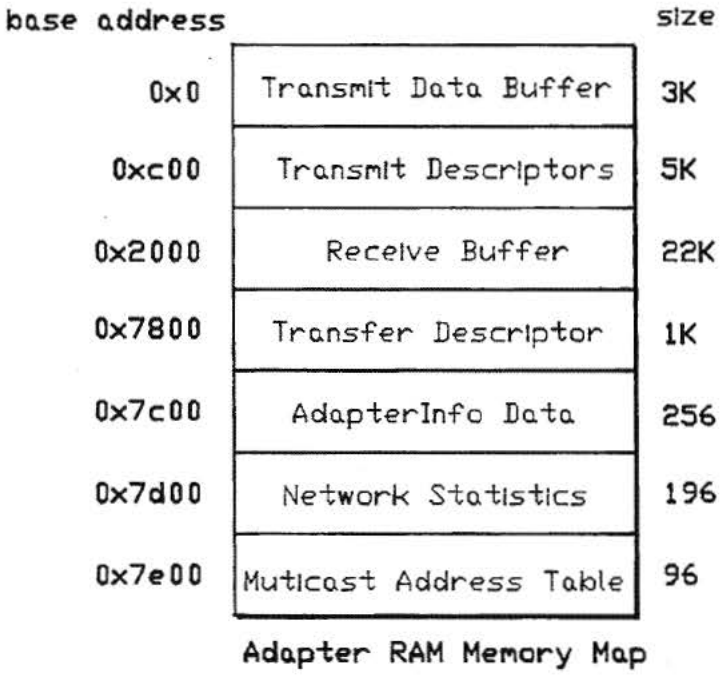

0010
XMIT AREA
2K
0800
XFER AREA
1K
100C
LOOK BUF
2K
INFO AND
STATUS
REGS.

FIG.—6



Adapter RAM Memory Map

FIG.—7

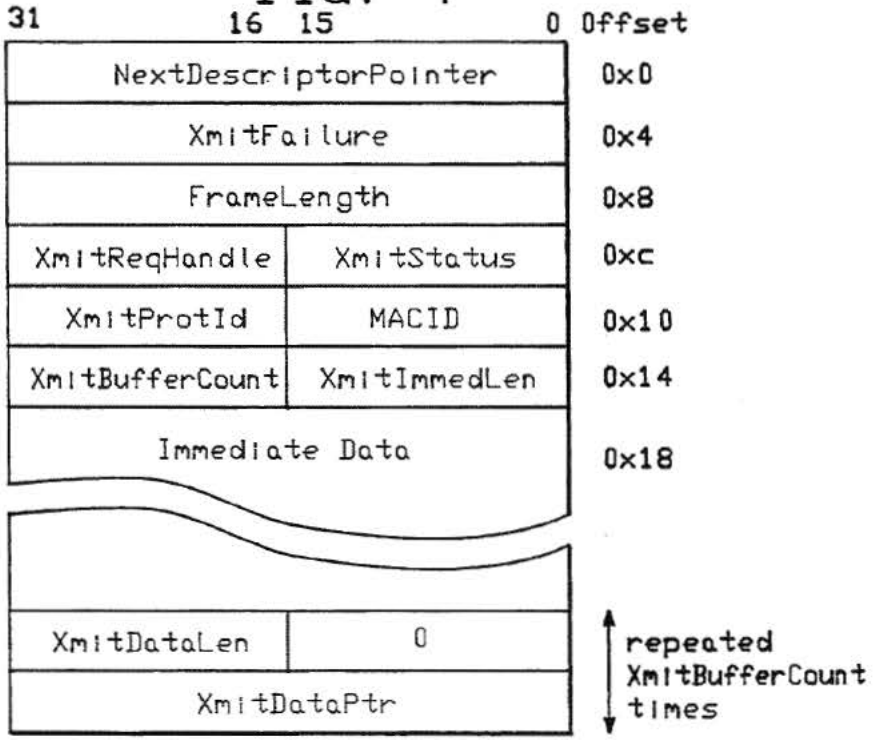


FIG.—8

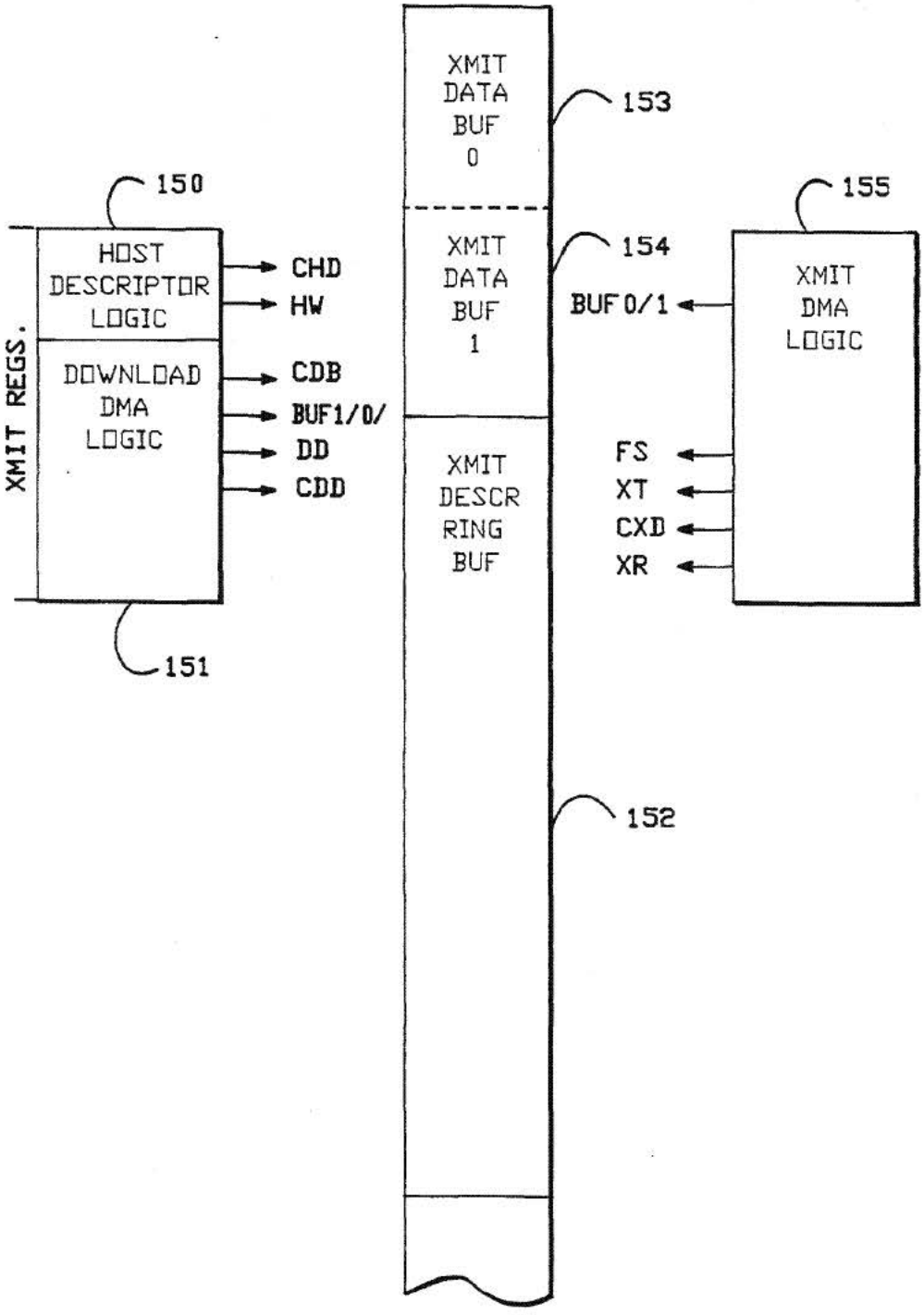

XMIT DATA BUF 0
153
150
155
HOST DESCRIPTOR LOGIC
CHD
HW
XMIT DATA BUF 1
154
BUF 0/1
XMIT DMA LOGIC
XMIT REGS.
DOWNLOAD DMA LOGIC
CDB
BUF1/0/
DD
CDD
XMIT DESCR RING BUF
FS
XT
CXD
XR
151
152

FIG.—9

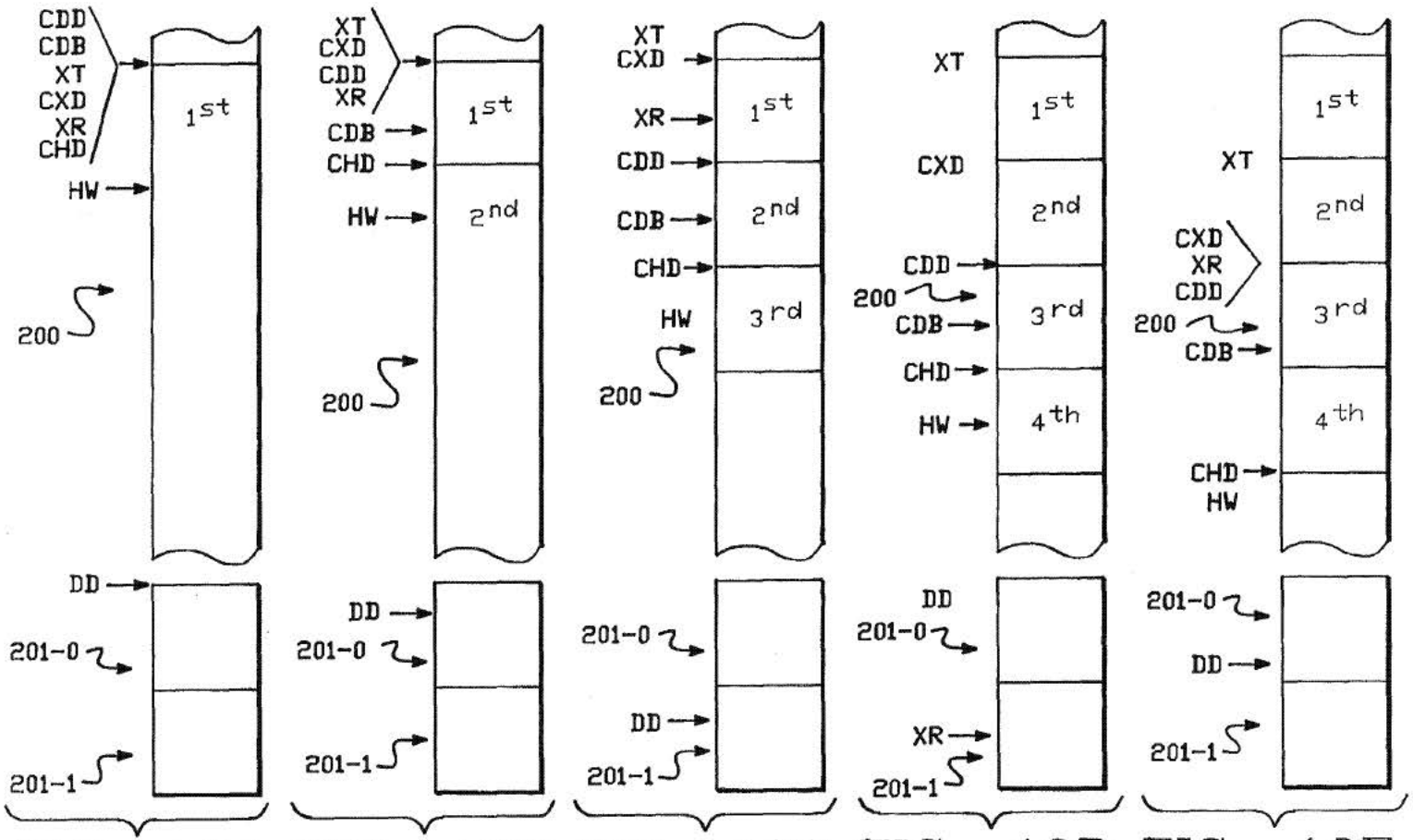


FIG.—10A FIG.—10B FIG.—10C FIG.—10D FIG.—10E

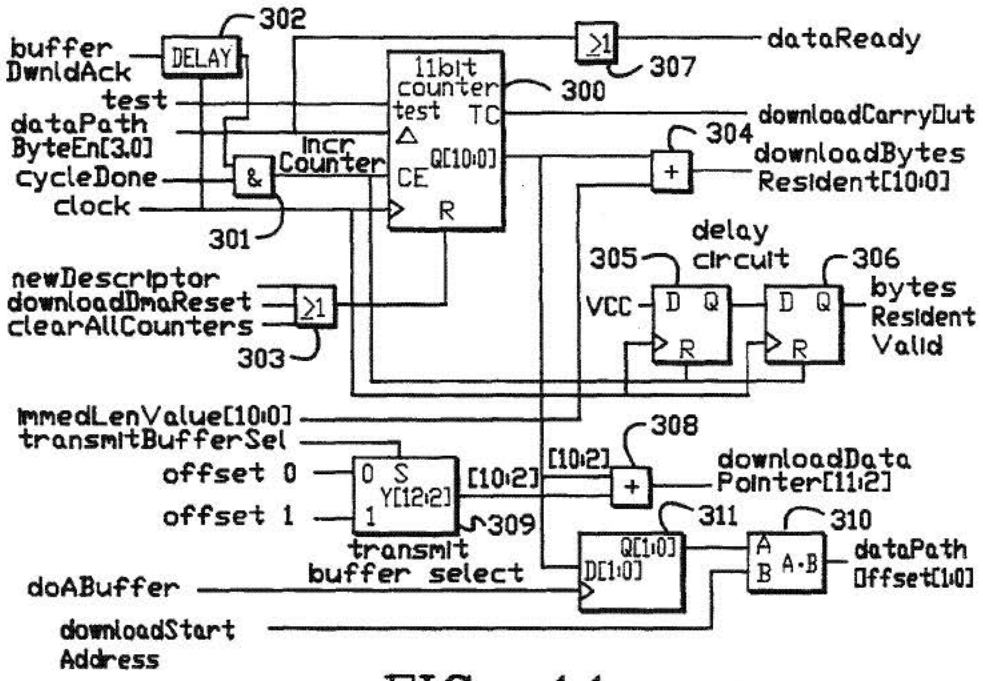

302
buffer DwnldAck
DELAY
test
dataPath ByteEn[3:0]
cycleDone
clock
301
&
Incr Counter
11bit counter
test TC
CE
Q[10:0]
R
300
≥1
307
dataReady
downloadCarryOut
304
+
downloadBytes Resident[10:0]
delay circuit
305
306
VCC
D Q
R
bytes Resident Valid
newDescriptor
downloadDmaReset
clearAllCounters
303
immedLenValue[10:0]
transmitBufferSel
offset 0
offset 1
0 S 1
Y[12:2]
[10:2]
308
downloadData Pointer[11:2]
309
transmit buffer select
311
310
Q[1:0]
D[1:0]
A B A-B
dataPath Offset[1:0]
doABuffer
downloadStart Address

FIG.—11

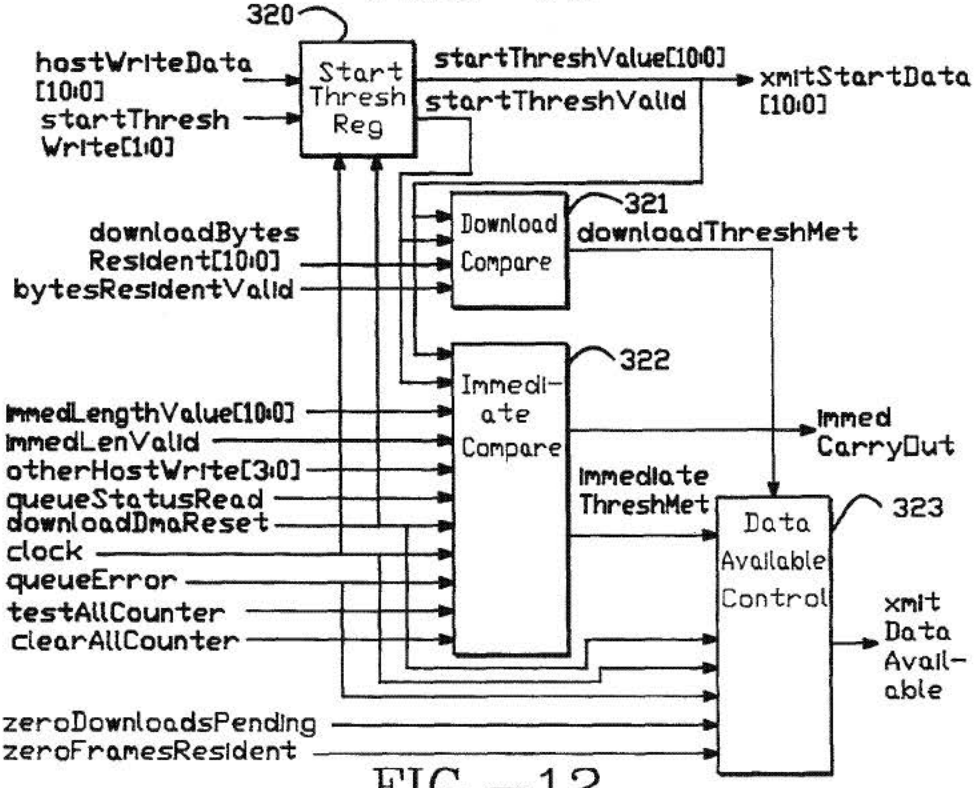

320
hostWriteData [10:0]
startThresh Write[1:0]
Start Thresh Reg
startThreshValue[10:0]
startThreshValid
xmitStartData [10:0]
downloadBytes Resident[10:0]
bytesResidentValid
Download Compare
321
downloadThreshMet
Immedi-ate Compare
322
immedLengthValue[10:0]
immedLenValid
otherHostWrite[3:0]
queueStatusRead
downloadDmaReset
clock
queueError
testAllCounter
clearAllCounter
immed CarryOut
immediate ThreshMet
Data Available Control
323
xmit Data Avail-able
zeroDownloadsPending
zeroFramesResident

FIG.—12

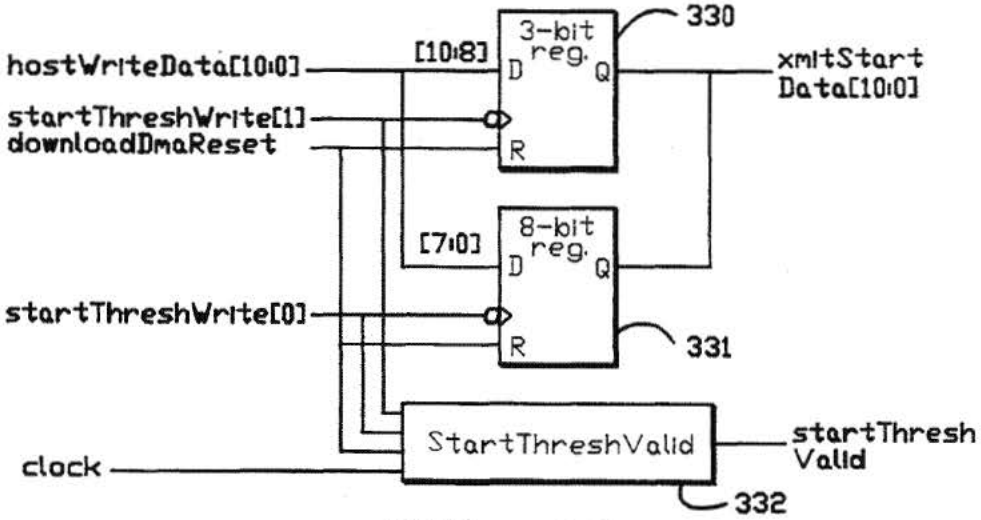
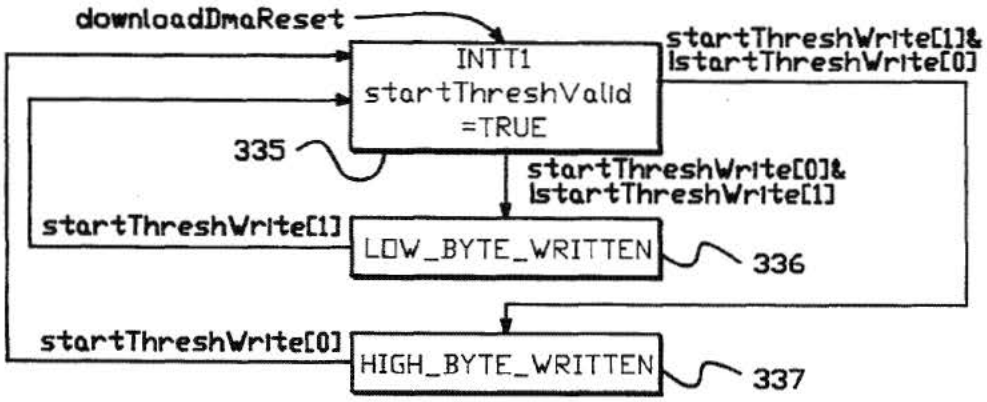


FIG.—13



Start Thresh Valid State Diagram

FIG.—14

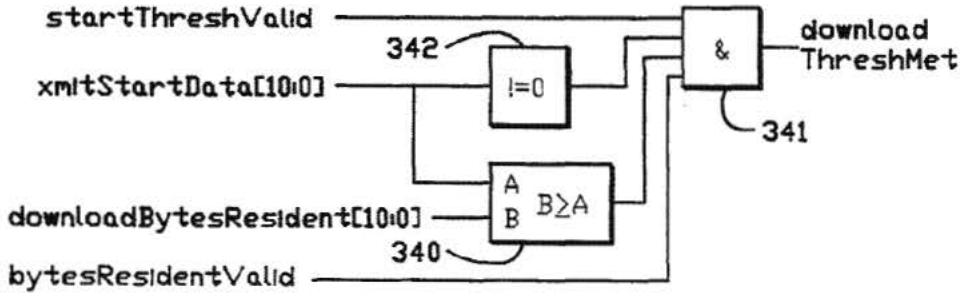


FIG.—15

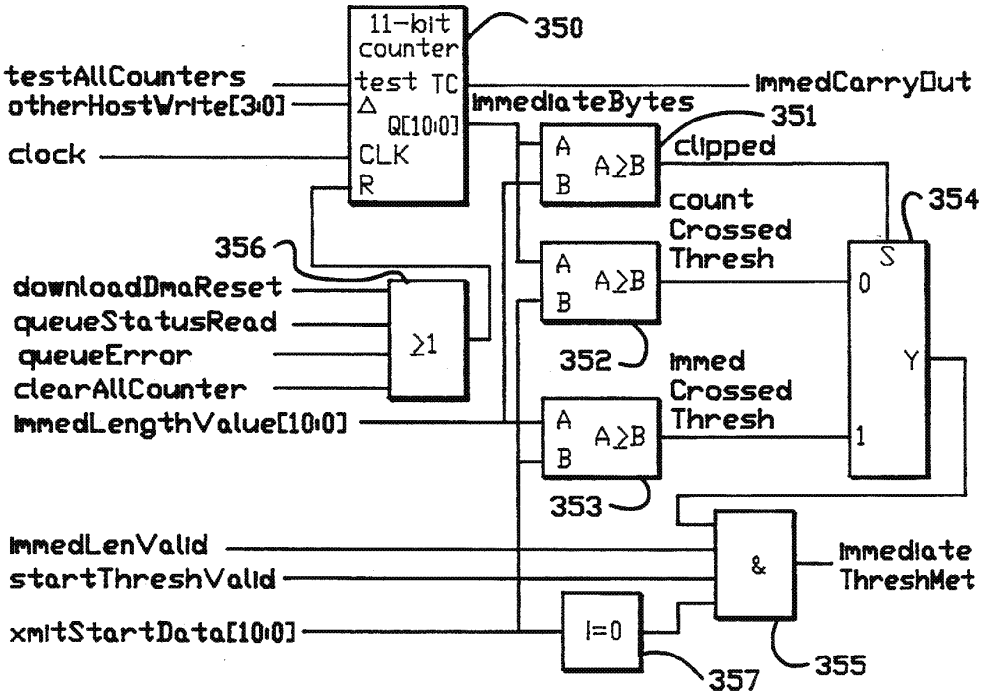

11-bit counter
test TC
Δ Q[10:0]
CLK
R
350
testAllCounters
otherHostWrite[3:0]
clock
immediateBytes
immedCarryOut
A
B
A≥B
351
clipped
count Crossed Thresh
354
S
0
Y
1
356
downloadDmaReset
queueStatusRead
queueError
clearAllCounter
≥1
352
immed Crossed Thresh
immedLengthValue[10:0]
353
immedLenValid
startThreshValid
&
immediate ThreshMet
xmitStartData[10:0]
!=0
357
355

FIG.—16

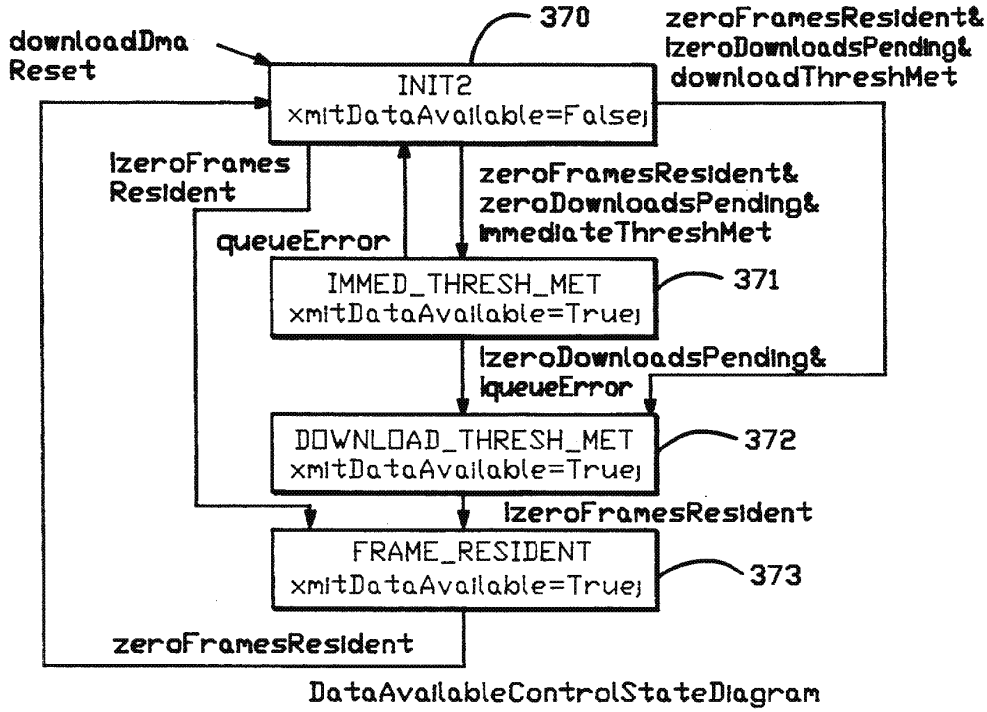

downloadDma Reset
370
zeroFramesResident&
!zeroDownloadsPending&
downloadThreshMet
INIT2
xmitDataAvailable=False
!zeroFrames Resident
zeroFramesResident&
zeroDownloadsPending&
immediateThreshMet
queueError
IMMED_THRESH_MET
xmitDataAvailable=True
371
!zeroDownloadsPending&
!queueError
DOWNLOAD_THRESH_MET
xmitDataAvailable=True
372
!zeroFramesResident
FRAME_RESIDENT
xmitDataAvailable=True
373
zeroFramesResident
DataAvailableControlStateDiagram

FIG.—17

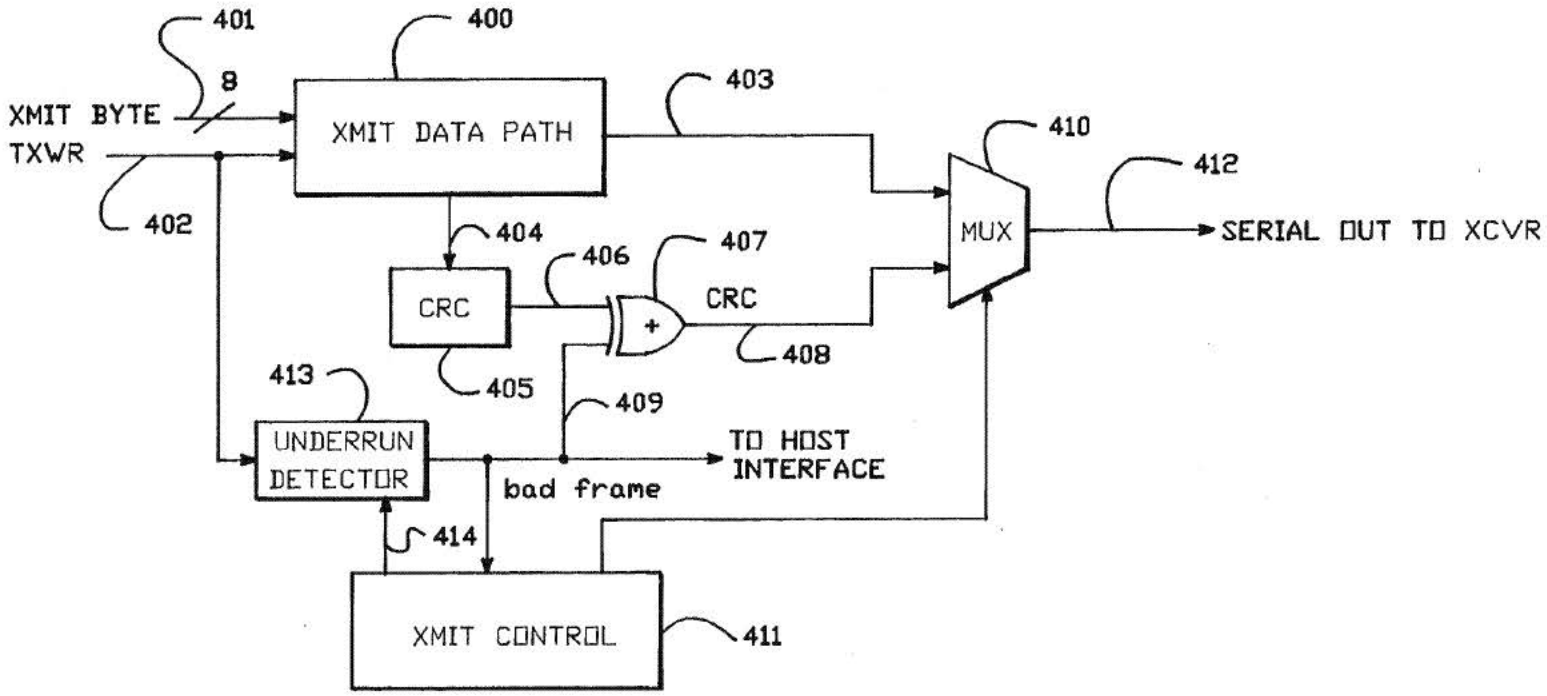

401
400
8
XMIT BYTE
TXWR
XMIT DATA PATH
403
410
412
MUX
SERIAL OUT TO XCVR
402
404
406
407
CRC
+
CRC
408
405
413
409
UNDERRUN
DETECTOR
TO HOST
INTERFACE
bad frame
414
XMIT CONTROL
411

FIG.—18

APPARATUS FOR AUTOMATIC INITIATION OF DATA TRANSMISSION

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application is related to copending U.S. patent application entitled NETWORK INTERFACE WITH HOST INDEPENDENT BUFFER MANAGEMENT, application Ser. No. 07/921,519, filed 28 Jul. 1992, now U.S. Pat. No. 5,299,313, which was owned at the time of invention and is currently owned by the same assignee.

FIELD OF THE INVENTION

The present invention relates to interfaces between communications networks and data processing systems; in particular, systems which involve the transmission of packets or frames of data across a communications network.

DESCRIPTION OF RELATED ART

Data communication systems are often based on the transmission of packets or frames of data that are composed by a sender. The packets or frames of data are designed to be compatible with the network protocol involved with the communication system. Thus, the sending system must compose the frames of data according to the network protocol prior to initiation of transmission of data. Often, a sending system will wait for acknowledgement that a frame of data sent to a network adapter has been transmitted prior to performing a subsequent task, such as composing a second frame of data to be transmitted.

Some network adapter interfaces include dedicated transmit buffers into which a frame of data composed by the sending system can be downloaded by the sending system. The frame is then stored in the transmit data buffer until the media access control functions associated with transmitting the frame on the network have successfully transmitted the frame, or cancelled the frame transmission. If the frame transmission is cancelled, the data may be retained in the transmit data buffer until the sending system initiates a second attempt to transmit the frame. Transmit data buffers are to be distinguished from first-in-first-out FIFO systems, in which the sending system downloads data of a frame into the FIFO, while the network adapter unloads the FIFO during a transmission. The data in FIFOs cannot be retained and reused by the media access control functions, or by the host, like data in transmit data buffers.

Although transmit data buffers enable a sending system to compose and download a frame into the transmit data buffer, and then attend to other tasks while the network adapter attempts to transmit the frame, it suffers the disadvantage that transmission of a frame is delayed until the entire frame has been downloaded into the buffer. Thus, transmit data buffer type systems improve host system efficiency at the expense of network throughput. Operations which are communication intensive suffer a performance downgrade.

Furthermore, the prior art systems which use transmit data buffers require the host or sending system to manage the transmit data buffer. A network interface controller transfers data from the host managed transmit data buffer using DMA techniques through a FIFO buffer in the interface controller and on to the network.

Representative prior art systems include the National Semiconductor DP83932B, a systems-oriented network interface controller (SONIC) and the Intel 82586 local area network coprocessor.

It is desirable to provide the advantages of a transmit data buffer, while maintaining the communications throughput available from the simpler FIFO based systems.

SUMMARY OF THE INVENTION

The present invention provides for the early initiation of transmission of data in a network interface that includes a dedicated transmit buffer. The system includes logic for transferring frames of data composed by the host computer into the transmit buffer. Also, the amount of data of a frame which is downloaded by the host to the transmit buffer is monitored to make a threshold determination of an amount of data of the frame resident in the transmit buffer. The network interface controller includes logic for initiating transmission of the frame when the threshold determination indicates that a sufficient portion of the frame is resident in the transmit buffer, and prior to the transfer of all of the data of the frame into the transmit buffer. In one aspect of the invention, the monitoring logic includes a threshold store, which is programmable by the host computer for storing a threshold value and logic for posting status information to the host. Thus, the threshold value may be set by the host system to optimize performance using the alterable threshold store and the posted status information.

According to another aspect of the present invention, the transmit buffer includes a transmit descriptor ring, and a transmit data buffer. The host system composes a frame by storing a transmit descriptor in the adapter managed transmit descriptor ring. The transmit descriptor may remain resident in the transmit descriptor ring for some time prior to an initiation of the transmission of the data by the adapter, because of other transmit descriptors being processed ahead of a current descriptor, or other reasons. When the adapter begins processing of a transmit descriptor, it retrieves immediate data from the descriptor itself, and begins a download process into the transmit data buffer of data identified in the descriptor. The threshold logic determines the amount of immediate data from the descriptor, and monitors the downloading of data of the frame into the download area. When the combination meets the threshold, then actual transmission of the frame is initiated. Thus, transmission of a frame may be initiated before the complete frame has been downloaded into the download area.

Accordingly, the present invention provides an architecture for a network interface controller which greatly increases the flexibility for host systems in composing and transmitting frames of data, while maintaining communications throughput for applications that are communications intensive.

Other aspects and advantages of the present invention can be seen upon review of the figures, the detailed description, and the claims which follow.

BRIEF DESCRIPTION OF THE FIGURES

FIG. 1 is a simplified block diagram of a system with a network interface according to the present invention.

FIG. 2 is a schematic functional block diagram of the early transmit system according to the present invention.

FIG. 3 is a block diagram of a preferred implementation of the host system and network interface according to the present invention.

FIGS. 4 and 4A are diagrams of the network interface processor of FIG. 3 implementing the present invention, and of the structure of the autonomous access address buses, respectively.

FIG. 5 a schematic diagram illustrating data flow from the host memory space through adapter memory to the network according to the present invention.

FIG. 6 is a map of the host system address space used for any transmission of data according to the present invention.

FIG. 7 is a memory map of the adapter memory independent of the host system address space.

FIG. 8 illustrates the transcript descriptor data structure according to one aspect of the present invention.

FIG. 9 illustrates the management of the transmit descriptor ring buffer and transmit data buffer, and pointers used during the transmit operation according to the present invention.

FIGS. 10A–10E are a schematic illustration of the management of the pointers for the transmit descriptor ring buffer and transmit data buffer.

FIG. 11 is a logic diagram illustrating the data path arithmetic used in the transmit function for the network interface controller of FIG. 4.

FIG. 12 is a logic diagram of the transmit start control logic for the network interface adapter of FIG. 4.

FIG. 13 is a logic diagram of the transmit start threshold register for the logic of FIG. 12.

FIG. 14 is a state diagram for the transmit start threshold register of FIG. 13.

FIG. 15 is a logic diagram of the download compare logic of FIG. 12.

FIG. 16 is a logic diagram of the immediate data comparator of FIG. 12.

FIG. 17 is a state diagram for the data available control function in the logic of 12.

FIG. 18 is a simplified block diagram of the transmit logic according to the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

A detailed description of preferred embodiments of the present invention is provided with respect to the figures. FIGS. 1 and 2 provide an overview of the data communications systems according to the present invention. FIGS. 3 through 10A–10E provide a detailed description of a preferred implementation of the present invention, including data structures and data flow. FIGS. 11–18 provide one implementation of logic for implementing the early transmission feature of the present invention.

I. System Overview

FIG. 1 illustrates a data communication system according to the present invention with a controller circuit using a dedicated transmit buffer memory which is automatically enabled to begin transmission of a frame on the network when the number of bytes available in the transmit buffer memory exceeds a preprogrammed threshold. As shown in FIG. 1, such system for communicating data includes a host data processing system, generally referred to by reference number 1, which includes a host system bus 2, a host central processing unit 3, host memory 4, and other host devices 5, all communicating across the bus 2. A network interface adapter 6 is coupled to the host bus 2. The adapter 6 is coupled to a transceiver 7 which is, in turn, connected to a network medium 8, such as coaxial cable, a twisted pair conductor, a fiber optic, satellite, wireless, or other communication medium. The network adapter 6 is, in turn, connected to an adapter memory 9, which is managed by the interface controller 6 or by the host CPU 3, depending on a particular implementation provided. The adapter 6 also includes early transmit logic 6A for monitoring the transfer of data from the host system 1 into the adapter memory 9. The early transmit logic 6A makes a threshold determination for a given frame being transferred from the host 1 into the adapter memory 9. When the threshold is met, then the adapter 6 begins transmitting the frame across the network. Of course, a wide variety of other configurations of these components could be implemented. For instance, the adapter memory 9 may be connected directly to the host bus 2. Also, multiple bus configurations might be utilized.

FIG. 2 illustrates the functional components of the early transmit system according to the present invention. In FIG. 2, a number of functional blocks are represented. The host computer or sending system 30 communicates with a host interface logic 31, receiving data as indicated by line 32 and control signals as indicated by line 33. The host interface logic 31 supplies data to a transmit buffer 34 across line 35. Also, threshold logic 36 is coupled with the host interface logic 31 to monitor the transfer of data from the host computer 30 into the frame buffer 34. The line 37 indicates the coupling of the threshold logic 36 to the host interface logic 31. The threshold logic makes the threshold determination and generates a signal as indicated by line 38 to transmit logic 39 including, for instance, media access control MAC logic. The transmit logic 39 receives data from the buffer 34 across lines 40 and supplies that data as indicated by line 41 to the network medium 42. Coupled with the threshold logic 36 is a threshold store 43 which stores a threshold value which indicates an amount of data of a frame that must be resident in the frame buffer 34 before transmission of that frame may be initiated by the transmit DMA logic and MAC 39.

The threshold store 43, in a preferred system, is dynamically programmable by the host computer 30. In this embodiment, the threshold store 43 is a register accessible by the host through the interface logic 31. Alternatively, the threshold store may be a read only memory set during manufacture. In yet other alternatives, the threshold store may be implemented using user specified data in non-volatile memory, such as EEPROMs, FLASH EPROMs, or other memory storage devices.

The transmit logic 39 also supplies status information across line 44 to the host interface logic 31, for posting to the host system. The status information includes indications of underrun conditions and may be used by the host to optimize the value in the threshold store 43.

In operation, the host computer composes a frame of data to be transmitted on the network medium 42. The host computer 30 then identifies that frame through the host interface 31. The host interface coupled with the identifiers of the frame move data from the host computer 30 into the buffer 34 according to the description of the frame. The threshold logic 36 monitors the transfer of data into the buffer 34. When the threshold

amount of data is resident in the buffer **34**, then the transmit logic **39** is instructed to begin transmission of the frame. The transmit logic **39** then begins retrieving data from the buffer **34** to support transmission of the frame on the medium **42**. This operation begins before the entire frame has been transferred from the host computer **30** into the buffer **34**, if the transmit logic **39** is available to transmit the frame subject of the ongoing download from the host computer **30**, the frame being downloaded into the buffer **34** is larger than the threshold set by the threshold store **43**, and the host computer **30** indicates that immediate transmission of the data is desired.

As mentioned above, the buffer **34** may be directly addressable by the host computer **30**, or in preferred systems is addressable through the host interface **31** which consists of a prespecified area of host address space into which the host computer **30** writes data and control signals associated with a transmission operation. The host interface **31** then maps the transfer of data from that prespecified address space into the buffer **34** and host accessible registers independent of the host.

A more detailed description of a preferred embodiment of the present invention is illustrated in FIG. **3**.

II. System Overview

FIG. **3** is a schematic diagram of a computer system including the network interface controller according to the present invention. The computer system includes a host system, including a host processor **10**, host memory **11**, and other host devices **12**, all communicating through a host system bus **13**, such as an EISA bus. The host system bus **13** includes address lines which define a host system address space. Typically, for an EISA bus, there are 32 address lines establishing a host system address space of about 4 Gigabytes.

The network interface controller includes a network interface processor **14**, implemented in one preferred system as an application specific integrated circuit designed to implement the functions outlined below using VERILOG design tools as known in the art (available from Cadence, Inc., San Jose, Calif.). The network interface processor **14** is coupled through appropriate buffers to the bus **13**. The network interface processor **14** is also coupled to random access memory **15**, BIOS ROM **16**, and INFO EEPROM **17**, a test port **18**, an encode/decode chip **19**, and a network transceiver **20**. The network transceiver **20** is, in turn, coupled to a network medium.

A majority of the functionality is embodied in the network interface processor **14**. In the preferred embodiment, all registers that are accessible across the bus **13** by the host system reside either in the processor **14**, or in the RAM **15**. If resident in the RAM **15**, their access is managed by the network interface processor **14**.

The RAM **15** is a primary resource on the network interface controller. This resource provides buffer memory outside the host address space used in the transmit and receive operations of the network interface. Details concerning the organization and utilization of this RAM **15** are described below.

The BIOS ROM **16** provides extension to the host system's basic input/output code through the network interface processor **14** during initialization. The addresses for the BIOS ROM **16** and the data from the BIOS ROM **16** are coupled to the network interface processor **14** across buses **21** and **22**, respectively, which are also shared by the RAM **15**.

The INFO EEPROM **17** stores critical adapter specific data used by drivers, diagnostics, and network management software. This data is stored during the manufacturing process. During initialization of the interface controller, the contents of the EEPROM **17** are loaded into a prespecified area of the RAM **15** for use during operation.

Coupled to the interface processor **16** is an encode/decode chip **19**, such as the National Semiconductor 8391 Manchester encode/decode chip. The signals coupled to the AUI connector are provided to allow use of a transceiver external to the board.

The transceiver **20** in a preferred system comprises either a thin Ethernet (coax/BNC) transceiver or a 10BaseT (Type 3/RJ-45) transceiver. Control signals for the transceiver **20** are produced on the network interface controller **14**, using the conversion logic on the encode/decode chip **19**.

A test port **18** is provided in a preferred system for use during manufacture and testing.

III. Controller Functional Units

FIG. **4** provides a block diagram of the network interface processor **14** of FIG. **3**, including functional blocks and data paths. There are numerous connections not shown having to do with the control of the various data flow paths. The interfaces illustrated include a RAM interface **50**, a host bus interface **51**, and a transceiver interface **52**. The bus interface **51** is implemented for an EISA bus, and operates at times either as a master or as a slave on the bus. Each of the functional units in the implementation shown in FIG. **4** is described below.

A. EISA Slave Interface **54**

The EISA slave interface **54** provides a path for the EISA host to access the registers and buffers managed by the network interface controller. The module contains configuration registers for the controller, and performs crude decoding of the EISA bus for the purpose of routing signals. The EISA slave interface **54** does not interpret any of the addressing of individual registers distributed throughout the controller.

In operation, the EISA slave interface continuously monitors the EISA address bus and determines when the configuration registers, memory mapped network registers, or BIOS ROM of the adapter are being accessed.

In addition, for every memory slave cycle initiated by the EISA bus, the EISA slave interface will post a cycle request to the cycle arbiter **56**. The cycle arbiter imposes wait states upon the host system until the request has been granted.

The EISA slave interface also provides a generic 32 bit bus interface to the remainder of the network controller. The generic nature of the interface allows for easy adaptation of the design to other bus types, such as the microchannel, without requiring redesign of the remainder of the chip.

Bits **14–2** of the EISA address bus are latched and pass through to other modules. The least significant two bits (**1**, **0**) of the address are represented by 4 byte enables that are also valid throughout a data transfer cycle. Four 8-bit byte lanes make up the slave data channel. The data bus is actually a pair of unidirectional buses, one for writes and one for reads in a preferred system. The data write bus is wired in a multi-drop fashion to all modules that require connection to the EISA data bus through the slave interface. The read bus is multiplexed and masked in the RAM interface module **50**. Write requests by the EISA bus can be held until they are

acknowledged by the cycle arbiter **56**. When a cycle is held, the EISA bus may be released from wait states while the cycle completes on the adapter. If a second cycle is generated by the EISA bus while the first one is still pending, then the EISA bus will be held off with wait states until the pending write is performed. In this specific embodiment, pipelining of EISA reads is not supported.

The EISA slave interface also provides an interface to the EEPROM **17**. This interface operates to transfer the contents of the EEPROM **17** into the adapter memory after reset.

There are numerous registers in the EISA slave interface module **54**, primarily related to configuration of the adapter that conform to the EISA bus specification. These registers do such things as set up the adapter's memory base address, the interrupt level, the transceiver type selection, and the BIOS ROM enable. The configuration registers also provide the host with a positive means of identifying the adapter type and to globally disable the adapter.

B. EISA Master Interface **55**

The EISA master interface **55** handles requests from the upload DMA **57** and download DMA **58** for performing bus master operations across the EISA bus. The EISA master interface **55** autonomously arbitrates between pending upload and download requests, because of the EISA bus disallowing mixed reads and writes while performing burst transfers, used by the DMA operations of the preferred embodiment.

The bus master transfers are always initiated by either the upload DMA **57** or the download DMA **58**. The transfers may be terminated by either the DMA modules upon completion of a transfer, or by the EISA master interface upon preemption by another arbitrary device on the EISA bus.

Thus, the function of the EISA master interface **55** is to arbitrate for access to the EISA bus when transfer requests are pending from either or both of the upload DMA **57** and the download DMA **58**. The EISA master interface **55** performs the signalling necessary to establish first transfers with address slaves on the bus. It also ensures compliance with the EISA bus definitions.

This module also converts real mode segment:offset addresses to 20 bit linear addresses when enabled by the adapter mode logic **59**.

C. Master/Slave Union Module **53**

The master/slave union module **53** provides for sharing of connections to the EISA bus by the EISA master interface **55** and the EISA slave interface **54**. This union module **53** consists primarily of a series of 2:1 multiplexers.

D. Interrupt Controller Module **60**

The controller also includes an interrupt controller module **60**. The interrupt controller module **60** implements various interrupt and indication functions, including masking and enable/disable mechanisms. Interrupt signals are generated by various modules within the controller, and are routed to the interrupt controller module **60**. The interrupt controller module **60** then passes the interrupt signals through various enables and masks before OR-ing them together and driving the result onto the host bus.

The interrupt controller module **60** does not detect interrupt worthy events or acknowledge the interrupts passed to the host. It includes a number of adapter resident registers utilized in the interrupt handling functions.

E. Adapter Mode Module **59**

The adapter mode module **59** provides a number of functions including setting various basic operating modes of the controller, and reporting status of various conditions of the controller. The adapter module **59** also establishes the base address of a window register used for diagnostics by the host system. Furthermore, the adapter mode module **59** generates reset functions for the adapter. Also, this module provides the MAC ID register which identifies the media access controller implemented by the device, for communication to various modules within the controller and to the host system.

F. Cycle Arbiter Module **56**

The cycle arbiter module **56** is responsible for distributing access to the adapter's RAM resident and ASIC resident registers through the RAM interface **50** among various requestors. It functions to allow timely access to the RAM by modules that are most in danger of suffering an overrun or underrun condition in response to a priority scheme.

G. Multicast Comparator Module **61**

The controller illustrated in FIG. **2** also includes a multicast comparator module **61**. When enabled by the adapter mode module **59**, the multicast comparator module **61** performs a bit by bit comparison of a received frame's destination address field with contents of the multicast address table. The multicast address table is established by the host and stored in RAM **15**. A mismatch during this compare, coupled with neither an individual address nor a broadcast address match, will result in the rejection of an incoming frame.

Thus, the multicast comparator module **61** monitors the activity of the Ethernet receiver module **62** and the receive DMA module **63** to determine when a new frame is being received. Each byte that is received by the Ethernet receiver **62** and presented at the parallel interface **64** of the receiver, is shadowed by the multicast comparator module **61**. These bytes are then compared against valid entries in a multicast address table accessible by the multicast comparator **61**.

The multicast comparator **61** does not establish or maintain the contents of the multicast address table. However, the module detects the host accesses to the table and supplies appropriate redirection offsets to the RAM interface module **50**.

H. Statistics Controller Module **65**

The preferred system also includes a statistics controller module **65**. This module monitors activity of various other modules within the controller, most particularly the Ethernet transmitter module **66** and the Ethernet receiver module **62**, and updates statistics maintained in RAM **15** as applicable events occur.

I. Download DMA Module **58**

The download DMA module **58** is responsible for issuing requests for bus master downloads of data from the host system to the adapter memory. This data is then deposited within the adapter's onboard transmit data buffer for either immediate or future transmission.

As soon as buffer descriptors are defined and one of the transmit data buffers becomes available, as described below, the download DMA module submits requests for download bus master operations to the EISA master interface **55**. The download DMA module **58** performs byte alignment, including any required packing and unpacking to align the data as implied by the respective starting addresses of the host and the adapter.

The download DMA module **58** also includes logic for maintaining the transmit descriptor ring buffer within the adapter's RAM. The download DMA module **58** generates an interrupt in the appropriate mode to indicate completion of the download operation. Also, the download DMA module **58** informs the transmit DMA module **67** when it is time to begin transmission. Various registers involved in the download DMA operation are described in more detail below as they are involved directly in the data buffering process of the present invention.

J. Transmit DMA Module **67**

The transmit DMA module **67** operates to fetch bytes from the transmit descriptor buffer, the transmit data buffer, or both, as described below, when directed to do so by the download DMA logic **58**. The fetched bytes are then presented in sequence to the Ethernet transmitter module **66**.

Thus, the transmit DMA module **67** functions to read the contents of the transmit descriptor ring buffer to determine the extent of immediate data, and the length of the overall frame to be transmitted. If a frame specified does not match the specifications for the network, for instance, is shorter than the 802.3 minimum, and the adapter is so enabled, this module will supply additional bytes of undefined data to the Ethernet transmitter module **66** as padding.

Collision retries in the CSMA/CD network are handled by the transmit DMA module **67**. When a collision is indicated by the Ethernet transmitter **66**, the transmit DMA module **67** will resubmit the same frame by reinterpreting the frame descriptor in the buffer.

If enabled, and when the transmit complete condition is satisfied, a transmit complete interrupt is generated for handling by the interrupt controller **60** in the transmit DMA module **67**.

The transmit DMA module **67** also stores appropriate frame status in the transmit descriptor ring buffer of the adapter upon completion of transmission.

The transmit DMA module **67** also detects underrun conditions, when a shortfall of data available for transmission occurs.

Again, registers involved in operation of the transmit DMA module **67** are described in detail below.

K. Ethernet Transmitter Module **66**

The Ethernet transmitter module **66** provides media access control function associated with transmission for an 802.3 standard network. This module accepts parallel data bytes from the transmit DMA module **67** and applies the 802.3 access rules, and supplies serial data to the external encoder/decoder chip.

L. Ethernet Receiver Module **62**

Similarly, the Ethernet receiver module **62** is an essential 802.3 implementation. This module accepts serial data from an external encoder/decoder, applies the 802.3 rules to the data and presents the data in parallel form for use by the receive DMA module **63**. Thus, the Ethernet transmitter **66** and Ethernet receivers perform the standard CSMA/CD functions.

M. Receive DMA Module **63**

The receive DMA module **63** is a functional complement to the transmit DMA module **67**. This module is responsible for accepting parallel data bytes on the Ethernet receiver **62** and depositing them in the adapter's receive ring buffer.

The receive DMA module **63** is responsible for assembling bytes from the Ethernet receiver into 32 bit words prior to storage in the adapter's RAM. At completion of frame reception, the frame's status and length are deposited within the receive ring buffer for use by the host system.

The receive DMA module **63** is also responsible for establishing and maintaining of the receive buffer ring within the RAM of the adapter as described in detail below. Furthermore, the positioning of the "LOOKBUF" register allowing the host to view received data, as mentioned below, is handled by the receive DMA module **63**.

The receive DMA module **63** also handles interrupt indications under appropriate conditions.

N. Upload DMA Module **57**

The upload DMA module **57** performs data transfers from the receive buffer through the RAM interface **50** to the host system. Thus, the receive ring buffer is managed by the receive DMA module **63** and interpreted by the upload DMA **57**. Actual bus master transfers are carried out by the EISA master interface module **55**.

The upload DMA module **67** interprets data structures deposited in the receive ring buffer by the receive DMA module **63** including the location and length of a received frame. It also reads the transfer descriptor which is defined by the host system to determine how many bytes of the frame to transfer, and where in the host memory to transfer the frame.

The upload DMA module **57** requests bus master cycles from the EISA master interface module **55** in response to valid transfer requests deposited by the host.

The upload DMA module **57** also utilizes an interlock between the receive DMA module **63** and the upload DMA module **57**, to reduce the transfer rate to the host system to prevent "getting ahead" of the frame being received through the receive DMA module **63**. Finally, this module generates interrupts indicating completion of a transfer for use by the host.

O. RAM Interface Module **50**

The RAM interface module **50** provides multiplexers and masks for various functions involved in addressing the RAM. The module multiplexes the various address and data sources together to form parameters for RAM access cycles. This module **50** is responsible for gathering up data from the various other modules on the controller that can be read by the host system. Additionally, this module applies a mask to the data to force unused upper bits to a zero, and latches data words for multicycle reads.

P. JTAG Module

Also included in the controller, though not shown, is a JTAG module which implements a state machine as specified in IEEE standard 1149.1-1990, May 21, 1990. This module provides a scan test of the ASICs pins for use during manufacture.

Q. The Autonomous Access Address Bus Definition

The multi-cast comparator **61** statistics controller **65**, download DMA **58**, transmit DMA **67**, receive DMA **63**, EISA slave interface **54**, and upload DMA **57** all require autonomous access capabilities through the RAM interface **50** to the adapter's RAM. The autonomous accesses are those that occur independently of any cycles that may be occurring on the host bus.

This requirement requires the capability to generate and communicate address information for use by the RAM interface **50**. In the case for the receive DMA **63**, transmit DMA **67**, and download DMA **58**, the capability is implied by the DMA logic which drives offset buses. Similar offset buses dedicated to the autonomous

accesses are provided for use by the statistics controller **65**, upload DMA **57**, and multi-cast comparator **61**.

FIG. **4**A schematically defines the structure of the autonomous access address buses. The various functional modules are given like reference numbers, as provided in FIG. **4**.

The multi-cast comparator **61** drives the multi-cast offset bus [**6**:**2**] to supply a variable offset to the RAM interface **50** for addressing individual locations within the multi-cast address table. A valid value is presented during the associated autonomous access cycle.

The statistics controller **65** drives the statistics offset bus [**7**:**2**] to supply a variable offset to the RAM interface **50** for addressing individual locations within statistics region of the adapter's RAM. A valid value is presented during the associated autonomous access cycle.

The upload DMA **57** drives the upload DMA offset bus [**14**:**2**] to supply a variable offset to the RAM interface **50** for addressing individual locations within a receive buffer ring. A valid value is presented during the associated autonomous access cycle.

The download DMA **58** drives a download DMA offset bus [**12**:**2**] to supply a variable offset to the transmit buffer/transmit descriptor region of the RAM during autonomous cycles. Two select signals generated by the download DMA logic **58** are used to distinguish between the two RAM regions. Also, the download DMA logic **58** generates a download DMA byte enable [**3**:**0**] to indicate which byte lanes of the 32 bit data bus extending from download DMA **58** to the RAM interface **50** contain valid data. The valid value is presented during the associated autonomous cycle.

The receive DMA **63** drives a receive DMA offset bus [**14**:**2**] to convey a variable offset within the 22K byte receive buffer region of the RAM.

Finally, the transmit DMA logic drives a transmit DMA offset bus [**12**:**2**] to supply a variable offset to the transmit buffer/transmit descriptor region of the RAM during autonomous cycles. The two select signals generated by the transmit DMA **67** are used to distinguish between the two RAM regions.

IV. Transmit and Receive Data Flow and Structure

FIG. **5** provides a heuristic data flow diagram of an adapter according to the present invention emphasizing the host interface, the adapter memory and the network interface data flow for transmission and reception.

As mentioned above, the host system will include a host memory space (generally **100**) defined by the addresses on the host bus. A pre-specified block **101** of the host memory space is set aside for the adapter interface addresses. The adapter includes host interface logic **102** which is responsive to accesses across the host bus within the adapter interface address block **101**. Also in the adapter is a host independent memory **103**. The host interface logic operates to map the transfer data between the specified block **101** of addresses and the independent memory. The adapter also includes network interface logic **104** which is coupled to the adapter memory. The network interface logic manages transfers of data from buffers in the independent memory **103** and the network transceiver **105**. The network transceiver **105** then supplies the data onto the network medium **106**.

The host interface logic includes a transmit descriptor logic and a download DMA logic (generally **107**) used in the transmit process, and view logic, transfer descriptor logic, and upload DMA logic (generally **108**) used in the receive process. These modules basically manage communication of data between the independent memory **103** and the host in response to reads/writes by the host system to the adapter interface address block **101**. This relieves the host of any address translations or buffer management functions for the transmit and receive operations.

The network interface logic **104** includes transmit DMA logic, (generally **109**) and receive DMA logic (generally **110**). The transmit DMA logic **109** is responsive to descriptors stored in the adapter memory **103**, as described below, for moving data out of the independent adapter memory **103** to the network transceiver **105**. Similarly, the receive DMA logic **110** is responsible for moving data from the transceiver **105** into the independent adapter memory **103**. Thus, all communications of data from the network medium **106** are coupled directly into host independent memory **103**. Communications from the host independent memory **103** are then controlled through the host interface logic **102** in response to a memory mapped region in the host memory space, greatly simplifying the protocol software necessary to communicate with the network.

FIG. **6** provides a simplified map of the host address block **101** used by the host interface. The addresses within this block appear to the host like memory mapped registers in a continuous 8K block of the host address space in a preferred system.

For the EISA embodiment, "registers", or mapped areas, in the block **101** are located on double word address boundaries, thus, addresses are a multiple of four. Many of the "registers" occupy several double words (as many as **509**) of memory space.

Although the "registers" are memory mapped to an arbitrary pre-specified block of host address space, none of the reads or writes performed by the host system to these registers actually directly access the adapter memory. Rather, the accesses to the memory mapped space are interpreted by the host interface logic **104** transparent to the host system. Thus, the memory in the adapter is independent of the host address space and of host management. FIG. **6** provides an overview mapping of the host address space used for accessing these registers. The registers include primarily a transmit area register (XMIT AREA) at offset 0010 (hex), a transfer area register (XFER AREA) at offset 0800 (hex), and a look buffer (LOOKBUF) at offset 100C (hex). Various status, statistics, information, and indication registers are distributed throughout the balance of the area.

The XMIT AREA register is used by the host to write transmit descriptors into the adapter. The transmit descriptors are described in more detail below, but include data that identifies data to be compiled and transmitted as a frame, and may include immediate data. The XMIT AREA at offset 0010 (hex) is approximately 2K bytes in size. This data is mapped into a transmit descriptor ring in the independent adapter memory as described below.

The XFER AREA at offset 0800 (hex) in the adapter interface host address block is a buffer of approximately 1K byte through which transfer descriptors are written into the independent memory of the adapter. The LOOKBUF area at offset 100C (hex) is a buffer of approximately 2K bytes providing a read only window into a receive ring buffer within the host independent adapter memory. More details of the receiver process are provided in the above cross-referenced application entitled NETWORK INTERFACE WITH HOST INDEPENDENT BUFFER MANAGEMENT.

FIG. 7 provides a map of the host independent memory on the adapter. This memory is organized into a transmit data buffer at offset 0 of approximately 3K bytes, a transmit descriptor ring at offset 0C00 (hex) of approximately 5K bytes, a receive buffer ring at offset 2000 (hex) of approximately 22K bytes, and a transfer descriptor area at offset 7800 (hex) of approximately 1K bytes is provided in the independent memory. The last three areas of memory include adapter information, network statistics, and multicast address tables for use by the adapter.

In the preferred system, the adapter uses 32K bytes of static RAM for the transmit buffers, receive buffers, control structures, and various status and statistics registers. Several of the regions in the adapter's memory defined in FIG. 5 provide defined data structures.

A. Transmit Data Buffer

The transmit data buffer occupies 3K bytes as mentioned above. This region is divided into two 1.5K buffers. Only the data that are downloaded to the adapter via bus master transfers are stored in these buffers. The controller will use both the contents of the transmit data buffer and the immediate data portion of the transmit descriptors, when encapsulating a frame for transmission. The adapter automatically alternates the use of the buffers after choosing the buffer closest to the base of the memory as the power up default.

The transmit buffers are shared by the download DMA logic and the transmit DMA logic. The transmit DMA logic may switch from buffer 0 to buffer 1 and back again freely. The only restriction being the availability of transmit data as defined by the transmit start threshold register. The transmit DMA module switches from one buffer to the other whenever it has completed a transmission. The buffer switch occurs regardless of whether or not the transmission was successful and regardless of whether or not bus master download data were used in the preceding transmission.

The download DMA module may only switch from one buffer to the other, if the buffer it is going to switch to is not being used by the transmit DMA module. Download DMA will attempt to switch from one buffer to another every time it completes processing of a transmit descriptor as described below, regardless of whether or not any bus master operations were called for in the preceding descriptor. However, it will not change to a buffer that is in use by the transmit DMA module.

B. Transmit Descriptors

Transmit descriptors define frames that are pending transmission, and hold the status of frames that have been transmitted. These descriptors are of variable length and are arranged in a sequential fashion around a 5K byte ring buffer as mentioned above. The first entry of the descriptor must align on a double word boundary. FIG. 8 illustrates the transmit descriptor data structure.

The bulk of the contents of an entry into the transmit descriptors region is copied verbatim from the data supplied by the host processor via the XMIT AREA illustrated in FIG. 6. However, in order to comply with the format requirements of the XMIT PROT ID and XMIT FRAME STATUS registers and to supply sufficient information for frame transmission and buffer management; one value must be relocated and several must be deposited automatically.

The host processor's writes to the XMIT AREA are offset automatically by the adapter such that the first value written—XMIT PROT ID and XMIT REQ HANDLE—end up in the fifth 32 bit word location (offset 10 (hex)) in the next available data structure in the ring. This means that the XMIT REQ HANDLE value is written by the host to the location reserved for the MACID value. Immediately after the XMIT REQ HANDLE value is written to the adapter's RAM, the adapter must copy the contents of the least significant 16 bits of the fifth 32 bit word location to the most significant 16 bits of the fourth 32 bit word location. After copying XMIT REQ HANDLE, the adapter will retrieve the MACID value from the MACID register and write that to the location vacated by XMIT REQ HANDLE. Later, after frame transmission, the least significant 16 bits of the fourth 32 bit location (offset C (hex)) will be updated with the transmit frame's status.

The NEXT DESCRIPTOR POINTER entry may be updated by the adapter any time after the XMIT BUFFER COUNT and XMIT IMMED LEN values have been written to the adapter. Because the data written to the XMIT AREA register must be written in a precise order to exact locations, the writes of these two values can be easily detected and used for the descriptor size calculations required to determine the start of the next descriptor without having to retrieve the values from RAM.

Finally, once the last XMIT DATA LEN value has been written to the adapter, the frame length can be calculated and deposited in the FRAME LENGTH position of the data structure. This value is also copied to the XMIT FRAME LENGTH register in the controller chip for immediate use by the host.

The next several paragraphs define each of the fields of the transmit descriptor data structure.

The NEXT DESCRIPTOR POINTER value points to the first word of the next descriptor. This value is updated immediately after the host reads XMIT QUEUE STATUS. NEXT DESCRIPTOR POINTER being defined does not necessarily imply that the location pointed to contains a valid descriptor. It merely indicates where the next valid descriptor may be found once it is defined.

The FRAME LENGTH field is computed and updated by the adapter. The frame length is calculated by summing all of the XMIT DATA LEN values and the XMIT IMMED LEN value. The resulting sum is the total number of bytes in the transmit frame. If the sum is less than the 802.3 minimum frame length, then the sum will be set to the minimum frame length value. The sum is written to the FRAME LENGTH line of the transmit descriptor and is also made available to the host via the XMIT FRAME LENGTH register.

The XMIT FAILURE field contains the error code that is made up of the status bits gathered from the Ethernet transmitter after the completion of transmission. This field is mapped to the XMIT FAILURE register for host access.

The XMIT REQ HANDLE value is interpreted by the transmit DMA controller to determine whether or not to generate an indication upon completion of the transmission attempt(s) for the associated frame. If the field is non-zero, an indication will be generated. Also, the frame's entry in the transmit descriptor ring will be maintained until the host has had an opportunity to examine the transmit status. The XMIT REQ HANDLE, XMIT STATUS, XMIT PROT ID, and the MACID fields are all made available to the host when an indication is generated. If XMIT REQ HANDLE is

a zero, then the transmit descriptor queue entry is discarded after transmission without notifying the host in any way. Transmit underrun conditions are posted regardless of a zero XMIT REQ HANDLE.

The XMIT STATUS field contains the transmit status for the associated frame. The contents of this field are updated immediately after the transmission attempt(s). The return codes are defined in the XMIT STATUS register definition.

The XMIT PROT ID value in this field is merely maintained within the queue for use by the host upon completion of the transmission to identify the particular protocol responsible for the frame. It allows simultaneous activity of multiple protocols. Together XMIT PROT ID and XMIT REQ HANDLE uniquely identify a frame passing through the adapter.

The MACID, like XMIT PROT ID, is maintained within the queue for use upon completion of the transmission. However, the host does not write this value to the adapter via the XMIT AREA register. Rather, the host stores this value once in the MACID register and then depends upon the adapter to deposit this value into the descriptor data structure after XMIT REQ HANDLE has been copied to its final position.

The contents of the XMIT BUFFER COUNT field are supplied by the host via a write to XMIT AREA. This field specifies the number of buffers within the host's memory that are used to make up the transmit frame. Each of the buffers is transferred in the order listed from the host's memory to the adapter's transmit data buffer as soon as one of the two transmit data buffers becomes available. If XMIT BUFFER COUNT is zero, then no bus master operations will be performed for this frame.

The XMIT IMMED LEN field, defined by a write from the host to XMIT AREA, specifies the number of "immediate" data bytes that will be supplied by the host. If this field is zero, then the next 32 bit word location will contain the first host data buffer descriptor and the entire frame will be transferred to the adapter via bus master cycles. The XMIT IMMED LEN value will not necessarily be a multiple of four. The location of the first host data buffer descriptor is determined as follows:

$$\text{DESCRIPTOR OFFSET} = ((\textit{XMIT IMMED LEN} + 3)\ \&\ \textit{fffc}(\text{hex})) + 18(\text{hex}).$$

The variable length IMMEDIATE DATA field contains the immediate data deposited to the adapter by the host using memory writes to XMIT AREA. This field may vary in length from 0 to 1.514 bytes. Immediate data is inserted into a transmit frame between the preamble and the transmit buffer data (if any) by the adapter during transmission. Generally, immediate data is made up of the destination and source addresses and any protocol-specific header data. It is reasonable, however, for the entire transmit frame to be considered immediate data. This would make it unnecessary for the adapter to perform any bus master operations to fetch the remainder of the transmit frame. If XMIT IMMED LEN is zero, then this field is skipped and the entire frame is assumed to reside in host memory resident data buffers. If XMIT IMMED LEN does not specify an integral number of double words, then the host may round up to the nearest multiple of 4 and write up to that number of bytes. The extra bytes, beyond XMIT IMMED LEN, will be ignored and not included as part of the transmitted frame.

The XMIT DATA LEN field, one of two entries per host data buffer descriptor, defines the number of bytes in the associated host buffer. This value need not be a multiple of four.

The 32 bit XMIT DATA PTR value is the physical starting address of the associated host data buffer. This value need not be a multiple of four.

V. Transmission Process

FIG. **9** illustrates the network interface logic and host interface logic used in managing the transmit data buffer and transmit descriptor ring buffer in the independent memory on the adapter. On the host interface side, the logic includes host descriptor logic **150** and download DMA logic **151**. The host descriptor logic **150** and download DMA logic **151** are coupled to the host address space through the transmit "registers" including the XMIT AREA register, the XMIT COMPLETE THRESH register, the XMIT FAILURE register, the XMIT FRAME LENGTH register, the XMIT FRAME STATUS register, the XMIT PROTID register, the XMIT QUEUE STATUS register, and the XMIT START THRESH register. Details of these registers are described below.

The descriptors illustrated in FIG. **6** are stored in the transmit descriptor area of the host independent RAM on the adapter by host writes to the XMIT AREA address block. Three different processes act upon entries in the descriptor queue. The host writes to initially create transmit descriptors, bus master downloads to move buffer data from host memory to the transmit data buffer and transmission of described frames on the network. The first two processes occur within the download DMA logic **151** and the host descriptor logic **150**. Transmission is performed by the transmit DMA logic **155**. During the course of adapter operations, the number and status of descriptors in the transmit descriptor ring buffer **152** will vary according to the relative speeds of the host write, download, and transmission processes.

Two variables within the download DMA logic **151** helped describe the status of the transmit descriptor queue. ZERO DOWNLOADS PENDING indicates that there are no complete frame descriptors yet to be processed by the download DMA logic **151** for download. The ZERO FRAMES RESIDENT variable indicates that there are no descriptors which have been already processed by the download process, but are yet to be transmitted.

Each of the three processes that make up the transmission mechanism maintains its own set of pointers to the descriptor queue.

The host descriptor logic **150** generates pointers for the transmit descriptor ring buffer **152** on the adapter memory, which identify the current position of host accesses for writing descriptors in the transmit descriptor ring buffer **152**. These pointers are designated the current host descriptor pointer CHD, and the host write pointer HW. The current host descriptor pointer CHD points to the base address of a descriptor currently being written, or expected to be written by the host. The host write pointer HW points to the location within the current descriptor (pointed to by the CHD pointer) to which the host is expected to write next. That is, the HW pointer predicts the offset within the XMIT AREA address block at which the host will be writing to complete the current descriptor.

The download DMA logic **151** generates 3 pointers, and a buffer select signal, while managing downloads

from the host system into the transmit data buffers XMIT DATA BUF0 **153** and XMIT DATA BUF1 **154**. The pointers generated by the download DMA logic **151** include the current download descriptor pointer CDD which points to the base address of a descriptor that is currently being processed by the bus master download DMA logic **151**. The second pointer generated by the download DMA logic includes the download data pointer DD which points to the location within the current transmit data buffer (either XMIT, DATA BUF0 or XMIT DATA BUF1) to which the download process is writing data. The third pointer generated by the download DMA logic **151** includes the current download buffer CDB pointer. The CDB pointer points to the buffer descriptor within the transmit descriptor pointed to by the CDD pointer, in which the specification of the buffer in host memory subject of a current download process resides.

The download DMA logic also selects the current transmit data buffer **153** and **154** to which the download DMA logic transfers data in the bus master operation as heuristically illustrated by signal BUF1/0.

The transmit DMA logic **155** generates three pointers for the transmission process. These pointers include the current transmit descriptor CXD pointer, which points to the base address of the descriptor in the transmit descriptor ring buffer **152** currently being processed by the transmit logic **155**. The transmit read XR pointer indicates the location within the current descriptor or current transmit data buffer (**153** or **154**) from which the transmission process is reading data to be transmitted.

The XMIT TAIL pointer (XT) points to the back end of the queue. The XT pointer points to an older descriptor in the transmit descriptor ring buffer **152** than the CXD pointer when there are frames that have completed transmission, but have not yet had their status information read by the host system.

The preferred system operates in two modes relative to generating indications to the host of the completion of a transmission. The preceding paragraphs described the data structures associated with host write, bus master download, and transmission processes when a XMIT COMPLETE ON DOWNLOAD variable is false and the indication is generated when the transmission is complete, or when XMIT COMPLETE THRESH (described below) is met. When XMIT COMPLETE ON DOWNLOAD is true, the download DMA logic **151** is also responsible for keeping track of the frames for which status has not been read. In this mode, the host receives the transmit complete indication for a frame upon download of the frame, so it is possible that the frame can be transmitted before the host is able to respond to the indication. This condition allows the XMIT TAIL pointer to no longer define the oldest useful entry in the descriptor ring buffer. Therefore, a frame status process is necessary. The frame status pointer FS points to the base address of the oldest descriptor for which status has not yet been read by the host, in this mode of operation. Since the FS pointer is related to the transmit process, it is illustrated heuristically in the transmit DMA logic **155** in FIG. **9**. In the preferred system, however, it is logically coupled with the download DMA logic **151**.

The transmit DMA also selects the current transmit data buffer **103** or **104** from which a current transmit frame is being composed, as indicated heuristically by the signal BUF0/1.

The interface to the adapter is seen by the host as if it were a set of registers at a prespecified address block. The significant "registers" involved in transmission are outlined below.

A. XMIT AREA

The purpose of this register is to provide a mechanism for the host to supply the adapter with immediate data and a data structure that defines where in the host's memory system the remainder of the frame (if any) resides. The adapter stores this information in the transmit descriptor ring **152** for use when the frame being described can eventually be transmitted.

The adapter uses the address to which the data is written and the order in which it is written to determine what the data represents. The data must be written to the adapter using the structure and order described above with respect to FIG. **6**.

Bus master downloads begin after a descriptor has been written to XMIT AREA and XMIT QUEUE STATUS (described below) has been read by the host.

Once the host has completed the transfer of the transmit descriptor buffer structures to XMIT AREA, the host may read XMIT FRAME LENGTH to determine the number of bytes that the host has specified to be included in the transmit frame. XMIT QUEUE STATUS should then be read to advance the CHD pointer so that another frame may be written to this register. After reading XMIT QUEUE STATUS, the contents of XMIT FRAME LENGTH are undefined, until XMIT AREA is filled again.

The actual frame transmission onto the network will commence when two conditions are met: (1) the XMIT START THRESH (described below) condition has been met, or, if XMIT START THRESH is zero, when the entire frame has been copied to the adapter's RAM, and (2) when there are no previously queued transmit requests. If more than XMIT START THRESH bytes of immediate data are written to XMIT AREA, then network transmission may begin before XMIT QUEUE STATUS is read.

If the adapter runs out of XMIT AREA resources while the host is writing data to XMIT AREA, the host will be returned a value of 6 when it reads XMIT QUEUE STATUS. The writes that exceed the capacity of the descriptor buffer will not corrupt data already queued up on the adapter.

The transmit frame's destination and source addresses must be explicitly supplied to the adapter by the host for each frame transmitted. This information can be provided as part of the immediate data or, if there is no immediate data, as the first twelve bytes of the first data buffer specified in the descriptor.

Essentially, the host provides every byte of the frame between the start of frame delimiter and the frame check sequence (CRC) through the XMIT AREA register or download DMA operation.

B. XMIT COMPLETE THRESH

XMIT COMPLETE THRESH provides for an early indication of transmission completion.

The XMIT COMPLETE THRESH register is used to specify the number of transmit bytes that remain to be either transmitted or downloaded to the adapter (depending upon the adapter mode) before the adapter will issue a XMIT COMPLETE indication. Only bits **10** through **0** are implemented in this register. Values greater than the maximum frame length will prevent this function from operating properly. The method for disabling this function is to set the register to zero. The

value in XMIT FRAME LENGTH (see below) is used to determine where the end of the transmit frame is.

If this threshold value is set too high, then the host will respond to the indication before the adapter can provide a valid transmit status indication. If XMIT FRAME STATUS returns a ff (hex), then XMIT COMPLETE THRESH should be adjusted to delay the indication slightly. This is accomplished by reducing the value in the XMIT COMPLETE THRESH register. The function of this register is disabled during the transmission of the first 60 bytes of the frame. This register is cleared to 0 during a reset.

C. XMIT FAILURE

XMIT FAILURE returns the cause of a transmit failure.

This register returns the cause of the failure of the attempt(s) to transmit a queued frame. A non-zero value indicates that the frame encountered one or more errors during the transmission attempt.

The bits in this register are defined as follows:

bit 0	DMA UNDERRUN
bit 1	LOSS OF CARRIER SENSE
bit 2	MAX COLLISIONS
bit 3	SQE TEST FAILED

This register will contain valid data regardless of the success or failure of the attempt to transmit a frame. If there was no failure, then this register will contain a value of 0 (hex). The contents of this register are valid after the frame has completed transmission (low byte of XMIT FRAME STATUS not equal to ff (hex)) and before XMIT PROT ID is read.

If a data underrun occurs, the adapter will force a CRC error into the frame during transmission to assure that the frame is received as a bad frame and is discarded by the destination device.

D. XMIT FRAME LENGTH

XMIT FRAME LENGTH returns the number of bytes to be transmitted.

The XMIT FRAME LENGTH register returns the total number of bytes queued up for transmission by the current transmit frame descriptor identified by CXD pointer. This value is the total of the number of immediate data bytes and of all of the buffer length fields downloaded to the adapter for this frame. The value returned by this register does not reflect the effects of any padding of the frame that may be done by the adapter when the frame is less than 60 bytes in length.

The XMIT FRAME LENGTH register becomes valid immediately after the host writes the last byte to XMIT AREA and remains valid until the first write to XMIT AREA after a read of XMIT QUEUE STATUS.

E. XMIT FRAME STATUS

XMIT FRAME STATUS returns the results of a transmit attempt.

The least significant 16 bits of this register return the status of the attempt(s) to transmit a queued frame. The most significant 16 bits returns the XMIT REQ HANDLE for the frame. A value of XXXX0000 (hex) (XXXX is the XMIT REQ HANDLE for this particular frame) is returned for a successful transmission while XXXX000a (hex) is returned for a failed transmission. XXXX00fe (hex) is returned if the adapter is in the process of retrying a transmission after a collision. If the transmission is still in progress, XMIT FRAME STATUS will return a XXXX00ff (hex).

If the frame was not transmitted successfully the specific cause of the transmit failure is available in XMIT FAILURE. Reading XMIT PROT ID advances XMIT FRAME STATUS to the status of the next transmitted frame, if any. If the "retry" status value is returned when XMIT FRAME STATUS is read, then reading XMIT FRAME STATUS will also clear the XMIT COMPLETE indication.

F. XMIT PROT ID

XMIT PROT ID returns the protocol ID of the transmit frame.

As soon as the adapter has completed its attempt(s) to transmit a queued frame and has posted its status, XMIT PROT ID can be read by the host as a method of identifying the frame. The value returned here is the same value that was written into the XMIT PROT ID field during the queuing of the frame via XMIT AREA.

Reading this register clears the XMIT COMPLETE indication except when the "retry" status value is read from XMIT FRAME STATUS. If "retry" was read, then reading XMIT FRAME STATUS will have cleared XMIT COMPLETE.

The XMIT PROT ID value resides in the upper 16 bits of the 32 bit register. The least significant 16 bits of this register will return the MACID value written to the MACID register. A double-word read will return both values simultaneously.

As multiple frames can be queued up for transmission, so can multiple transmission results be queued. Reading both words of XMIT PROT ID advances the completion status in XMIT FRAME STATUS, XMIT FAILURE, and XMIT PROT ID to the status of the next frame which has completed transmission, if any.

G. XMIT QUEUE STATUS

XMIT QUEUE STATUS returns the results of queuing a transmit frame.

A read of the XMIT QUEUE STATUS register returns the status of the host's attempt to queue up a transmit frame via XMIT AREA.

2 (hex)—Success: If the transmit request was successfully queued, this value is returned when XMIT QUEUE STATUS is read.

6 (hex)—Out of Resources: If the adapter runs out of queue storage RAM, then a status of 6 (hex) is returned.

7 (hex)—Frame Too Long: if the total number of bytes to be transmitted in a single frame exceeds the maximum frame length, this register will return a 7 (hex).

a (hex)—Order Violation: If the data written to XMIT AREA is written out of order, then this error code is returned.

ff (hex)—Adapter Not Ready: If XMIT QUEUE STATUS is read too quickly after the completion of the writes to XMIT AREA, it is possible to read the status value before the queuing process is complete.

Reading this register also advances XMIT AREA so that another transmit may be queued. This register MUST be read after all of the data has been written to XMIT AREA and before the next transmit request is written to XMIT AREA.

If an error code (6 (hex), 7 (hex), a (hex), or ff (hex)) is returned by XMIT QUEUE STATUS, then the frame was not queued and the host will have to attempt to queue it up another time. If the frame was flagged as being too long, it will have to be broken up into multiple frames before another queue attempt can be made.

If the code indicating success (2) is returned, then the host may immediately proceed to attempt to queue up an additional frame. The number of frames that can be queued depends on the amount of RAM on the adapter allocated for this purpose and the amount of immediate data included in each frame.

If the host attempts to queue a frame that is too large and also exceeds the available TRANSMIT DESCRIPTOR free space, the error that occurs first will take precedence and will be returned to the host.

H. XMIT START THRESH

XMIT START THRESH provides for an early start of transmission.

The XMIT START THRESH register is used to specify the number of transmit bytes that must reside on the adapter before it will start transmission. Values greater than the maximum frame length will prevent this function from operating properly. The method for disabling this function is to set the register to zero. Bytes are counted starting with the first byte of the destination field of the transmit frame.

The number of bytes considered to be available is the sum of the immediate data written to XMIT AREA by the host and those bytes transferred to the transmit data buffers in the adapter using bus master DMA operations. The transmit request will be posted immediately after XMIT START THRESH transmit frame bytes are made available from the immediate data or when the adapter has bus-mastered XMIT START THRESH-XMIT IMMED LEN bytes onto the adapter.

The number of bytes resident on the adapter must be equal to or greater than the value in XMIT START THRESH for the transmission to commence, unless the total frame size is less than XMIT START THRESH. In that case, the frame will begin transmission when the entire frame has been copied to the adapter. The actual transmission of the frame may be delayed by previous pending transmit frames and by deferrals to network traffic. This register is set to zero during a reset.

I. TRANSMISSION RING MANAGEMENT

FIGS. **10A–10E** illustrate the progression of the pointers used in the transmit operation. In FIGS. **10A–10E**, a portion of the transmit descriptor ring generally **200** and the transmit data buffers **201-0** and **201-1** are shown. Also, the pointers abbreviated as discussed above are indicated. One might also represent the transmit descriptor ring **200** in a circular fashion, as done below in FIGS. **12A–12F** for the receive ring.

In FIG. **10A**, the host descriptor logic is writing a first descriptor into the transmit descriptor ring **200**. Thus, the CXD pointer points to the base address of the first descriptor, the HW pointer points to the offset from the base address to which the host is expected to write the next double word of the descriptor. The download pointers including CDD, CDB are also pointing to the base address of the first descriptor as no download operations have begun. Likewise, the transmit descriptors CXD and XR point to the same base address. Finally, the tail of the ring XT points to the beginning descriptor. The download data pointer DD points to the top of a first buffer, for instance, buffer zero.

As illustrated in FIG. **10B**, the first descriptor has completed writing and has begun the download process, and the host has begun writing a second descriptor. Thus, the host descriptor logic pointer CHD points to the base address of the next descriptor, and the HW pointer points to the expected address of the next byte. The download pointer CDD points to the base address of the first descriptor. The down DMA logic is assumed to be in the process of transferring buffers from host into the transmit data buffer. Thus, the CDB pointer points to a descriptor of a download buffer in the first descriptor, and the DD pointer points to an offset within the transmit data buffer at which data is being downloaded. No transmit operations are yet to begin because the threshold has not been reached. Thus, the transmit pointers and the tail pointer still point to the first descriptor.

In FIG. **10C**, the host descriptor logic is working on a third descriptor, the download logic is working on the second descriptor, and the transmit logic is working on the first descriptor. The host descriptor logic pointers CXD and XR are working on the first descriptor. Thus, the CXD pointer points to the base address of the first descriptor, and the XR pointer points to immediate data being read for transmission by the transmit DMA logic.

The transmit read pointer XR will complete reading the immediate data and then move to the transmit data buffer filled by the download DMA logic when processing the first descriptor as illustrated in FIG. **10B**.

The transmit tail pointer XT still points to base address of the first descriptor.

The download logic is working on the second descriptor. Thus, the CDD pointer points to the base address of the second descriptor, the CDB pointer points to a buffer descriptor within the second descriptor, the DD pointer points to an offset within the second transmit data buffer to which the download DMA logic is transferring data from the host.

In FIG. **10C**, the host is writing a third descriptor into the XMIT AREA register. Thus, the CHD pointer points to the base address of the third descriptor, and the HW pointer points to the offset to which the next byte is expected to be written by the host.

In FIG. **10D**, the process has moved on so that the host is writing to a fourth descriptor. Thus, the CHD pointer points to the base address of the fourth descriptor, and the HW pointer points to the expected address of the next write to the XMIT AREA register.

The download logic is working on the third descriptor. Thus, the CDD pointer points to the base address of the third descriptor, the CDB pointer points to a buffer descriptor within the third descriptor, and the download data DD pointer points to a location in the first data buffer at which the download is occurring. This operation assumes that the transmit of the first descriptor has freed up the first data buffer for use by the download logic.

The transmit logic is working on the second descriptor. Thus, the CXD pointer points to the base address of the second descriptor, and the XR pointer points to a location in the transmit data buffer from which data is being read by the transmit logic. Since the status of the first descriptor is yet to be read, the transmit tail XT pointer still points to the base address of the first descriptor.

In FIG. **10E**, the process is assumed to have progressed so that the write of the fourth descriptor is completed, but the host has ceased writing new descriptors temporarily. In this case, the CHD pointer and the HW pointer point to the base address of a fifth descriptor waiting for further actions by the host. It is assumed that the download process has yet to complete downloading the third descriptor. Thus, the current CDD pointer points to the base address of the third descrip-

tor, and the CDB pointer points to a buffer descriptor within the third descriptor. The DD pointer points to a location in the transmit data buffer to which the download process is downloading data.

In FIG. **10E**, it is also assumed that the transmission of the frame identified by the second descriptor is complete, and the transmit logic is waiting for the download operation on a third descriptor to either complete, or download sufficient data that the transmit logic may begin transmission. Thus, the CXD and the XR pointers point to the base address of the third descriptor.

It is assumed that the host has also read the transmit status of the first descriptor. This has the effect of moving the XT pointer to the base address of the second descriptor as illustrated in FIG. **11E**.

This process continues with automatic ring wrap-around of the descriptors handled by the adapter. Also, overrun and underrun conditions are monitored and appropriate error signals indicated by the adapter.

VI. The Early Transmit Logic

The early transmit logic **6A** illustrated in FIG. **1** is implemented primarily in the download DMA logic **58** in the preferred implementation. The key components of a preferred implementation include data path arithmetic described with reference to FIG. **11**, and transmit start control logic illustrated with respect to FIGS. **12–17**.

FIG. **11** illustrates the basic logic of the data path arithmetic. This logic is founded primarily on 11 bit counter **300**. The 11 bit counter **300** includes a test input, a delta input, a counter enable CE input, a clock input, and a reset R input. The outputs of the 11 bit counter **300** include a carryout TC, and a counter output Q **[10:0]**. The test input is used for testing unrelated to the present invention. The delta input is driven by the download DMA byte enable **[3:0]**, and sets an increment amount for the counter **300** between 1 and 4, based upon the number of bytes in the 32 bit bus that are active in the download process.

The counter enable CE input is driven by the output of AND gate **301**. The inputs to the AND gate **301** include the cycleDone signal, and the output of delay circuit **302**. The delay circuit **302** receives as inputs the clock and the buffer download acknowledge bufferDwnAck signal. The cycleDone signal is asserted during the last clock of a transfer cycle, and is used in concert with other control signals to generate read and write strobes within the download DMA. The bufferDwnAck signal is a cycle acknowledge signal for buffer download autonomous accesses on the adapter. It is asserted in response to a buffer download request. The delay circuit **302** operates to insure that it stays true for at least as long as the cycleDone signal to insure proper incrementing of the counter.

The cycleDone signal is generated during the last clock of a transfer cycle. It is the basis for incrementing the counter **300** by the amount indicated by data path byte enable **[3:0]**.

The clock input **300** is driven by the system clock. The reset input is driven by OR gate **303**. The inputs to the OR gate include a newDescriptor signal indicating when a new descriptor is being processed by the download buffer, a downloadDmaReset signal, and a clearAllCounters signal for clearing counters on the adapter. The output of the counter **300** is supplied on an 11 bit line Q**[10:0]**. This output is supplied to adder **304**. The second input to the adder **304** is the immediate length value immedLenValue **[10:0]** which is read from the current descriptor. The sum of the immediate length value and the output of counter **300** is supplied as the downloadBytesResident value **[10:0]**.

The data path arithmetic in FIG. **11** also illustrates generation of the bytesResidentValid signal. This signal is generated by a sequence of D-type flip-flops **305, 306** which cause the bytesResidentValid signal, which is normally true, to go inactive for three clocks whenever the counter changes value. This allows for the download bytesResidentValue **[10:0]** to settle out of the adder and propagate through a comparator in the transmit start control logic described below. Thus, the D-type flip-flops **305, 306** have a reset input which is driven by the output of AND gate **301**, and a clock driven by the clock signal. Thus, these flip-flops are reset every time the counter **300** is incremented.

This logic is also used to generate a dataReady signal at the output of OR gate **307**. The inputs to this OR gate **307** are the dataPathByteEn signal **[3:0]**. Thus, the dataReady signal is asserted whenever a valid dataPathByteEn signal is generated.

A downloadCarryOut signal is generated for test purposes by the carryout signal TC of the counter **300**.

This logic is also used to generate a downloadDataPointer **[11:2]**. The downloadDataPointer is generated at the output of adder **308**. The inputs to the adder **308** include bits **[10:2]** of the output Q of counter **300**, and the output of the transmit buffer select MUX **309**. The inputs to the transmit buffer select MUX **309** include the offsets of the two transmit data buffers which are selected in response to transmit buffer select signal at the select input. Finally, this logic generates a dataPathOffset signal **[1:0]** at the output of the subtractor **310**. The subtractor generates a difference between signals A and B supplied at its inputs. Signal A is supplied at the output of latch **311**. The latch **311** latches the two least significant bits **[1:0]** of the output Q of the counter **300**. These two bits are latched in response to the doABuffer signal. The doABuffer signal is a command supplied to the buffer download logic to perform a bus master download of a buffer described in a transmit descriptor. Thus, these two bits are latched at the start of a download operation. The corresponding bits in the 32 bit download start address are subtracted from them to determine the data path offset for the buffer transfer.

Thus, this logic is coupled with the host address and monitors the downloading of data from the host into the transmit data buffer for use in making a threshold determination of the amount of data of a given frame resident in the adapter memory.

The downloadBytesResident **[10:0]** value is supplied to the transmit start control logic described with reference to FIGS. **12–15**. A block diagram of the transmit start control logic is provided in FIG. **12**. Each of the functional components of the block diagram in FIG. **12** are described with reference to FIGS. **13–17**.

FIG. **12** is a block diagram of the transmit start control logic. The transmit start control logic includes a start threshold register **320**, and a download compare block **321** which compares the value of the start threshold with the download bytes resident value generated in the data path arithmetic of FIG. **11**. Also, an immediate data comparator **322** is included to perform the comparison in the event that immediate data is present in the transmit descriptor. Finally, a data available control block **323** is included which generates the xmit-

DataAvailable signal for supply to the transmit DMA logic block of the adapter.

The transmit start control logic in FIG. **12** is responsible for generating the xmitDataAvailable signal, which is supplied to the transmit DMA module to indicate that there is data available to be transmitted onto the network. The xmitDataAvailable signal is asserted under three conditions: (1) there are one or more complete frames resident on the adapter ready to be transmitted; (2) there are no complete frames on the adapter, but the number of bytes resident for a frame currently being downloaded is greater than the value in the transmit start threshold register; and (3) there are no complete frames on the adapter, and there are no pending download operations, and the number of immediate bytes resident for the frame currently being written to the transmit area register is greater than the value of transmit start threshold.

The start threshold register **320** contains the physical XMIT START THRESH register, and logic to determine when the value in the register is valid. In particular, if the XMIT START THRESH value is written one byte at a time, then the START THRESH VALID signal is deasserted until the second byte is written. This keeps the download compare **321** and immediate compare **322** modules from making comparisons on transitional values. The inputs to the start thresh register **320** include the hostWriteData [**10:0**] which carries data through the start thresh register **320** during writes to the XMIT START THRESH register in the host interface. The startThreshWrite [**1:0**] signal are write strobes for the XMIT START THRESH register. The outputs of the start thresh register **320** include the startThreshValue [**10:0**] and the startThreshValid signal.

The inputs to the download compare module **321** include the startThreshValue [**10:0**] and the startThreshValid signal. Also, the downloadBytesResident [**10:0**] and the bytesResidentValid signals are supplied as inputs to the compare module **321**. The output of the download compare module **321** is the downloadThreshMet signal.

The inputs to the immediate compare module **322** include the startThreshValue [**10:0**] and the startThreshValid signal. Also, the immediate length value immedLengthValue [**10:0**] read from the transmit descriptor, the immedLengthValid signal. The immedLengthValid signal indicates that the value on immedLengthValue [**10:0**] is valid. Other inputs include the otherHostWrite [**3:0**] signal which are strobes indicating a host write of immediate data or buffer descriptor data. The four bits of the signal correspond to transfers on the four EISA bus byte lanes. The queueStatusRead signal indicating the host is reading status to prepare writing of a new transmit descriptor, the downloadDmaReset signal, which is a control signal from other portions of the adapter, a clock, a queueError signal, which indicates errors in host writes to the transmit area register; and a testAllCounters and clearAllCounters signals used for diagnostics and testing.

The output of the immediate compare module **322** is the immediateThreshMet signal. The downloadThreshMet and immediateThreshMet signals are supplied to the data available control block **322**. The other inputs include the downloadDmaReset signal, the clock signal, and the queueError signals. Also, the zeroDownloadsPending signal and zeroFramesResident signal are supplied to the data available control block **323**. These signals indicate when there are no complete transmit frames on the adapter, and when there are no pending download operations occurring, respectively. The output of the data available control block **323** is the xmitDataAvailable signal.

FIG. **13** illustrates the implementation of the start thresh register **320** of FIG. **12**. This register is composed of a 3 bit register **330** and an 8 bit register **331** to accommodate high and low bytes of the 11 bit threshold value. The 3 bit register stores bits **10:8** of the hostWriteData [**10:0**]bus. The 8 bit register **331** stores bits **7:0**. The 3 bit register **330** is clocked by the startThreshWrite [**1**] strobe and the 8 bit register **331** is clocked by the startThreshWrite [**0**] strobe. Both registers are reset on downloadDmaReset. The outputs of the registers **330** and **331** are supplied as the xmitStartData [**10:0**].

A start thresh valid state machine **332** is included. This state machine **322** receives as input the startThreshWrite strobes [**1:0**], the downloadDmaReset signal, and the clock. It contains a simple state machine as illustrated in FIG. **14** to monitor the startThreshWrite [**1:0**] and deassert the startThreshValid signal whenever one of the startThreshValue bytes is invalid. Thus, the state machine includes 3 states. The first state INIT1 **335** is entered on downloadDmaReset. If the startThreshWrite [**0**] signal is true, and the startThreshWrite [**1**] signal is false, then the state machine goes to the LOW_BYTE_WRITTEN state **336**. When the startThreshWrite [**1**] signal is next asserted, then the state machine returns to the INIT1 state **335**. If, in the INIT1 state **335**, the startThreshWrite [**1**] signal is valid and the startThreshWrite [**0**] signal is not true, then the third state HIGH_BYTE_WRITTEN **337** is entered. When the startThreshWrite [**0**] signal is asserted, then the state machine returns to the INIT1 state **335**, during which the startThreshValid signal is asserted.

The download compare module **321** is illustrated in FIG. **15**. This logic includes a comparator **340** which receives a downloadBytesResident [**10:0**] value from the data path arithmetic of FIG. **11**, and the xmitStartData [**10:0**] from the start thresh register **320**. If the downloadBytesResident value is greater than or equal to the xmitStartData value, then the output of the comparator is asserted. The output of the comparator **340** is supplied to AND gate **341**. The AND gate **341** includes three inputs in addition to the output of the comparator **340**. The inputs include the startThreshValid signal from the start thresh register module **320**, and the output of logic **342** which tests whether the xmitStartData value is equal to zero. If it is not equal to zero, then the signal is asserted.

Finally, the bytesResidentValid signal from the data path arithmetic must be asserted before the downloadThreshMet signal is asserted through the AND gate **341**.

The immediate compare module **322** is illustrated in FIG. **16**. The immediate compare module **322** comprises an 11 bit counter **350**, a first comparator **351**, a second comparator **352**, and a third comparator **353**. Also, the output of the comparators is supplied through multiplexer **354**. The immedThreshMet signal is generated at the output of AND gate **355**. Also, an OR gate **356** and not equal to zero test logic **357** is included. The immediate compare module keeps track of the number of immediate data bytes resident using the 11 bit counter **350**. The inputs to the 11 bit counter include a test signal, otherHostWrite [**3:0**] signal, which indicates the delta or increment added by each clock, and a clock signal.

The 11 bit counter **350** is reset at the output of OR gate **356**. The inputs to the OR gate **356** include the downloadDmaReset signal, the queueStatusRead signal, the queueError signal, and the clearAllCounters signal. The outputs include an immedCarryout signal at the TC output, and the Q [10:0] bus which indicates the number of immediate bytes stored. The immediate bytes output is supplied as inputs to the first and second comparators **351** and **352**. The second input to the first comparator **351** is the immedLengthValue [10:0] which is retrieved from a descriptor currently being written. The second input to the second comparator **352** is the xmitStartData [10:0] from the start threshold register module **320**.

A third comparator **353** receives inputs including the immedLengthValue and the xmitStartData value.

The first comparator **351** generates a "clipped" signal in the event that the immediate bytes is greater than the immedLengthValue. The second comparator **352** generates the countCrossedThresh signal in the event that the immediate bytes is greater than or equal to the immedLengthValue. The third comparator **353** generates an immedCrossedThresh signal in the event that the immedLengthValue is greater than or equal to the xmitStartData value. The clipped signal is supplied at the select input of multiplexer **354**. This operates to cause the multiplexer **354** to select the countCrossedThresh signal as output Y so long as the number of immediate bytes does not exceed the immedLengthValue. When the number of immediate bytes exceeds the immedLengthValue, then the immedCrossedThresh signal is supplied at the output Y of the multiplexer.

The 11 bit counter **350** is incremented by 1, 2, 3, or 4, depending on the value of otherHostWrite [3:0].

The immediateThreshMet signal is supplied at the output of the AND gate **355**. The inputs to the AND gate **355** include the output of the logic **357** which is asserted when the xmitStartData value is non-zero. Also, the startThreshValid and immedLengthValid signals are supplied as inputs to the gate **355**. When all three of these signals are asserted, then the output of the multiplexer **354** is enabled for supply as the immediateThreshMet signal. When immediateBytes is less than immedLengthValue, clipped is inactive, and the output of comparator **352** is supplied as immediateThreshMet. When immediateBytes exceeds the immedLengthValue, then the value of immediateBytes is disregarded, and the immedCrossedThresh comparator **353** takes over.

The assertion of queueStatusRead clears the counter, ensuring that it is always properly synchronized with the writing of a new frame descriptor to transmit area.

The data available control module **323** consists of a four-state state machine illustrated in FIG. **17**. The four states include an INIT2 state **370**, an IMMEDIATE_THRESH_MET state **371**, a DOWNLOAD_THRESH _MET state **372**, and a FRAME_RESIDENT state **373**. The INIT2 state **370** is entered upon downloadDmaReset. If the zeroFramesResident, and zeroDownloadsPending, and immediateThreshMet signals are true, then the IMMEDIATE_THRESH_MET state **371** is entered. In this state, the xmitDataAvailable signal is asserted. The IMMEDIATE_THRESH_MET state returns to the INIT2 state **370** upon the assertion of queueError. Alternatively, when the zeroDownloadsPending signal is deasserted and the queueError signal is not asserted, then the control proceeds from the IMMEDIATE_THRESH_MET state **371** to the DOWNLOAD_THRESH_MET state **372**. During this state, the xmitDataAvailable signal is true. If the zeroFramesResident signal is not true, then the state machine proceeds to the FRAME_RESIDENT state **373** from the DOWNLOAD_THRESH_MET state **372**. In this state, the xmitDataAvailable signal remains asserted. Upon assertion of zeroFramesResident, the state machine returns to the INIT2 state **370** where xmitDataAvailable is false.

Alternatively, from the INIT2 state **370**, if the zeroFramesResident signal is deasserted, then the state machine proceeds directly to the FRAME_RESIDENT state **373**. Also, in the event that the zeroFramesResident signal is true, the zeroDownloadsPending signal is false, and the downloadThreshMet signal is true, then the state machine proceeds from the INIT2 state **370** to the DOWNLOAD_THRESH_MET state **372**. The state machine prevents transitions on the data available control input variables (downloadThreshMet, immediateThreshMet, zeroDownloadsPending, zeroFramesResident) from causing spurious indications on the xmitDataAvailable control line.

VII. Underrun Control Logic

The transmit logic **39** of FIG. **2** includes underrun control logic coupled to the data path as shown in the simplified block diagram of FIG. **18**. In FIG. **18**, the transmit data path **400** is illustrated receiving transmit bytes across line **401** from the transmit DMA logic. Also, a transmit write signal TXWR is supplied on line **402** to the transmit data path **40**. The transmit data path receives the transmit bytes across line **401** and serializes them for supply on output line **403**. Also, data from the frame is supplied across line **404** to CRC logic **405** for generating error detection codes to be appended to the end of frames according to standard media access control functions. The output of the CRC logic **405** is supplied on line **406** to an exclusive OR gate **407**. Exclusive OR gate **407** passes the CRC code through to line **408** unless a bad frame signal on line **409** is asserted.

The data on line **403** and the CRC on line **408** are supplied to a multiplexer **410** which is controlled by transmit control logic **411** to supply the data of the frame followed by the CRC of the frame for valid frames to the serial output line **412**. Line **412** is then coupled to a transceiver for transmission on the network.

According to the present invention, this transmit data path includes an underrun detector **413** for detecting a condition in which the transferring of data into the transmit data buffer, or immediate data to the transmit descriptor buffer, by the host interface falls behind the transferring of data into the transmit data path **400** by the transmit DMA logic. The underrun detector **413** is controlled by the transmit control logic **411**. The transmit control logic **411** indicates intervals across line **414** during which a transmit write TXWR signal is expected on line **402**. The underrun detector determines that a transmit write TXWR signal is not present during an expected interval of the frame transmission, then a bad frame signal is generated on line **409**. In response to the bad frame signal, the CRC data is inverted by the exclusive OR gate **407** which causes a bad CRC to be generated for the already transmitted portions of the frame suffering the underrun. Transmit control logic **411** also responds to the bad frame signal on line **409** to select the bad CRC data through multiplexer **410**. Finally, the bad frame signal on line **409** is used for posting status infor-

mation through the xmitFailureRegister of an underrun condition.

VIII. Conclusion

Thus, the present invention allows for early transmission start in the system described above. By loading the XMIT START THRESH register, the system is enabled to begin transmission of a frame onto the network media prior to transfer of the entire frame into the adapter's buffer. The early transmission start is enabled by setting the XMIT START THRESH register to a non-zero value.

XMIT START THRESH is used to specify the number of bytes of the transmit frame that must reside on the adapter, either as immediate data or as data transferred onto the adapter via bus master data transfers, or as a combination of the two, before the adapter can commence with the media access control functions associated with transmitting the frame.

If the number of immediate data bytes expected by the adapter (as determined by the value written to the XMIT IMMEDIATE LENGTH field of the XMIT AREA) exceeds the value set in XMIT START THRESH, then the adapter will begin transmitting the frame immediately after sufficient writes to the XMIT AREA have occurred. Otherwise, the start of transmission will be delayed until, as a bus master, the adapter has downloaded enough data into its onboard buffer to satisfy the early transmit start threshold requirement. If, however, the threshold value is greater than or equal to the frame length, then transmission will commence once the entire frame is resident on the adapter. Of course, the start of the transmission may be delayed by other queued transmission, or by deferral to network traffic.

If this register set to zero, then the early transmit feature is disabled and the entire transmit frame must reside on the adapter before the adapter will begin to transmit it.

The value for this register may be programmed by the host to optimize performance. If set too low, system latencies or bandwidth limitations may cause the adapter to underrun the network during transmission, causing a partial frame with a guaranteed bad CRC to be transmitted. If the value is set too high, then unnecessary delays will be incurred before the start of transmission. The adapter generates an indication of an underrun condition which is made available to the host through the XMIT FAILURE register. If such an underrun indication occurs, then the host driver should increase the value on the XMIT START THRESH register. Further underrun indications should cause the driver to continually increase the XMIT START THRESH value. If the XMIT START THRESH value is increased to a value of greater than the maximum length expected by the system, then the early transmit start features should be disabled by writing a zero to the XMIT START THRESH register.

The foregoing description of preferred embodiments of the present invention has been provided for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Obviously, many modifications and variations will be apparent to practitioners skilled in this art. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, thereby enabling others skilled in the art to understand the invention for various embodiments and with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims and their equivalents.

What is claimed is:

1. For a system transmitting frames of data across a communications medium; an apparatus comprising:
   - buffer memory for storing data of frames composed by the host computer for transmission on the communications medium;
   - means, having a host system interface, for transferring data of frames to the buffer memory;
   - means, coupled with the buffer memory, for monitoring the transferring of data of a frame to the buffer memory to make a threshold determination of an amount of data of the frame transferred to the buffer memory;
   - means, responsive to the threshold determination of the means for monitoring, for initiating transmission of the frame prior to transfer of all the data of the frame to the buffer memory from the host computer;
   - transmit logic, responsive to the means for initiating transmission, for retrieving data from the buffer memory and supplying retrieved data for transmission on the communications medium; and
   - underrun control logic, which detects a condition in which the means for transferring falls behind the transmit logic, and supplies a bad frame signal to the communications medium in response to the underrun condition.

2. The apparatus of claim 1, wherein the transmit buffer includes a transmit descriptor ring buffer and a transmit data buffer, the transmit descriptors including data identifying data to be transmitted on the communications medium and optionally immediate data, and wherein the means for monitoring includes the immediate data in the threshold determination.

3. The apparatus of claim 1, wherein the buffer memory comprises a buffer independent of the host address space.

4. The apparatus of claim 1, wherein the transmit logic includes means for appending an error detection code to transmitted frames, and the bad frame signal comprises a corrupted error detection code.

5. The apparatus of claim 1, wherein the means for monitoring includes a threshold store alterable by the system for storing a threshold value, and logic responsive to the threshold value for making the threshold determination.

6. The apparatus of claim 5, wherein the frames have an identifiable beginning and the threshold value specifies an amount of data from the beginning of the frame.

7. The apparatus of claim 1, wherein the buffer includes a transmit descriptor ring buffer and a transmit data buffer, and the means for transferring includes:
   - transmit descriptor logic for mapping transmit descriptors from the system to the transmit descriptor ring buffer; and
   - download logic, responsive to transmit descriptors in the transmit descriptor ring buffer, for retrieving data from memory in the system and storing retrieved data in the transmit data buffer.

8. The apparatus of claim 7, wherein the transmit descriptors further include immediate data to be transmitted on the communications medium, and wherein the means for monitoring includes the immediate data in the threshold determination.

9. The apparatus of claim 8, the transmit logic further including:

logic, responsive to transmit descriptors in the transmit ring buffer and the means for initiating transmission, for retrieving data from the transmit descriptor ring buffer and transmit data buffer and supplying retrieved data for transmission on the communications medium.

10. An apparatus for controlling communication between a host system and a network transceiver coupled with a network, comprising:

a buffer memory;

host interface means, having an interface to the host system, for transferring data between the host system and the buffer memory;

means, coupled with the buffer memory and including a host system alterable threshold store for storing a threshold value, for monitoring the transferring of data of a frame to the buffer memory to make a threshold determination of an amount of data of the frame transferred to the buffer memory; and

means, responsive to the threshold determination of the means for monitoring, for initiating transmission of the frame, prior to transfer of all the data of the frame to the buffer memory from the host computer;

network interface means, having an interface to the network transceiver and responsive to the means for initiating, for transferring data between the buffer memory and the network transceiver for transmission; and

control means, coupled with the network interface means, for posting status information for use by the host system, as feedback for optimizing the threshold value.

11. The apparatus of claim 10, wherein the host interface means includes:

transmit descriptor logic for transferring transmit descriptors identifying data to be transmitted from the host system to the buffer memory; and

download logic, responsive to transmit descriptors in the buffer memory, for retrieving data from memory in the host system and storing retrieved data in the buffer memory.

12. The apparatus of claim 11, wherein the transmit descriptors include immediate data.

13. The apparatus of claim 12, wherein the means for monitoring includes logic for determining amounts of immediate data in transmit descriptors.

14. The apparatus of claim 12, wherein the means for monitoring includes logic for determining an amount of immediate data in a transmit descriptor and an amount of data retrieved by the download logic in response to the transmit descriptor in making the threshold determination.

15. The apparatus of claim 10, wherein the control means further includes:

underrun control logic, which detects an underrun condition in which the host interface means in transferring data to the buffer memory falls behind the network interface means in transferring data to the transceiver, and means for supplying a bad frame signal to the communication media in response to the underrun condition.

16. The apparatus of claim 15, wherein the control means includes means for appending an error detection code to transmitted frames, and the bad frame signal comprises a corrupted error detection code.

17. The apparatus of claim 15, wherein the status information posted by the control means includes an indication of an underrun condition.

18. An apparatus for controlling communication between a host system and a network transceiver coupled with a network, comprising:

a buffer memory, including a transmit descriptor buffer for storing transmit descriptors identifying data in the host system to be transmitted and including immediate data, and a transmit data buffer for buffering data of frames transferred from the host system;

host interface means, having an interface to the host system, for transferring transmit descriptors and data of frames between the host system and the buffer memory and including means, responsive to transmit descriptors in the transmit descriptor buffer, for downloading data of frames from the host system to the transmit data buffer;

means, coupled with the buffer memory, for monitoring the immediate data in transmit descriptors in the transmit descriptor buffer and the downloading of data of a frame to the transmit data buffer to make a threshold determination of an amount of data of the frame transferred to the buffer memory; and

means, responsive to the threshold determination of the means for monitoring, for initiating transmission of the frame, prior to transfer of all the data of the frame to the buffer memory from the host computer;

network interface means, having an interface to the network transceiver and responsive to the means for initiating, for transferring data between the buffer memory and the network transceiver for transmission; and

underrun control means, coupled with the network interface means, for detecting an underrun condition in which the host interface means in downloading data to the transmit data buffer falls behind the network interface means in transferring data to the transceiver, and for supplying a bad frame signal to the network transceiver in response to the underrun condition.

19. The apparatus of claim 18, wherein the means for monitoring includes a threshold store alterable by the system for storing a threshold value, and logic responsive to the threshold value for making the threshold determination; and the underrun control means includes means for posting status information which may be used by the host system as feedback for optimizing the threshold value.

20. The apparatus of claim 18, wherein the network interface means includes means for appending an error detection code to transmitted frames, and the bad frame signal comprises a corrupted error detection code.

21. A network interface adapter for a carrier sense, multiple access network with collision detection (CSMA/CD), comprising:

buffer memory which stores data of frames composed by a host computer for transmission on the network;

data transfer circuitry, having a host system interface, for transferring data of frames to the buffer memory;

logic, coupled to the buffer memory, which monitors the transferring of data of a frame to the buffer memory to make a threshold determination of an amount of data of the frame transferred to the buffer memory;

a medium access controller for the CSMA/CD network coupled to the buffer memory for managing transmission of frames of data from the buffer memory to the network; and

logic, responsive to the threshold determination of the logic which monitors the transferring of data to the buffer memory, which initiates transmission of the frame from the buffer memory to the medium access controller prior to transfer of all of the data of the frame to the buffer memory, including logic which initiates transmission of the frame when no complete frame of data is present in the buffer memory.

22. The adapter of claim of **21**, wherein the buffer memory includes a transmit descriptor ring buffer and a transmit data buffer, the transmit descriptors including data identifying data to be transmitted on the network and optionally immediate data, and wherein the threshold logic includes the immediate data in the threshold determination.

23. The adapter of claim **21**, wherein the threshold logic includes an alterable threshold store which stores the threshold value, and logic responsive to the threshold value for making the threshold determination.

24. The adapter of claim **23**, wherein the frames have an identifiable beginning and the threshold value specifies an amount of data from the beginning of the frame.

25. The adapter of claim **21**, further including:

underrun control logic, which detects a condition in which the data transfer circuitry falls behind the medium access controller, and supplies a bad frame signal to the network in response to the underrun condition.

26. The apparatus of claim **25**, wherein the medium access controller includes logic which appends an error detection code to the transmitted frames, and a bad frame signal generated by the underrun control logic comprises a corrupted error detection code.

* * * * *

Exhibit 8

US005732094A

United States Patent [19]

Petersen et al.

[11] Patent Number: 5,732,094

[45] Date of Patent: *Mar. 24, 1998

[54] **METHOD FOR AUTOMATIC INITIATION OF DATA TRANSMISSION**

[75] Inventors: **Brian Petersen**, Mountain View; **David R. Brown**, San Jose; **W. Paul Sherer**, Sunnyvale, all of Calif.

[73] Assignee: **3Com Corporation**, Santa Clara, Calif.

[*] Notice: The term of this patent shall not extend beyond the expiration date of Pat. No. 5,434,872.

[21] Appl. No.: **715,253**

[22] Filed: **Sep. 16, 1996**

Related U.S. Application Data

[63] Continuation of Ser. No. 397,997, Mar. 3, 1995, abandoned, which is a continuation of Ser. No. 920,893, Jul. 28, 1992, Pat. No. 5,434,872.

[51] **Int. Cl.**[6] **G06F 13/36**

[52] **U.S. Cl.** **371/51.1**; 395/200.13; 395/872; 395/877

[58] **Field of Search** 371/51.1; 395/872, 395/877, 200.13

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,258,418	3/1981	Heath	395/250
4,590,467	5/1986	Lare	340/825.5
4,715,030	12/1987	Koch et al.	370/85.13
4,852,088	7/1989	Gulick et al.	370/94
4,860,193	8/1989	Bentley et al.	395/250
5,043,981	8/1991	Firoozmand et al.	370/85.1
5,119,374	6/1992	Firoozmand et al.	370/85.4
5,133,062	7/1992	Joshi et al.	395/500
5,133,078	7/1992	Minassian et al.	395/800
5,136,582	8/1992	Firoozmand	370/85.1
5,195,093	3/1993	Tarrab et al.	371/3
5,210,749	5/1993	Firoozmand	370/85.1
5,235,595	8/1993	O'Dowd	370/94.1
5,247,626	9/1993	Firoozmand	395/250
5,278,956	1/1994	Thomsen et al.	395/250
5,289,579	2/1994	Punj	395/200.2
5,293,487	3/1994	Russo et al.	395/200.2
5,299,313	3/1994	Petersen et al.	395/200.2
5,329,622	7/1994	Belsan et al.	395/250
5,412,780	5/1995	Rushton	395/250
5,434,872	7/1995	Petersen et al.	371/57.1
5,487,152	1/1996	Young	395/200.01

OTHER PUBLICATIONS

Tamir, Y., "Design and Implementation of High-Speed Asynchronous Communication Ports for VLSI Multicomputer Nodes", ISCAS '88, pp. 805–809, Aug. 1988.

Kwok, C., et al., "Cut-Through Bridging for CSMA/CD Local Area Networks", IEEE Transactions on Communications, vol. 38, No. 7, pp. 938–942, Jul. 1990.

Seigel, M., et al., "Overcoming Bottlenecks in High-Speed Transport Systems", Local Computer Networks, 16th Conf., pp. 399–407, Sep. 1991.

AMD, "The Supernet® 2 Family for FDDI", 1991/1992 World Network Data Book, (Oct. 1991), pp. 1–1 to A–4.

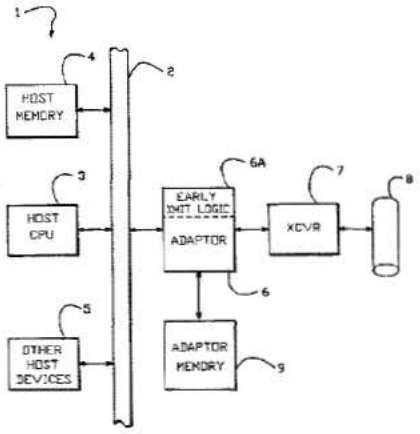
(List continued on next page.)

Primary Examiner—Stephen M. Baker

[57] **ABSTRACT**

Early initiation of transmission of data in a network interface that includes a dedicated transmit buffer is provided in a system which includes logic for transferring frames of data composed by the host computer into the transmit buffer. The amount of data of a frame which is downloaded by the host to the transmit buffer is monitored to make a threshold determination of an amount of data of the frame resident in the transmit data buffer. The network interface controller includes logic for initiating transmission of the frame when the threshold determination indicates that a sufficient portion of the frame is resident in the transmit buffer, and prior to transfer of all of the data of the frame into the transmit buffer. The monitoring logic includes a threshold store, which is programmable by the host computer for storing a threshold value. Thus, the threshold value may be set by the host system to optimize performance in a given setting.

53 Claims, 13 Drawing Sheets



OTHER PUBLICATIONS

Fickel, Louise, "AMD Chip Set Promises to Bring FDDI Closer to the Desktop", *Info World,* Jan. 21, 1991, ©1991 IDG Communications, Inc., p. 8.

"AMD Introduces FDDI Chip for half-size card", *The Local Area Network Magazine,* Feb. 1991, vol. 6, No. 2, ©1991 Miller Freeman, Inc., p. 18.

Wilson Ron, "FDDI chips struggle towards the desktop; Fibert Distributed Data Interface", *Computer Design,* Feb. 1, 1991, vol. 30, No. 3, ©1991 Pen Well Publishing Company, p.61.

Preliminary Specification for the Am79C830A FORMAC Plus Chip Integrated Circuit, *Advanced Micro Devices,* Jul. 1992, Publication No. 16287, pp.1–140.

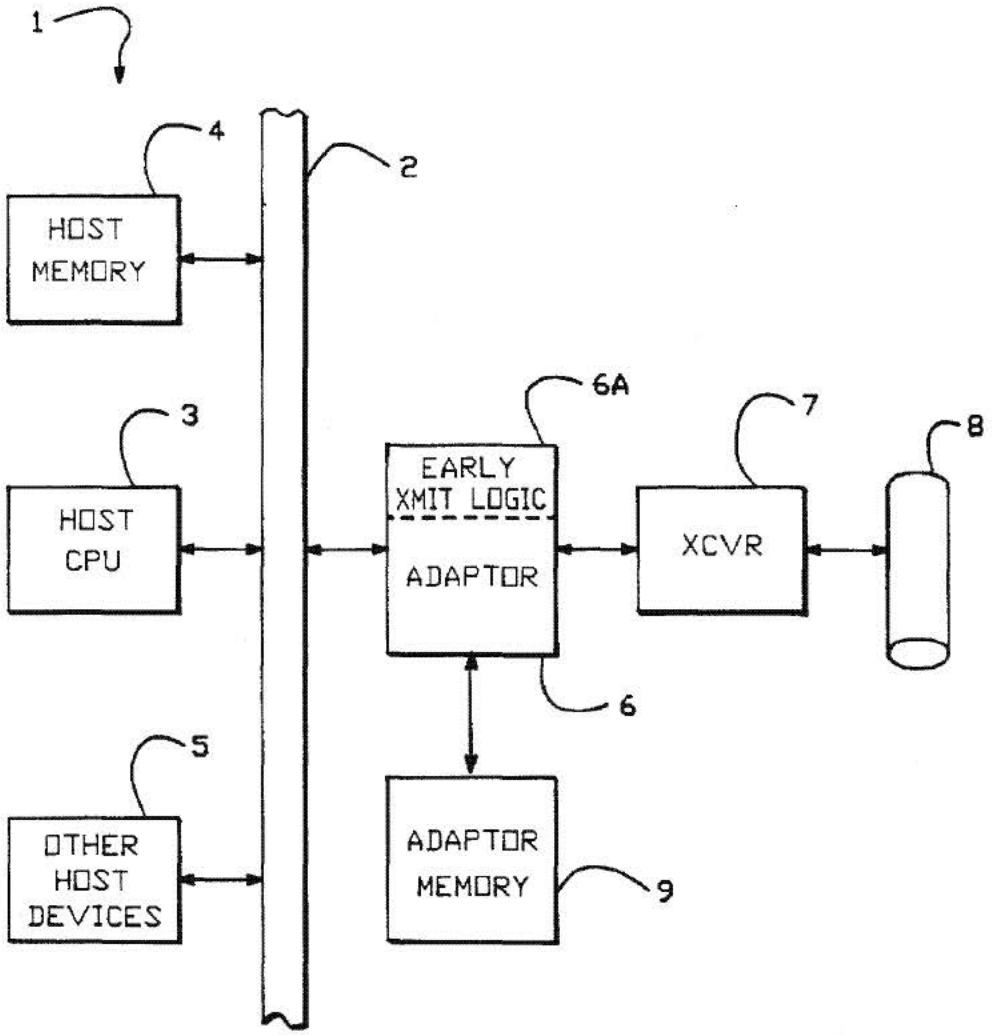

1
4
2
HOST MEMORY
3
HOST CPU
6A
EARLY XMIT LOGIC
ADAPTOR
7
XCVR
8
6
5
OTHER HOST DEVICES
ADAPTOR MEMORY
9

FIG.—1

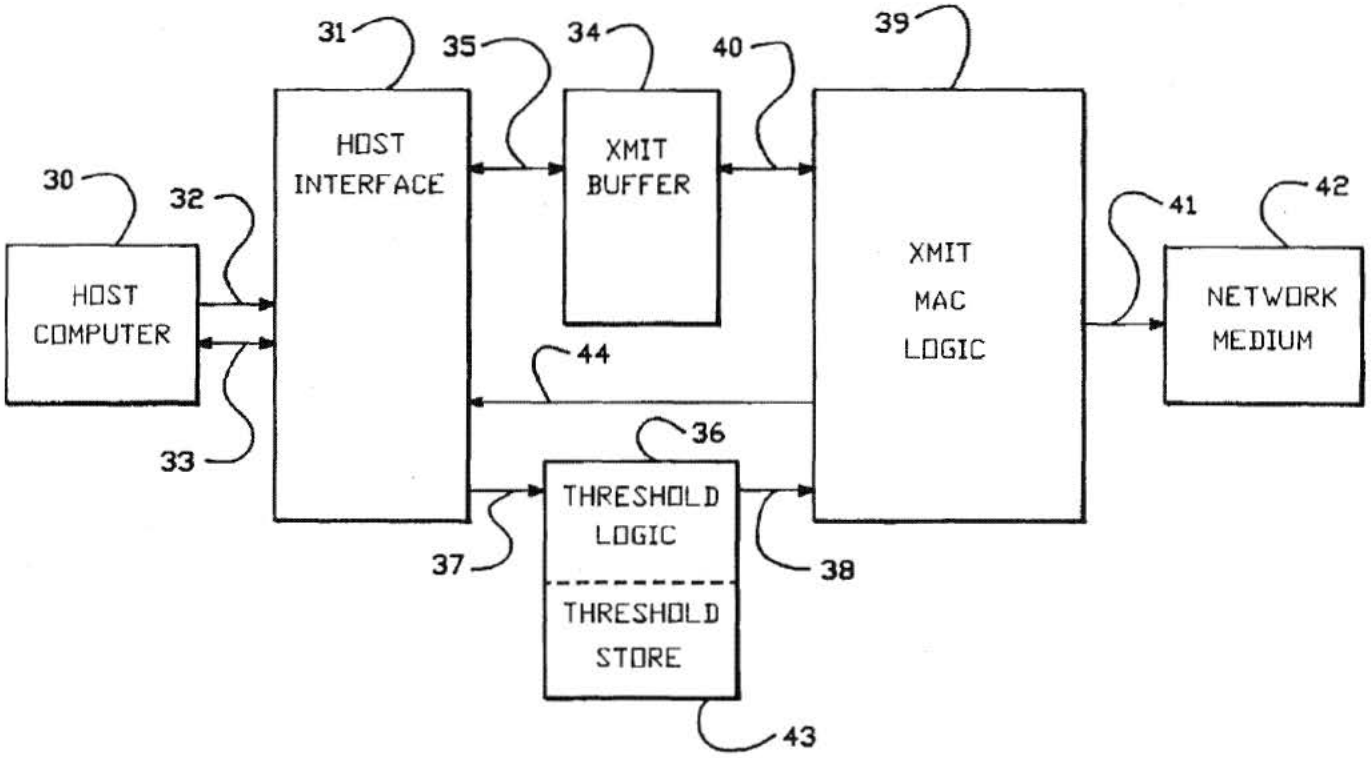

31
35
34
40
39
HOST INTERFACE
XMIT BUFFER
XMIT MAC LOGIC
30
32
41
42
HOST COMPUTER
NETWORK MEDIUM
44
33
36
THRESHOLD LOGIC
37
38
THRESHOLD STORE
43

FIG.—2

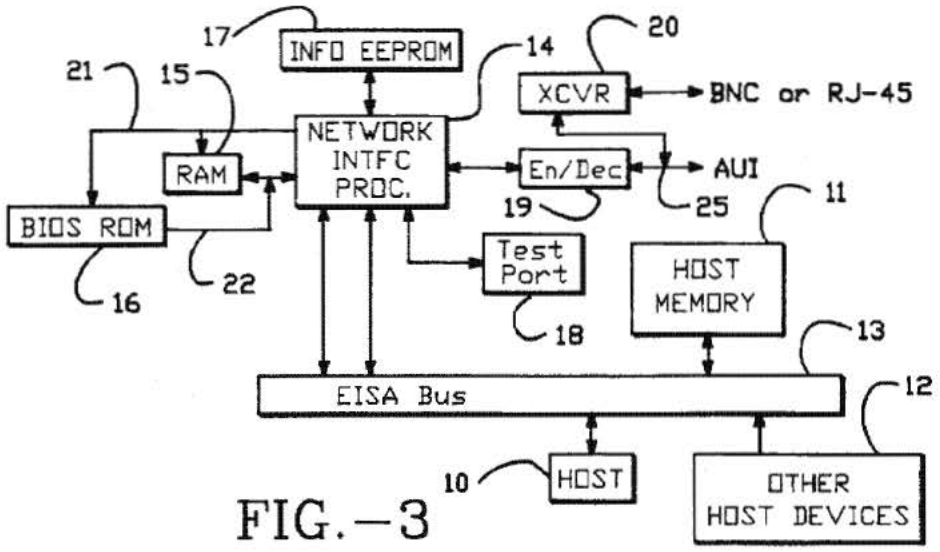

17
INFO EEPROM
14
20
XCVR
BNC or RJ-45
21
15
NETWORK INTFC PROC.
RAM
En/Dec
AUI
19
25
11
BIOS ROM
16
22
Test Port
18
HOST MEMORY
13
EISA Bus
12
10
HOST
OTHER HOST DEVICES

FIG.—3

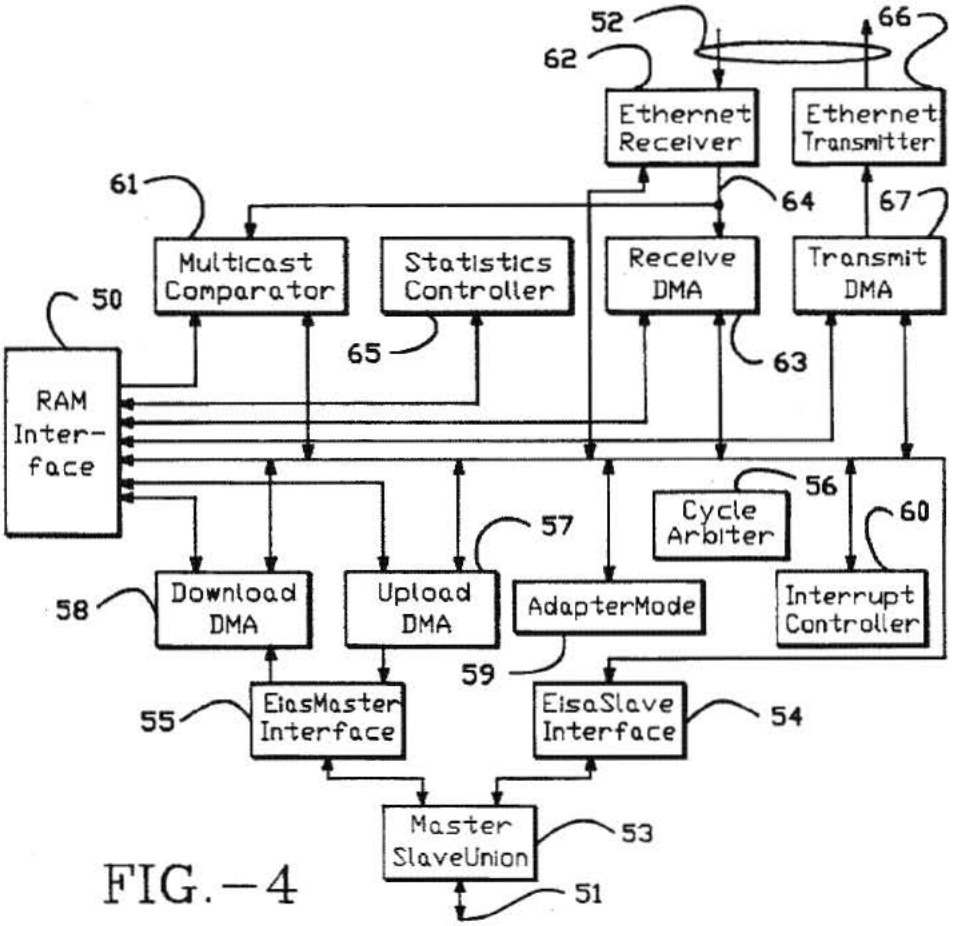

52
62
66
Ethernet Receiver
Ethernet Transmitter
61
64
67
Multicast Comparator
Statistics Controller
Receive DMA
Transmit DMA
50
65
63
RAM Inter-face
56
Cycle Arbiter
60
57
58
Download DMA
Upload DMA
AdapterMode
Interrupt Controller
59
55
EisaMaster Interface
EisaSlave Interface
54
Master SlaveUnion
53
51

FIG.—4

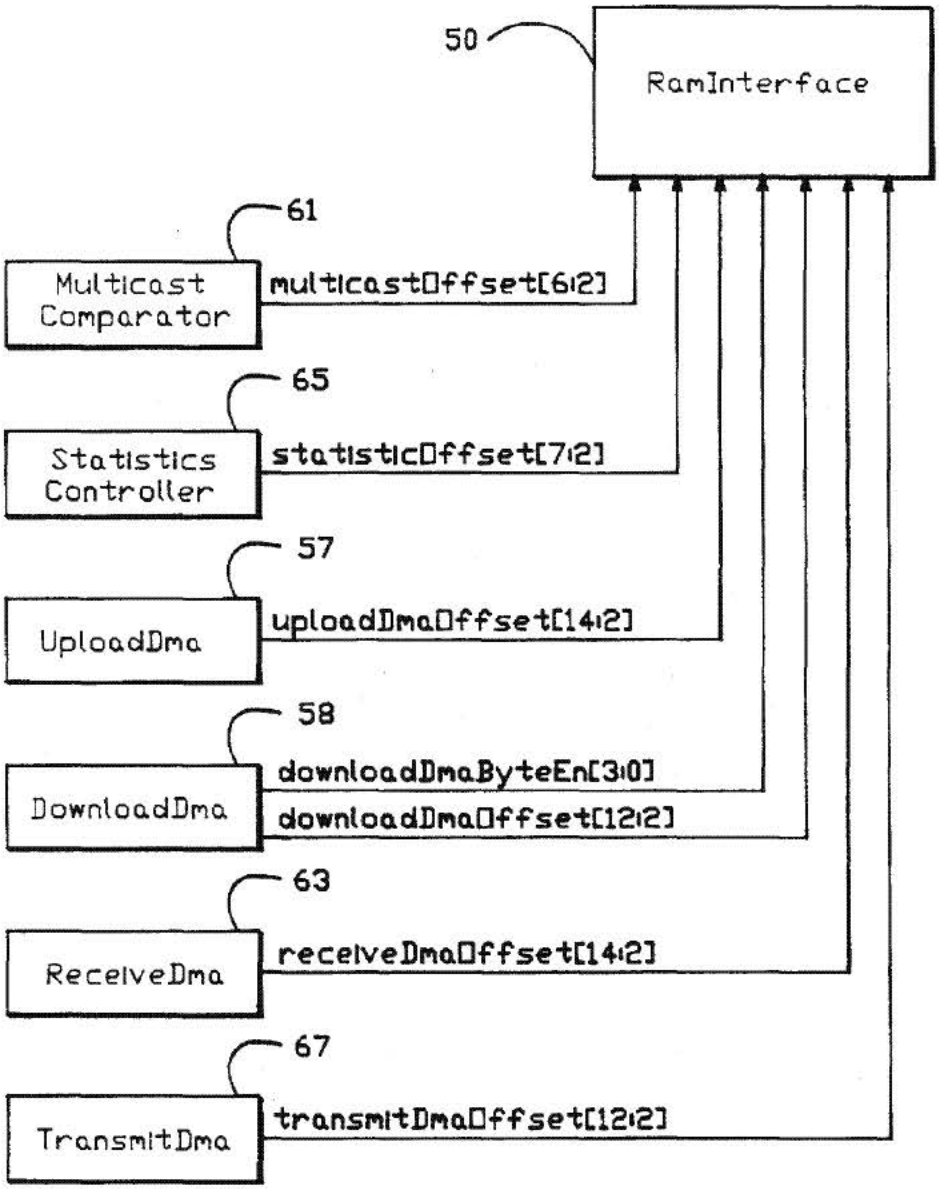

50
RamInterface
61
Multicast Comparator
multicastOffset[6:2]
65
Statistics Controller
statisticOffset[7:2]
57
UploadDma
uploadDmaOffset[14:2]
58
DownloadDma
downloadDmaByteEn[3:0]
downloadDmaOffset[12:2]
63
ReceiveDma
receiveDmaOffset[14:2]
67
TransmitDma
transmitDmaOffset[12:2]
Autonomous Access Address Bus Structure

FIG.—4A

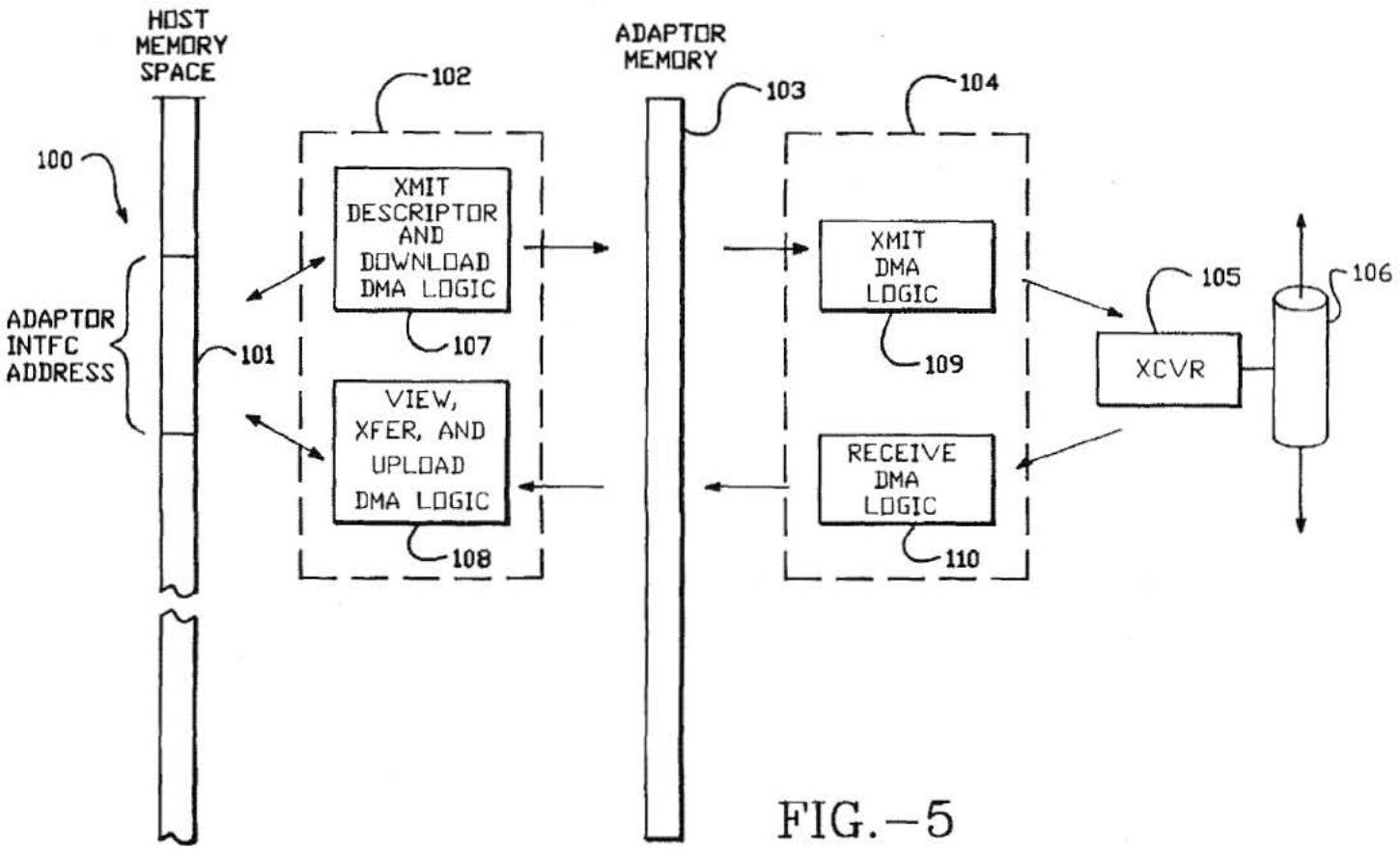

HOST MEMORY SPACE
100
ADAPTOR INTFC ADDRESS
101
102
XMIT DESCRIPTOR AND DOWNLOAD DMA LOGIC
107
VIEW, XFER, AND UPLOAD DMA LOGIC
108
ADAPTOR MEMORY
103
104
XMIT DMA LOGIC
109
RECEIVE DMA LOGIC
110
105
XCVR
106

FIG.—5

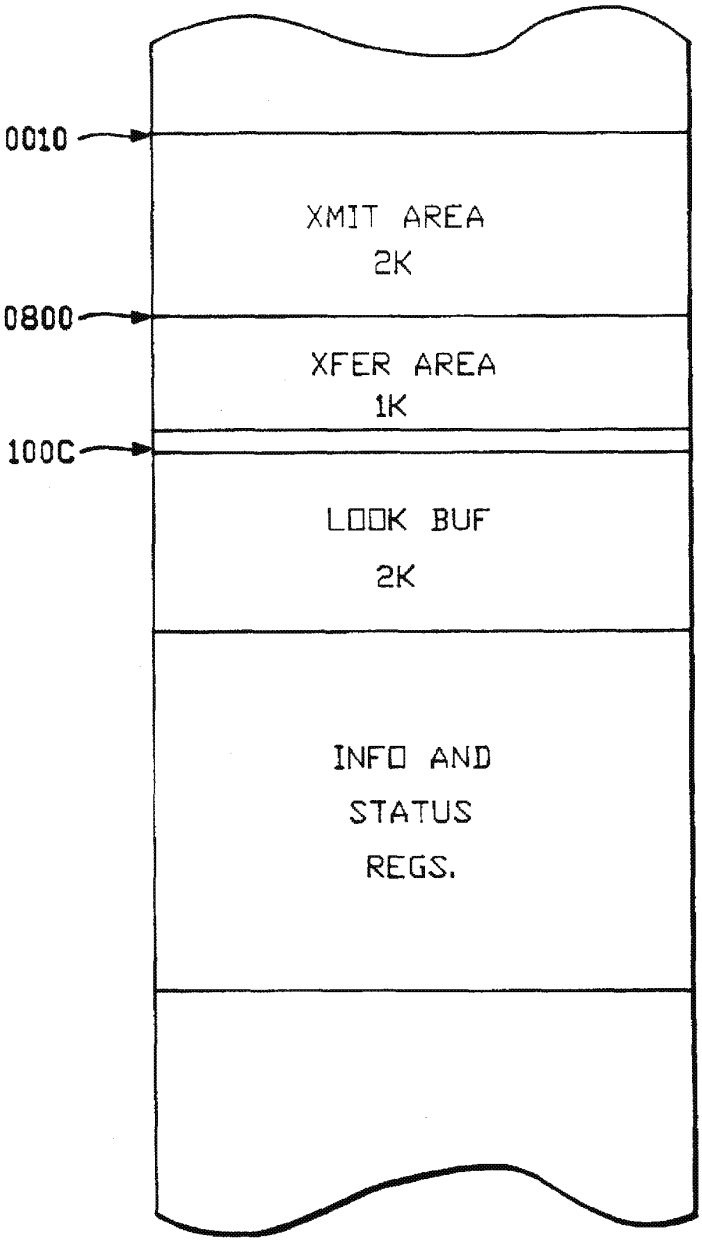
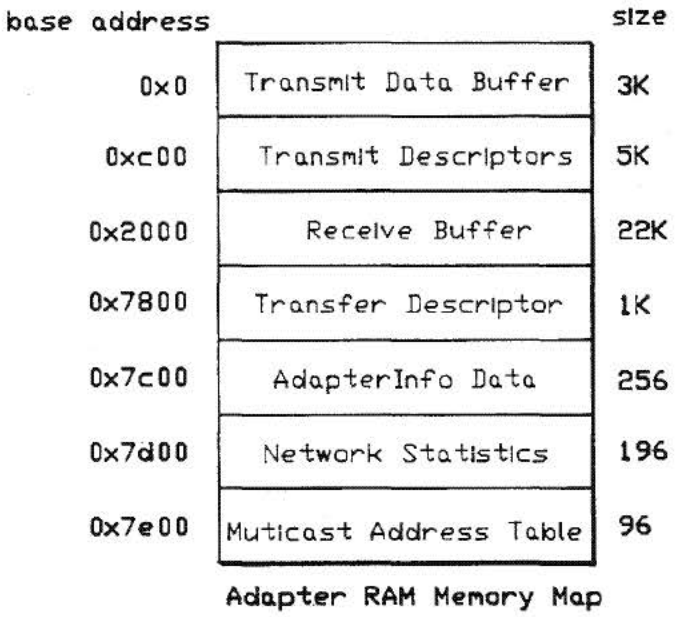

0010
XMIT AREA
2K
0800
XFER AREA
1K
100C
LOOK BUF
2K
INFO AND
STATUS
REGS.

FIG.—6



Adapter RAM Memory Map

FIG.—7

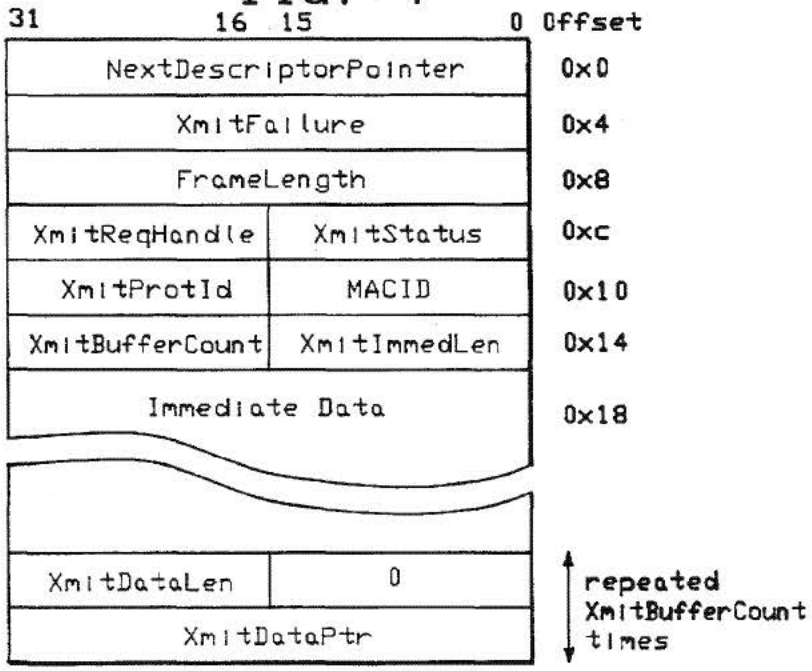


FIG.—8

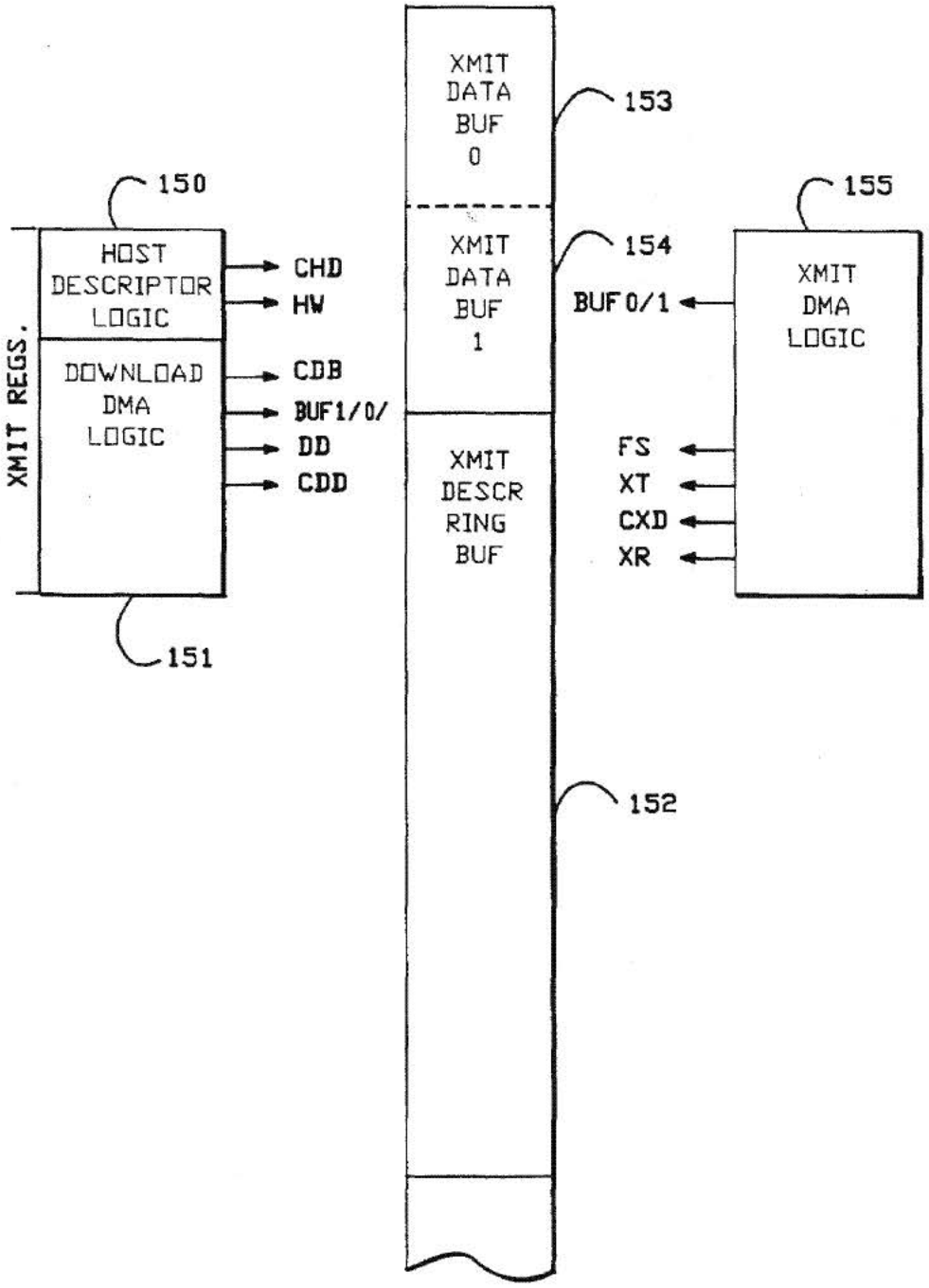

XMIT DATA BUF 0
153
150
155
HOST DESCRIPTOR LOGIC
CHD
HW
XMIT DATA BUF 1
154
BUF 0/1
XMIT DMA LOGIC
XMIT REGS.
DOWNLOAD DMA LOGIC
CDB
BUF1/0/
DD
CDD
XMIT DESCR RING BUF
FS
XT
CXD
XR
151
152

FIG.—9

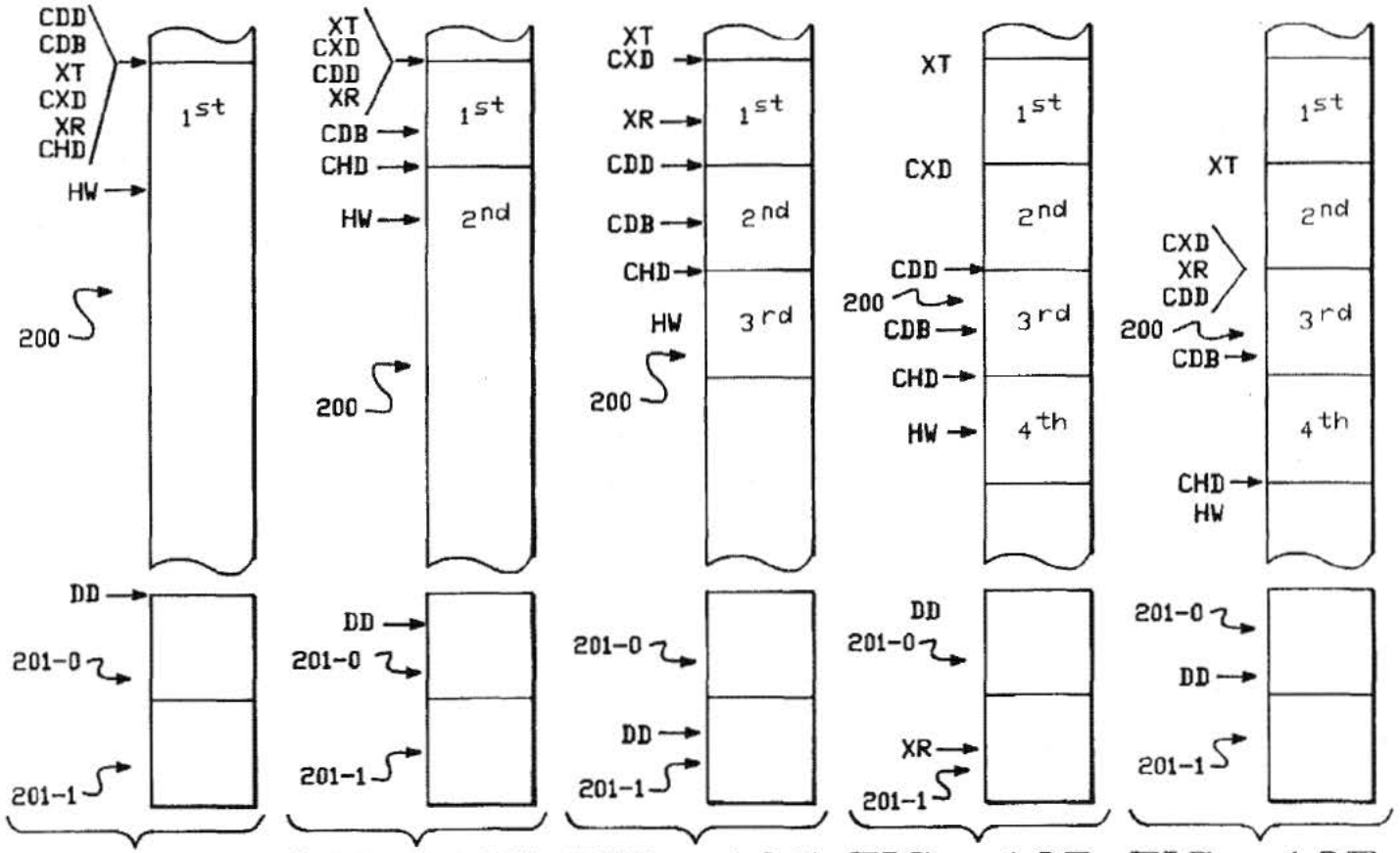


FIG.—10A FIG.—10B FIG.—10C FIG.—10D FIG.—10E

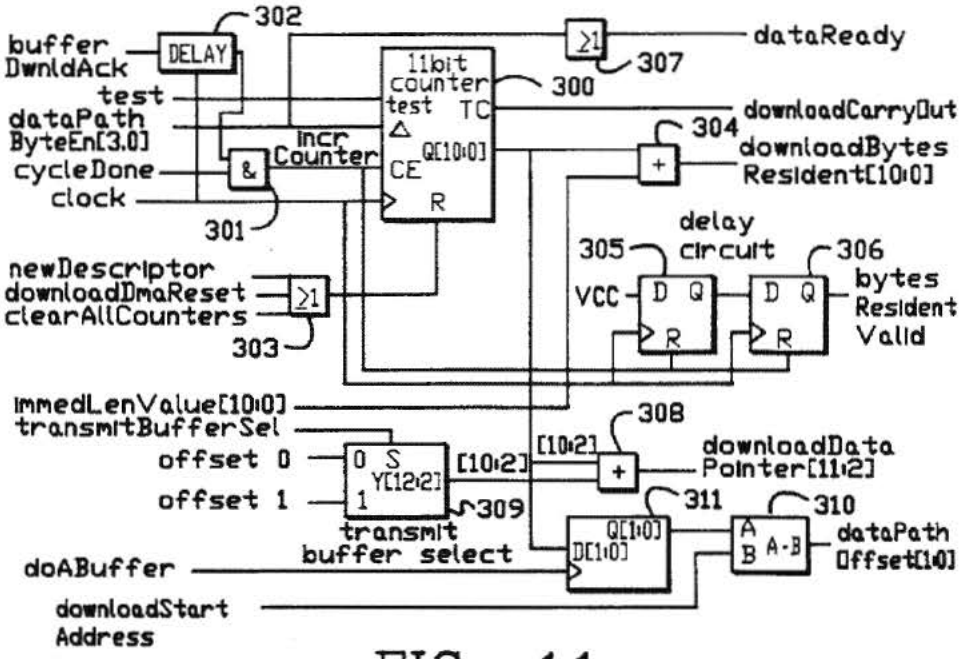

302
buffer DwnldAck
DELAY
test
dataPath ByteEn[3:0]
cycleDone
clock
&
301
incr Counter
11bit counter
test TC
CE Q[10:0]
R
≥1
307
dataReady
300
downloadCarryOut
304
+
downloadBytes Resident[10:0]
delay circuit
305
306
VCC
D Q
R
bytes Resident Valid
newDescriptor
downloadDmaReset
clearAllCounters
≥1
303
immedLenValue[10:0]
transmitBufferSel
offset 0
offset 1
0 S
1 Y[12:2]
[10:2]
309
transmit buffer select
308
[10:2]
+
downloadData Pointer[11:2]
311
310
Q[1:0]
D[1:0]
A
B A-B
dataPath Offset[1:0]
doABuffer
downloadStart Address

FIG.—11

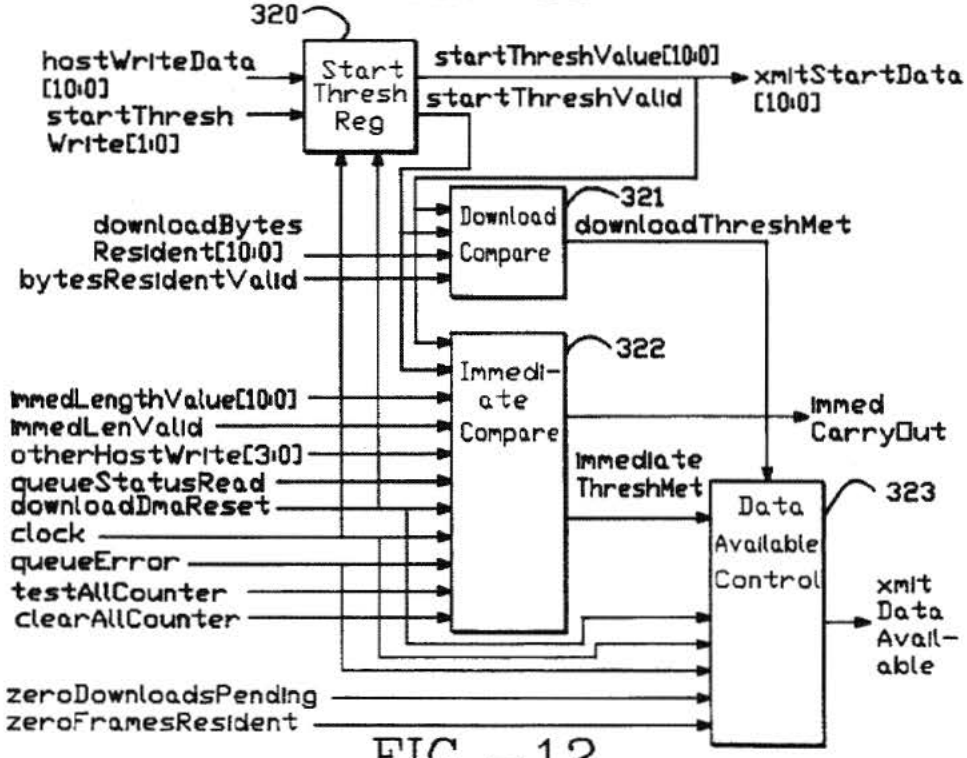

320
hostWriteData [10:0]
startThresh Write[1:0]
Start Thresh Reg
startThreshValue[10:0]
startThreshValid
xmitStartData [10:0]
downloadBytes Resident[10:0]
bytesResidentValid
Download Compare
321
downloadThreshMet
322
Immedi-ate Compare
immedLengthValue[10:0]
immedLenValid
otherHostWrite[3:0]
queueStatusRead
downloadDmaReset
clock
queueError
testAllCounter
clearAllCounter
immed CarryOut
immediate ThreshMet
Data Available Control
323
xmit Data Avail-able
zeroDownloadsPending
zeroFramesResident

FIG.—12

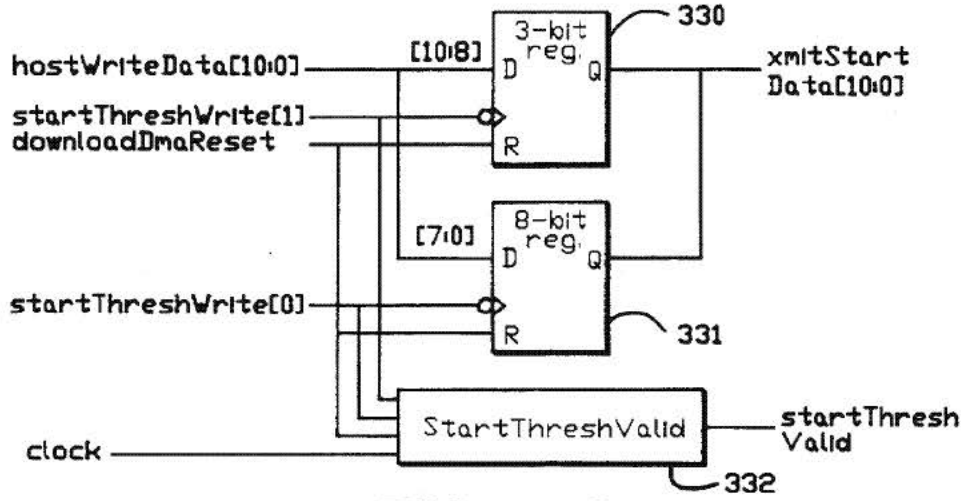
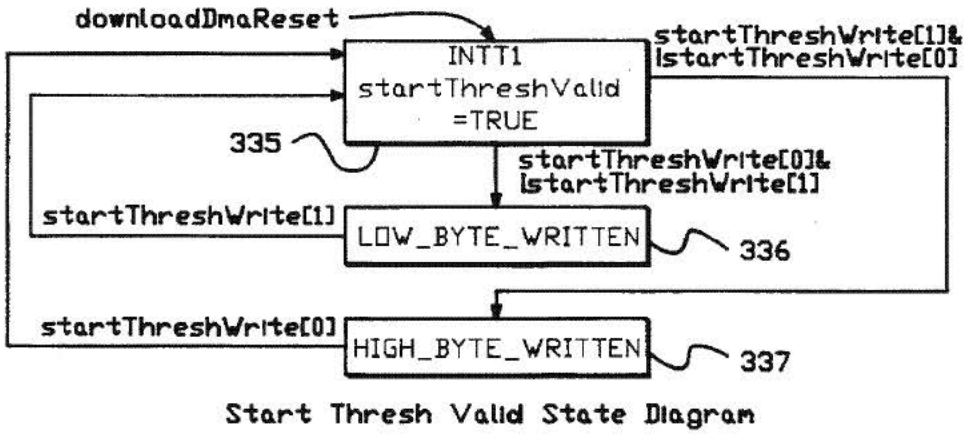


FIG.—13



Start Thresh Valid State Diagram

FIG.—14

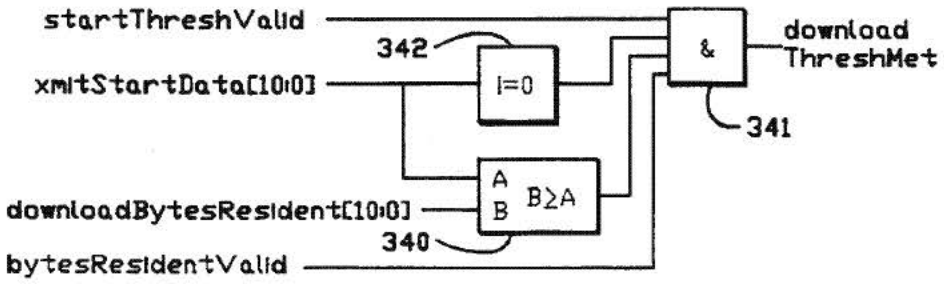


FIG.—15

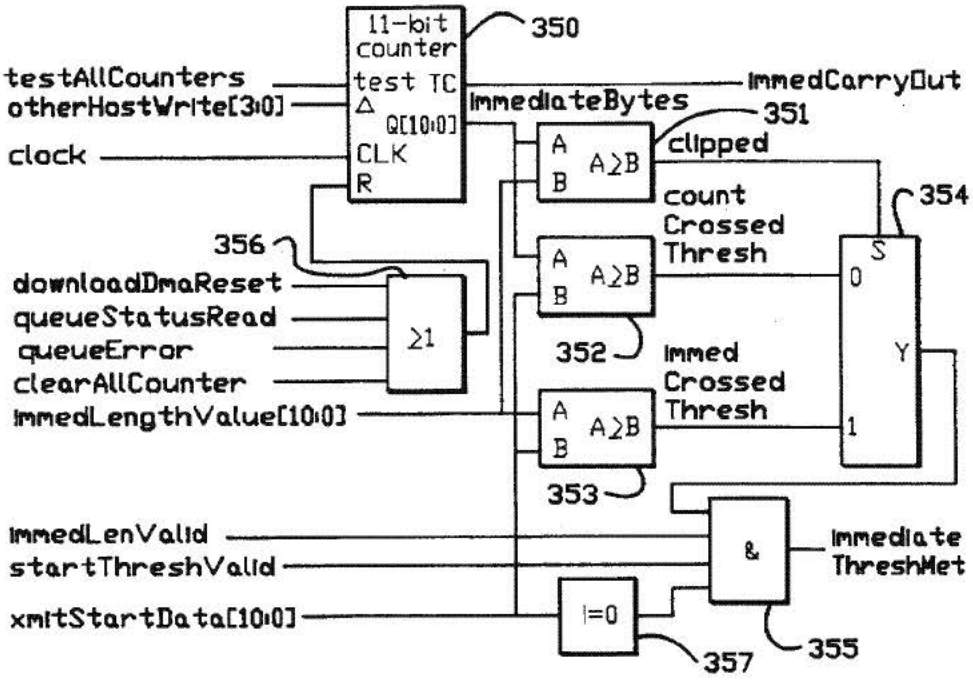

11-bit counter
350
testAllCounters
otherHostWrite[3:0]
clock
test TC
Δ Q[10:0]
CLK
R
immedCarryOut
immediateBytes
351
clipped
A
B
A≥B
count Crossed Thresh
354
S
0
Y
1
356
downloadDmaReset
queueStatusRead
queueError
clearAllCounter
≥1
352
Immed Crossed Thresh
immedLengthValue[10:0]
353
immedLenValid
startThreshValid
&
immediate ThreshMet
xmitStartData[10:0]
!=0
357
355

FIG.—16

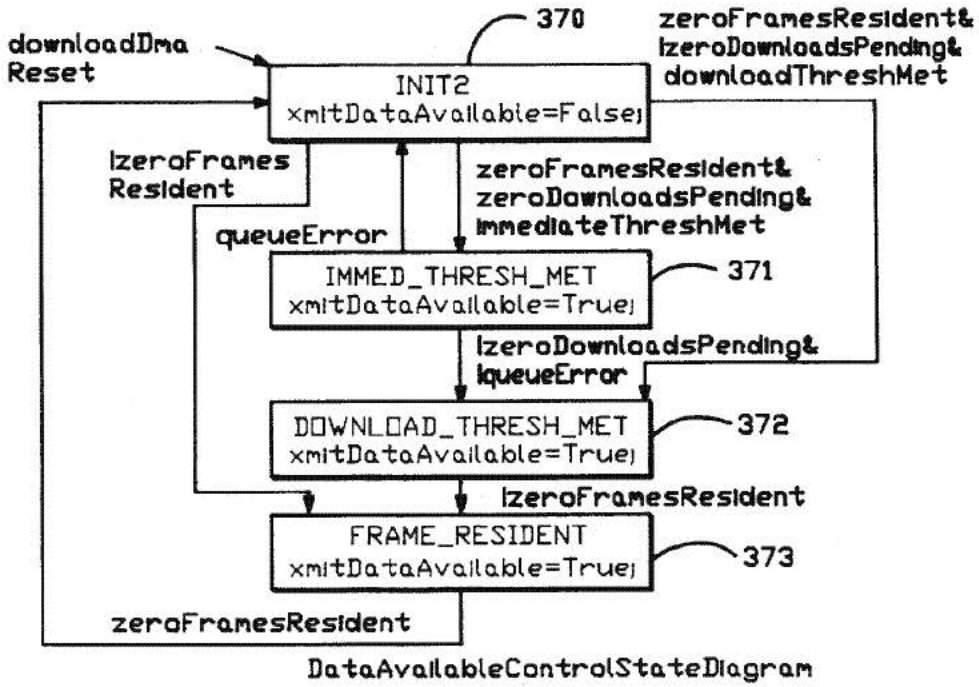

downloadDma Reset
370
zeroFramesResident& !zeroDownloadsPending& downloadThreshMet
INIT2
xmitDataAvailable=False)
!zeroFrames Resident
zeroFramesResident& zeroDownloadsPending& immediateThreshMet
queueError
IMMED_THRESH_MET
xmitDataAvailable=True)
371
!zeroDownloadsPending& !queueError
DOWNLOAD_THRESH_MET
xmitDataAvailable=True)
372
!zeroFramesResident
FRAME_RESIDENT
xmitDataAvailable=True)
373
zeroFramesResident
DataAvailableControlStateDiagram

FIG.—17

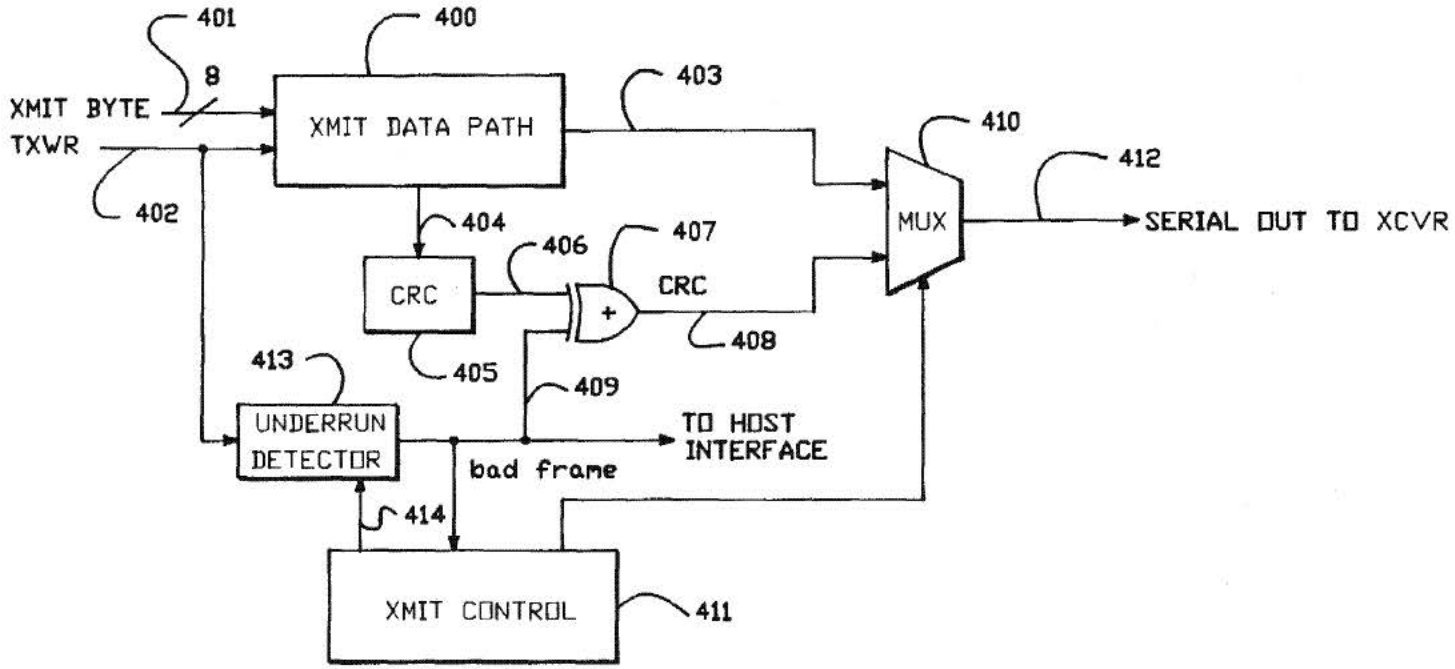

401
400
8
XMIT BYTE
TXWR
XMIT DATA PATH
403
410
412
402
404
406
407
MUX
SERIAL OUT TO XCVR
CRC
CRC
+
408
405
413
409
UNDERRUN
DETECTOR
TO HOST
INTERFACE
bad frame
414
XMIT CONTROL
411

FIG.—18

METHOD FOR AUTOMATIC INITIATION OF DATA TRANSMISSION

This application is a continuation of application Ser. No. 08/397,997, filed Mar. 3, 1995, now abandoned; which is a continuation of parent application Ser. No. 07/920,893, filed Jul. 28, 1992, now U.S. Pat. No. 5,434,872.

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application is related to copending U.S. patent application entitled NETWORK INTERFACE WITH HOST INDEPENDENT BUFFER MANAGEMENT, application Ser. No. 07/921,519, filed 28, Jul. 1992, now U.S. Pat. No. 5,299,313, which was owned at the time of invention and is currently owned by the same assignee.

1. Field of the Invention

The present invention relates to interfaces between communications networks and data processing systems; in particular, systems which involve the transmission of packets or frames of data across a communications network.

2. Description of the Related Art

Data communication systems are often based on the transmission of packets or frames of data that are composed by a sender. The packets or frames of data are designed to be compatible with the network protocol involved with the communication system. Thus, the sending system must compose the frames of data according to the network protocol prior to initiation of transmission of data. Often, a sending system will wait for acknowledgement that a frame of data sent to a network adapter has been transmitted prior to performing a subsequent task, such as composing a second frame of data to be transmitted.

Some network adapter interfaces include dedicated transmit buffers into which a frame of data composed by the sending system can be downloaded by the sending system. The frame is then stored in the transmit data buffer until the media access control functions associated with transmitting the frame on the network have successfully transmitted the frame, or cancelled the frame transmission. If the frame transmission is cancelled, the data may be retained in the transmit data buffer until the sending system initiates a second attempt to transmit the frame. Transmit data buffers are to be distinguished from first-in-first-out FIFO systems, in which the sending system downloads data of a frame into the FIFO, while the network adapter unloads the FIFO during a transmission. The data in FIFOs cannot be retained and reused by the media access control functions, or by the host, like data in transmit data buffers.

Although transmit data buffers enable a sending system to compose and download a frame into the transmit data buffer, and then attend to other tasks while the network adapter attempts to transmit the frame, it suffers the disadvantage that transmission of a frame is delayed until the entire frame has been downloaded into the buffer. Thus, transmit data buffer type systems improve host system efficiency at the expense of network throughput. Operations which are communication intensive suffer a performance downgrade.

Furthermore, the prior art systems which use transmit data buffers require the host or sending system to manage the transmit data buffer. A network interface controller transfers data from the host managed transmit data buffer using DMA techniques through a FIFO buffer in the interlace controller and on to the network.

Representative prior art systems include the National Semiconductor DP83932B, a systems-oriented network interface controller (SONIC) and the Intel 82586 local area network coprocessor.

It is desirable to provide the advantages of a transmit data buffer, while maintaining the communications throughput available from the simpler FIFO based systems.

SUMMARY OF THE INVENTION

The system and method of the present invention provides for the early initiation of transmission of data in a network interface that includes a dedicated transmit buffer. The system includes logic for transferring frames of data composed by the host computer into the transmit buffer. Also, the amount of data of a frame which is downloaded by the host to the transmit buffer is monitored to make a threshold determination of an amount of data of the frame resident in the transmit buffer. The network interface controller includes logic for initiating transmission of the frame when the threshold determination indicates that a sufficient portion of the frame is resident in the transmit buffer, and prior to the transfer of all of the data of the frame into the transmit buffer. In one aspect of the invention, the monitoring logic includes a threshold store, which is programmable by the host computer for storing a threshold value and logic for posting status information to the host. Thus, the threshold value may be set by the host system to optimize performance using the alterable threshold store and the posted status information.

According to another aspect of the present invention, the transmit buffer includes a transmit descriptor ring, and a transmit data buffer. The host system composes a frame by storing a transmit descriptor in the adapter managed transmit descriptor ring. The transmit descriptor may remain resident in the transmit descriptor ring for some time prior to an initiation of the transmission of the data by the adapter, because of other transmit descriptors being processed ahead of a current descriptor, or other reasons. When the adapter begins processing of a transmit descriptor, it retrieves immediate data from the descriptor itself, and begins a download process into the transmit data buffer of data identified in the descriptor. The threshold logic determines the amount of immediate data from the descriptor, and monitors the downloading of data of the frame into the download area. When the combination meets the threshold, then actual transmission of the frame is initiated. Thus, transmission of a frame may be initiated before the complete frame has been downloaded into the download area.

Accordingly, the present invention provides an architecture for a network interface controller which greatly increases the flexibility for host systems in composing and transmitting frames of data, while maintaining communications throughput for applications that are communications intensive.

Other aspects and advantages of the present invention can be seen upon review of the figures, the detailed description, and the claims which follow.

BRIEF DESCRIPTION OF THE FIGURES

The system and method of the present invention may be better understood with the aid of the following text and accompanying drawings, FIGS. 1–18.

FIG. 1 is a simplified block diagram of a system with a network interface according to the present invention.

FIG. 2 is a schematic functional block diagram of the early transmit system according to the present invention.

FIG. 3 is a block diagram of a preferred implementation of the host system and network interface according to the present invention.

FIGS. **4** and **4A** are block diagrams of the network interface processor of FIG. **3** and the structure of the autonomous access address buses, respectively, implementing the present invention.

FIG. **5** is a schematic diagram illustrating data flow from the host memory space through adapter memory to the network according to the present invention.

FIG. **6** is a map of the host system address space used for any transmission of data according to the present invention.

FIG. **7** is a memory map of the adapter memory independent of the host system address space.

FIG. **8** illustrates the transcript descriptor data structure according to one aspect of the present invention.

FIG. **9** illustrates the management of the transmit descriptor ring buffer and transmit data buffer, and pointers used during the transmit operation according to the present invention.

FIGS. **10A–10E** are a schematic illustration of the management of the pointers for the transmit descriptor ring buffer and transmit data buffer.

FIG. **11** is a logic diagram illustrating the data path arithmetic used in the transmit function for the network interface controller of FIG. **4**.

FIG. **12** is a logic diagram of the transmit start control logic for the network interface adapter of FIG. **4**.

FIG. **13** is a logic diagram of the transmit start threshold register for the logic of FIG. **12**.

FIG. **14** is a state diagram for the transmit start threshold register of FIG. **13**.

FIG. **15** is a logic diagram of the download compare logic of FIG. **12**.

FIG. **16** is a logic diagram of the immediate data comparator of FIG. **12**.

FIG. **17** is a state diagram for the data available control function in the logic of FIG. **12**.

FIG. **18** is a simplified block diagram of the transmit logic according to the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

A detailed description of preferred embodiments of the present invention is provided with respect to the figures. FIGS. **1** and **2** provide an overview of the data communications systems according to the present invention. FIGS. **3** through **10A–10E** provide a detailed description of a preferred implementation of the present invention, including data structures and data flow. FIGS. **11–18** provide one implementation of logic for implementing the early transmission feature of the present invention.

I. System Overview

FIG. **1** illustrates a data communication system according to the present invention with a controller circuit using a dedicated transmit buffer memory which is automatically enabled to begin transmission of a frame on the network when the number of bytes available in the transmit buffer memory exceeds a preprogrammed threshold. As shown in FIG. **1**, such system for communicating data includes a host data processing system, generally referred to by reference number **1**, which includes a host system bus **2**, a host central processing unit **3**, host memory **4**, and other host devices **5**, all communicating across the bus **2**. A network interface adapter **6** is coupled to the host bus **2**. The adapter **6** is coupled to a transceiver **7** which is, in turn, connected to a network medium **8**, such as coaxial cable, a twisted pair conductor, a fiber optic, satellite, wireless, or other communication medium. The network adapter **6** is, in turn, connected to an adapter memory **9**, which is managed by the interface controller **6** or by the host CPU **3**, depending on a particular implementation provided. The adapter **6** also includes early transmit logic **6A** for monitoring the transfer of data from the host system **1** into the adapter memory **9**. The early transmit logic **6A** makes a threshold determination for a given frame being transferred from the host **1** into the adapter memory **9**. When the threshold is met, then the adapter **6** begins transmitting the frame across the network. Of course, a wide variety of other configurations of these components could be implemented. For instance, the adapter memory **9** may be connected directly to the host bus **2**. Also, multiple bus configurations might be utilized.

FIG. **2** illustrates the functional components of the early transmit system according to the present invention. In FIG. **2**, a number of functional blocks are represented. The host computer or sending system **30** communicates with a host interface logic **31**, receiving data as indicated by line **32** and control signals as indicated by line **33**. The host interface logic **31** supplies data to a transmit buffer **34** across line **35**. Also, threshold logic **36** is coupled with the host interface logic **31** to monitor the transfer of data from the host computer **30** into the frame buffer **34**. The line **37** indicates the coupling of the threshold logic **36** to the host interface logic **31**. The threshold logic makes the threshold determination and generates a signal as indicated by line **38** to transmit logic **39** including, for instance, media access control MAC logic. The transmit logic **39** receives data from the buffer **34** across lines **40** and supplies that data as indicated by line **41** to the network medium **42**. Coupled with the threshold logic **36** is a threshold store **43** which stores a threshold value which indicates an amount of data of a frame that must be resident in the frame buffer **34** before transmission of that frame may be initiated by the transmit DMA logic and MAC **39**.

The threshold store **43**, in a preferred system, is dynamically programmable by the host computer **30**. In this embodiment, the threshold store **43** is a register accessible by the host through the interface logic **31**. Alternatively, the threshold store may be a read only memory set during manufacture. In yet other alternatives, the threshold store may be implemented using user specified data in non-volatile memory, such as EEPROMs, FLASH EPROMs, or other memory storage devices.

The transmit logic **39** also supplies status information across line **44** to the host interface logic **31**, for posting to the host system. The status information includes indications of underrun conditions and may be used by the host to optimize the value in the threshold store **43**.

In operation, the host computer composes a frame of data to be transmitted on the network medium **42**. The host computer **30** then identifies that frame through the host interface **31**. The host interface coupled with the identifiers of the frame move data from the host computer **30** into the buffer **34** according to the description of the frame. The threshold logic **36** monitors the transfer of data into the buffer **34**. When the threshold amount of data is resident in the buffer **34**, then the transmit logic **39** is instructed to begin transmission of the frame. The transmit logic **39** then begins retrieving data from the buffer **34** to support transmission of the frame on the medium **42**. This operation begins before the entire frame has been transferred from the host computer **30** into the buffer **34**, if the transmit logic **39** is available to transmit the frame subject of the ongoing download from the host computer **30**, the frame being downloaded into the

buffer **34** is larger than the threshold set by the threshold store **43**, and the host computer **30** indicates that immediate transmission of the data is desired.

As mentioned above, the buffer **34** may be directly addressable by the host computer **30**, or in preferred systems is addressable through the host interface **31** which consists of a prespecified area of host address space into which the host computer **30** writes data and control signals associated with a transmission operation. The host interface **31** then maps the transfer of data from that prespecified address space into the buffer **34** and host accessible registers independent of the host.

A more detailed description of a preferred embodiment of the present invention is illustrated in FIG. **3**.

II. System Overview

FIG. **3** is a schematic diagram of a computer system including the network interface controller according to the present invention. The computer system includes a host system, including a host processor **10**, host memory **11**, and other host devices **12**, all communicating through a host system bus **13**, such as an EISA bus. The host system bus **13** includes address lines which define a host system address space. Typically, for an EISA bus, there are 32 address lines establishing a host system address space of about 4 Gigabytes.

The network interlace controller includes a network interface processor **14**, implemented in one preferred system as an application specific integrated circuit designed to implement the functions outlined below using VERILOG design tools as known in the art (available from Cadence, Inc., San Jose, Calif.). The network interface processor **14** is coupled through appropriate buffers to the bus **13**. The network interface processor **14** is also coupled to random access memory **15**, BIOS ROM **16**, and INFO EEPROM **17**, a test port **18**, an encode/decode chip **19**, and a network transceiver **20**. The network transceiver **20** is, in turn, coupled to a network medium.

A majority of the functionality is embodied in the network interface processor **14**. In the preferred embodiment, all registers that are accessible across the bus **13** by the host system reside either in the processor **14**, or in the RAM **15**. If resident in the RAM **15**, their access is managed by the network interface processor **14**.

The RAM **15** is a primary resource on the network interface controller. This resource provides buffer memory outside the host address space used in the transmit and receive operations of the network interface. Details concerning the organization and utilization of this RAM **15** are described below.

The BIOS ROM **16** provides extension to the host system's basic input/output code through the network interface processor **14** during initialization. The addresses for the BIOS ROM **16** and the data from the BIOS ROM **16** are coupled to the network interface processor **14** across buses **21** and **22**, respectively, which are also shared by the RAM **15**.

The INFO EEPROM **17** stores critical adapter specific data used by drivers, diagnostics, and network management software. This data is stored during the manufacturing process. During initialization of the interface controller, the contents of the EEPROM **17** are loaded into a prespecified area of the RAM **15** for use during operation.

Coupled to the interface processor **14** is an encode/decode chip **19**, such as the National Semiconductor 8391 Manchester encode/decode chip. The signals coupled to the AUI connector are provided to allow use of a transceiver external to the board.

The transceiver **20** in a preferred system comprises either a thin Ethernet (coax/BNC) transceiver or a 10 BaseT (Type 3/RJ-45) transceiver. Control signals for the transceiver **20** are produced on the network interface controller **14**, using the conversion logic on the encode/decode chip **19**.

A test port **18** is provided in a preferred system for use during manufacture and testing.

III. Controller Functional Units

FIG. **4** provides a block diagram of the network interface processor **14** of FIG. **3**, including functional blocks and data paths. There are numerous connections not shown having to do with the control of the various data flow paths. The interfaces illustrated include a RAM interface **50**, a host bus interface **51**, and a transceiver interface **52**. The bus interface **51** is implemented for an EISA bus, and operates at times either as a master or as a slave on the bus. Each of the functional units in the implementation shown in FIG. **4** is described below.

A. EISA Slave Interface **54**

The EISA slave interface **54** provides a path for the EISA host to access the registers and buffers managed by the network interface controller. The module contains configuration registers for the controller, and performs crude decoding of the EISA bus for the purpose of routing signals. The EISA slave interface **54** does not interpret any of the addressing of individual registers distributed throughout the controller.

In operation, the EISA slave interface continuously monitors the EISA address bus and determines when the configuration registers, memory mapped network registers, or BIOS ROM of the adapter are being accessed.

In addition, for every memory slave cycle initiated by the EISA bus, the EISA slave interface will post a cycle request to the cycle arbiter **56**. The cycle arbiter imposes wait states upon the host system until the request has been granted.

The EISA slave interface also provides a generic 32 bit bus interface to the remainder of the network controller. The generic nature of the interface allows for easy adaptation of the design to other bus types, such as the microchannel, without requiring redesign of the remainder of the chip.

Bits **14-2** of the EISA address bus are latched and pass through to other modules. The least significant two bits (**1**, **0**) of the address are represented by 4 byte enables that are also valid throughout a data transfer cycle. Four 8-bit byte lanes make up the slave data channel. The data bus is actually a pair of unidirectional buses, one for writes and one for reads in a preferred system. The data write bus is wired in a multi-drop fashion to all modules that require connection to the EISA data bus through the slave interface. The read bus is multiplexed and masked in the RAM interface module **50**. Write requests by the EISA bus can be held until they are acknowledged by the cycle arbiter **56**. When a cycle is held, the EISA bus may be released from wait states while the cycle completes on the adapter. If a second cycle is generated by the EISA bus while the first one is still pending, then the EISA bus will be held off with wait states until the pending write is performed. In this specific embodiment, pipelining of EISA reads is not supported.

The EISA slave interface also provides an interface to the EEPROM **17**. This interface operates to transfer the contents of the EEPROM **17** into the adapter memory after reset.

There are numerous registers in the EISA slave interface module **54**, primarily related to configuration of the adapter that conform to the EISA bus specification. These registers do such things as set up the adapter's memory base address, the interrupt level, the transceiver type selection, and the BIOS ROM enable. The configuration registers also provide

the host with a positive means of identifying the adapter type and to globally disable the adapter.

B. EISA Master Interface **55**

The EISA master interface **55** handles requests from the upload DMA **57** and download DMA **58** for performing bus master operations across the EISA bus. The EISA master interface **55** autonomously arbitrates between pending upload and download requests, because of the EISA bus disallowing mixed reads and writes while performing burst transfers, used by the DMA operations of the preferred embodiment

The bus master transfers are always initiated by either the upload DMA **57** or the download DMA **58**. The transfers may be terminated by either the DMA modules upon completion of a transfer, or by the EISA master interface upon preemption by another arbitrary device on the EISA bus.

Thus, the function of the EISA master interface **55** is to arbitrate for access to the EISA bus when transfer requests are pending from either or both of the upload DMA **57** and the download DMA **58**. The EISA master interface **55** performs the signalling necessary to establish first transfers with address slaves on the bus. It also ensures compliance with the EISA bus definitions.

This module also converts real mode segment:offset addresses to 20 bit linear addresses when enabled by the adapter mode logic **59**;

C. Master/Slave Union Module **53**

The master/slave union module **53** provides for sharing of connections to the EISA bus by the EISA master interface **55** and the EISA slave interface **54**. This union module **53** consists primarily of a series of 2:1 multiplexers.

D. Interrupt Controller Module **60**

The controller also includes an interrupt controller module **60**. The interrupt controller module **60** implements various interrupt and indication functions, including masking and enable/disable mechanisms. Interrupt signals are generated by various modules within the controller, and are routed to the interrupt controller module **60**. The interrupt controller module **60** then passes the interrupt signals through various enables and masks before OR-ing them together and driving the result onto the host bus.

The interrupt controller module **60** does not detect interrupt worthy events or acknowledge the interrupts passed to the host it includes a number of adapter resident registers utilized in the interrupt handling functions.

E. Adapter Mode Module **59**

The adapter mode module **59** provides a number of functions including setting various basic operating modes of the controller, and reporting status of various conditions of the controller. The adapter module **59** also establishes the base address of a window register used for diagnostics by the host system. Furthermore, the adapter mode module **59** generates reset functions for the adapter. Also, this module provides the MAC ID register which identifies the media access controller implemented by the device, for communication to various modules within the controller and to the host system.

F. Cycle Arbiter Module **56**

The cycle arbiter module **56** is responsible for distributing access to the adapter's RAM resident and ASIC resident registers through the RAM interface **50** among various requestors. It functions to allow-timely access to the RAM by modules that are most in danger of suffering an overrun or underrun condition in response to a priority scheme.

G. Multicast Comparator Module **61**

The controller illustrated in FIG. **2** also includes a multicast comparator module **61**. When enabled by the adapter mode module **59**, the Multicast comparator module **61** performs a bit by bit comparison of a received frame's destination address field with contents of the multicast address table. The multicast address table is established by the host and stored in RAM **15**. A mismatch during this compare, coupled with neither an indivdual address nor a broadcast address match, will result in the rejection of an incoming frame.

Thus, the multicast comparator module **61** monitors the activity of the Ethernet receiver module **62** and the receive DMA module **63** to determine when a new frame is being received. Each byte that is received by the Ethernet receiver **62** and presented at the parallel interface **64** of the receiver, is shadowed by the multicast comparator module **61**. These bytes are then compared against valid entries in a multicast address table accessible by the multicast comparator **61**.

The multicast comparator **61** does not establish or maintain the contents of the multicast address table. However, the module dotects the host accesses to the table and supplies appropriate redirection offsets to the RAM interface module **50**.

H. Statistics Controller Module **65**

The preferred system also includes a statistics controller module **65**. This module monitors activity of various other modules within the controller, most particularly the Ethernet transmitter module **66** and the Ethernet receiver module **62**, and updates statistics maintained in RAM **15** as applicable events occur.

I. Download DMA Module **58**

The download DMA module **58** is responsible for issuing requests for bus master downloads of data from the host system to the adapter memory. This data is then deposited within the adapter's onboard transmit data buffer for either immediate or future transmission.

As soon as buffer descriptors are defined and one of the transmit data buffers becomes available, as described below, the download DMA module submits requests for download bus master operations to the EISA master interface **55**. The download DMA module **58** performs byte alignment, including any required packing and unpacking to align the data as implied by the respective starting addresses of the host and the adapter.

The download DMA module **58** also includes logic for maintaining the transmit descriptor ring buffer within the adapter's RAM. The download DMA module **58** generates an interrupt in the appropriate mode to indicate completion of the download operation. Also, the download DMA module **58** informs the transmit DMA module **67** when it is time to begin transmission. Various registers involved in the download DMA operation are described in more detail below as they are involved directly in the data buffering process of the present invention.

J. Transmit DMA Module **67**

The transmit DMA module **67** operates to fetch bytes from the transmit descriptor buffer, the transmit data buffer, or both, as described below, when directed to do so by the download DMA logic **58**. The fetched bytes are then presented in sequence to the Ethernet transmitter module **66**.

Thus, the transmit DMA module **67** functions to read the contents of the transmit descriptor ring buffer to determine the extent of immediate data, and the length of the overall frame to be transmitted. If a frame specified does not match the specifications for the network, for instance, is shorter than the 802.3 minimum, and the adapter is so enabled, this module will supply additional bytes of undefined data to the Ethernet transmitter module **66** as padding.

Collision retries in the CSMA/CD network are handled by the transmit DMA module **67**. When a collision is indicated

by the Ethernet transmitter **66**, the transmit DMA module **67** will resubmit the same frame by reinterpreting the frame descriptor in the buffer.

If enabled, and when the transmit complete condition is satisfied, a transmit complete interrupt is generated for handling by the interrupt controller **60** in the transmit DMA module **67**.

The transmit DMA module **67** also stores appropriate frame status in the transmit descriptor ring buffer of the adapter upon completion of transmission.

The transmit DMA module **67** also detects underrun conditions, when a shortfall of data available for transmission occurs.

Again, registers involved in operation of the transmit DMA module **67** are described in detail below.

K. Ethernet Transmitter Module **66**

The Ethernet transmitter module **66** provides media access control function associated with transmission for an 802.3 standard network. This module accepts parallel data bytes from the transmit DMA module **67** and applies the 802.3 access rules, and supplies serial data to the external encoder/decoder chip.

L. Ethernet Receiver Module **62**

Similarly, the Ethernet receiver module **62** is an essential 802.3 implementation. This module accepts serial data from an external encoder/decoder, applies the 802.3 rules to the data and presents the data in parallel form for use by the receive DMA module **63**. Thus, the Ethernet transmitter **66** and Ethernet receivers perform the standard CSMA/CD functions.

M. Receive DMA Module **63**

The receive DMA module **63** is a functional complement to the transmit DMA module **67**. This module is responsible for accepting parallel data bytes on the Ethernet receiver **62** and depositing them in the adapter's receive ring buffer.

The receive DMA module **63** is responsible for assembling bytes from the Ethernet receiver into 32 bit words prior to storage in the adapter's RAM. At completion of frame reception, the frame's status and length are deposited within the receive ring buffer for use by the host system.

The receive DMA module **63** is also responsible for establishing and maintaining of the receive buffer ring within the RAM of the adapter as described in detail below. Furthermore, the positioning of the "LOOKBUF" register allowing the host to view received data, as mentioned below, is handled by the receive DMA module **63**.

The receive DMA module **63** also handles interrupt indications under appropriate conditions.

N. Upload DMA Module **57**

The upload DMA module **57** performs data transfers from the receive buffer through the RAM interface **50** to the host system. Thus, the receive ring buffer is managed by the receive DMA module **63** and interpreted by the upload DMA **57**. Actual bus master transfers are carried out by the EISA master interface module **55**.

The upload DMA module **67** interprets data structures deposited in the receive ring buffer by the receive DMA module **63** including the location and length of a received frame. It also reads the transfer descriptor which is defined by the host system to determine how many bytes of the frame to transfer, and where in the host memory to transfer the frame.

The upload DMA module **57** requests bus master cycles from the EISA master interface module **55** in response to valid transfer requests deposited by the host.

The upload DMA module **57** also utilizes an interlock between the receive DMA module **63** and the upload DMA module **57**, to reduce the transfer rate to the host system to prevent "getting ahead" of the frame being received through the receive DMA module **63**. Finally, this module generates interrupts indicating completion of a transfer for use by the host.

O. RAM Interface Module **50**

The RAM interface module **50** provides multiplexers and masks for various functions involved in addressing the RAM. The module multiplexes the various address and data sources together to form parameters for RAM access cycles. This module **50** is responsible for gathering up data from the various other modules on the controller that can be read by the host system. Additionally, this module applies a mask to the data to force unused upper bits to a zero, and latches data words for multicycle reads.

P. JTAG Module

Also included in the controller, though not shown, is a JTAG module which implements a state machine as specified in IEEE standard 1149.1-1990, May 21, 1990. This module provides a scan test of the ASICs pins for use during manufacture.

Q. The Autonomous Access Address Bus Definition

The multi-cast comparator **61** statistics controller **65**, download DMA **58**, transmit DMA **67**, receive DMA **63**, EISA slave interface **54**, and upload DMA **57** all require autonomous access capabilities through the RAM interface **50** to the adapter's RAM. The autonomous accesses are those that occur independently of any cycles that may be occurring on the host bus.

This requirement requires the capability to generate and communicate address information for use by the RAM interface **50**. In the case for the receive DMA **63**, transmit DMA **67**, and download DMA **58**, the capability is implied by the DMA logic which drives offset buses. Similar offset buses dedicated to the autonomous accesses are provided for use by the statistics controller **65**, upload DMA **57**, and multi-cast comparator **61**.

FIG. **4A** schematically defines the structure of the autonomous access address buses. The various functional modules are given like reference numbers, as provided in FIG. **4**.

The multi-cast comparator **61** drives the multi-cast offset bus [**6:2**] to supply a variable offset to the RAM interface **50** for addressing individual locations within the multi-cast address table. A valid value is presented during the associated autonomous access cycle.

The statistics controller **65** drives the statistics offset bus [**7:2**] to supply a variable offset to the RAM interface **50** for addressing individual locations within statistics region of the adapter's RAM. A valid value is presented during the associated autonomous access cycle.

The upload DMA **57** drives the upload DMA offset bus [**14:2**] to supply a variable offset to the RAM interface **50** for addressing individual locations within a receive buffer ring. A valid value is presented during the associated autonomous access cycle.

The download DMA **58** drives a download DMA offset bus [**122**] to supply a variable offset to the transmit buffer/transmit descriptor region of the RAM during autonomous cycles. Two select signals generated by the download DMA logic **58** are used to distinguish between the two RAM regions. Also, the download DMA logic **58** generates a download DMA byte enable [**3:0**] to indicate which byte lanes of the 32 bit data bus extending from download DMA **58** to the RAM interface **50** contain valid data. The valid value is presented during the associated autonomous cycle.

The receive DMA **63** drives a receive DMA offset bus [**14:2**] to convey a variable offset within the 22K byte receive buffer region of the RAM.

Finally, the transmit DMA logic drives a transmit DMA offset bus [12:2] to supply a variable offset to the transmit buffer/transmit descriptor region of the RAM during autonomous cycles. The two select signals generated by the transmit DMA **67** are used to distinguish between the two RAM regions.

IV. Transmit and Receive Data Row and Structure

FIG. **5** provides a heuristic data flow diagram of an adapter according to the present invention emphasizing the host interface, the adapter memory and the network interface data flow for transmission and reception.

As mentioned above, the host system will include a host memory space (generally **100**) defined by the addresses on the host bus. A pre-specified block **101** of the host memory space is set aside for the adapter interface addresses. The adapter includes host interface logic **102** which is responsive to accesses across the host bus within the adapter interface address block **101**. Also in the adapter is a host independent memory **103**. The host interface logic operates to map the transfer of data between the specified block **101** of addresses and the independent memory **103**. The adapter also includes network interface logic **104** which is coupled to the adapter memory **103**. The network interface logic manages transfers of data from buffers in the independent memory **103** and the network transceiver **105**. The network transceiver **105** then supplies the data onto the network medium **106**.

The host interface logic **102** includes a transmit descriptor logic and a download DMA logic (generally **107**) used in the transmit process, and view logic, transfer descriptor logic, and upload DMA logic (generally **108**) used in the receive process. These modules basically manage communication of data between the independent memory **103** and the host in response to reads/writes by the host system to the adapter interface address block **101**. This relieves the host of any address translations or buffer management functions for the transmit and receive operations.

The network interface logic **104** includes transmit DMA logic, (generally **109**) and receive DMA logic (generally **110**). The transmit DMA logic **109** is responsive to descriptors stored in the adapter memory **103**, as described below, for moving data out of the independent adapter memory **103** to the network transceiver **105**. Similarly, the receive DMA logic **110** is responsible for moving data from the transceiver **105** into the independent adapter memory **103**. Thus, all communications of data from the network medium **106** are coupled directly into host independent memory **103**. Communications from the host independent memory **103** are then controlled through the host interface logic **102** in response to a memory mapped region in the host memory space, greatly simplifying the protocol software necessary to communicate with the network.

FIG. **6** provides a simplified map of the host address block **101** used by the host interface. The addresses within this block appear to the host like memory mapped registers in a continuous 8K block of the host address space in a preferred system.

For the EISA embodiment, "registers", or mapped areas, in the block **101** are located on double word address boundaries, thus, addresses. are a multiple of four. Many of the "registers" occupy several double words (as many as 509) of memory space.

Although the "registers" are memory mapped to an arbitrary prespecified block of host address space, none of the reads or writes performed by the host system to these registers actually directly access the adapter memory. Rather, the accesses to the memory mapped space are interpreted by the host interface logic **104** transparent to the host system. Thus, the memory in the adapter is independent of the host address space and of host management. FIG. **6** provides an overview mapping of the host address space used for accessing these registers. The registers include primarily a transmit area register (XMIT AREA) at offset 0010 (hex), a transfer area register (XFER AREA) at offset 0800 (hex), and a look buffer (LOOKBUF) at offset 100 C (hex). Vadous status, statistics, information, and indication registers are distributed throughout the balance of the area.

The XMIT AREA register is used by the host to write transmit descriptors into the adapter. The transmit descriptors are described in more detail below, but include data that identifies data to be compiled and transmitted as a frame, and may include immediate data. The XMIT AREA at offset 0010 (hex) is approximately 2K bytes in size. This data is mapped into a transmit descriptor ring in the independent adapter memory as described below.

The XFER AREA at offset 0800 (hex) in the adapter interface host address block is a buffer of approximately 1K byte through which transfer descriptors are written into the independent memory of the adapter. The LOOKBUF area at offset 100 C (hex) is a buffer of approximately 2K bytes providing a read only window into a receive ring buffer within the host independent adapter memory. More details of the receiver process are provided in the above cross-referenced application entitled NETWORK INTERFACE WITH HOST INDEPENDENT BUFFER MANAGEMENT.

FIG. **7** provides a map of the host independent memory on the adapter. This memory is organized into a transmit data buffer at offset 0 of approximately 3K bytes, a transmit descriptor ring at offset 0C00 (hex) of approximately 5K bytes, a receive buffer ring at offset 2000 (hex) of approximately 22K bytes, and a transfer descriptor area at offset 7800 (hex) of approximately 1K bytes is provided in the independent memory. The last three areas of memory include adapter information, network statistics, and multicast address tables for use by the adapter.

In the preferred system, the adapter uses 32K bytes of static RAM for the transmit buffers, receive buffers, control structures, and various status and statistics registers. Several of the regions in the adapter's memory defined in FIG. **5** provide defined data structures.

A. Transmit Data Buffer

The transmit data buffer occupies 3K bytes as mentioned above. This region is divided into two 1.5K buffers. Only the data that are downloaded to the adapter via bus master transfers are stored in these buffers. The controller will use both the contents of the transmit data buffer and the immediate data portion of the transmit descriptors, when encapsulating a frame for transmission. The adapter automatically alternates the use of the buffers after choosing the buffer closest to the base of the memory as the power up default.

The transmit buffers are shared by the download DMA logic and the transmit DMA logic. The transmit DMA logic may switch from buffer **0** to buffer **1** and back again freely. The only restriction being the availability of transmit data as defined by the transmit start threshold register. The transmit DMA module switches from one buffer to the other whenever it has completed a transmission. The buffer switch occurs regardless of whether or not the transmission was successful and regardless of whether or not bus master download data were used in the preceding transmission.

The download DMA module may only switch from one buffer to the other, if the buffer it is going to switch to is not being used by the transmit DMA module. Download DMA will attempt to switch from one buffer to another every time

it completes processing of a transmit descriptor as described below, regardless of whether or not any bus master operations were called for in the preceding descriptor. However, it will not change to a buffer that is in use by the transmit DMA module.

B. Transmit Descriptors

Transmit descriptors define frames that are pending transmission, and hold the status of frames that have been transmitted. These descriptors are of variable length and are arranged in a sequential fashion around a 5K byte ring buffer as mentioned above. The first entry of the descriptor must align on a double word boundary. FIG. 8 illustrates the transmit descriptor data structure.

The bulk of the contents of an entry into the transmit descriptors region is copied verbatim from the data supplied by the host processor via the XMIT AREA illustrated in FIG. 6. However, in order to comply with the format requirements of the XMIT PROT ID and XMIT FRAME STATUS registers and to supply sufficient information for frame transmission and buffer management; one value must be relocated and several must be deposited automatically.

The host processor's writes to the XMIT AREA are offset automatically by the adapter such that the first value written—XMIT PROT ID and XMIT REQ HANDLE—end up in the fifth 32 bit word location (offset 10 (hex)) in the next available data structure in the ring. This means that the XMIT REQ HANDLE value is written by the host to the location reserved for the MACID value. Immediately after the XMIT REQ HANDLE value is written to the adapter's RAM, the adapter must copy the contents of the least significant 16 bits of the fifth 32 bit word location to the most significant 16 bits of the fourth 32 bit word location. After copying XMIT REQ HANDLE, the adapter will retrieve the MACID value from the MACID register and write that to the location vacated by XMIT REQ HANDLE. Later, after frame transmission, the least significant 16 bits of the fourth 32 bit location (offset C (hex)) will be updated with the transmit frame's status.

The NEXT DESCRIPTOR POINTER entry may be updated by the adapter any time after the XMIT BUFFER COUNT and XMIT IMMED LEN values have been written to the adapter. Because the data written to the XMIT AREA register must be written in a precise order to exact locations, the writes of these two values can be easily detected and used for the descriptor size calculations required to determine the start of the next descriptor without having to retrieve the values from RAM.

Finally, once the last XMIT DATA LEN value has been written to the adapter, the frame length can be calculated and deposited in the FRAME LENGTH position of the data structure. This value is also copied to the XMIT FRAME LENGTH register in the controller chip for immediate use by the host.

The next several paragraphs define each of the fields of the transmit descriptor data structure.

The NEXT DESCRIPTOR POINTER value points to the first word of the next descriptor. This value is updated immediately after the host reads XMIT QUEUE STATUS, NEXT DESCRIPTOR POINTER being defined does not necessarily imply that the location pointed to contains a valid descriptor. It merely indicates where the next valid descriptor may be found once it is defined.

The FRAME LENGTH field is computed and updated by the adapter, The frame length is calculated by summing all of the XMIT DATA LEN values and the XMIT IMMED LEN value. The resulting sum is the total number of bytes in the transmit frame. If the sum is less than the 802.3 minimum frame length, then the sum will be set to the minimum frame length value. The sum is written to the FRAME LENGTH line of the transmit descriptor and is also made available to the host via the XMIT FRAME LENGTH register.

The XMIT FAILURE field contains the error code that is made up of the status bits gathered from the Ethernet transmitter after the completion of transmission. This field is mapped to the XMIT FAILURE register for host access.

The XMIT REQ HANDLE value is interpreted by the transmit DMA controller to determine whether or not to generate an indication upon completion of the transmission attempt(s) for the associated frame. If the field is non-zero, an indication will be generated. Also, the frame's entry in the transmit descriptor ring will be maintained until the host has had an opportunity to examine the transmit status. The XMIT REQ HANDLE, XMIT STATUS, XMIT PROT ID, and the MACID fields are all made available to the host when an indication is generated. If XMIT REQ HANDLE is a zero, then the transmit descriptor queue entry is discarded after transmission without notifying the host in any way. Transmit underrun conditions are posted regardless of a zero XMIT REQ HANDLE.

The XMIT STATUS field contains the transmit status for the associated frame. The contents of this field are updated immediately after the transmission attempt(s). The return codes are defined in the XMIT STATUS register definition.

The XMIT PROT ID value in this field is merely maintained within the queue for use by the host upon completion of the transmission to identify the particular protocol responsible for the frame. It allows simultaneous activity of multiple protocols. Together XMIT PROT ID and XMIT REQ HANDLE uniquely identify a frame passing through the adapter.

The MACID, like XMIT PROT ID, is maintained within the queue for use upon completion of the transmission. However, the host does not write this value to the adapter via the XMIT AREA register. Rather, the host stores this value once in the MACID register and then depends upon the adapter to deposit this value into the descriptor data structure after XMIT REQ HANDLE has been copied to its final position.

The contents of the-XMIT BUFFER COUNT field are supplied by the host via a write to XMIT AREA. This field specifies the number of buffers within the host's memory that are used to make up the transmit frame. Each of the buffers is transferred in the order listed from the host's memory to the adapter's transmit data buffer as soon as one of the two transmit data buffers becomes available. If XMIT BUFFER COUNT is zero, then no bus master operations will be performed for this frame.

The XMIT IMMED LEN field, defined by a write from the host to XMIT AREA, specifies the number of "immediate" data bytes that will be supplied by the host. If this field is zero, then the next 32 bit word location will contain the first host data buffer descriptor and the entire frame will be transferred to the adapter via bus master cycles. The XMIT IMMED LEN value will not necessarily be a multiple of four. The location of the first host data buffer descriptor is determined as follows:

DESCRIPTOR OFFSET=((XMIT IMMED LEN+3) & fffc(hex))+ 18(hex).

The variable length IMMEDIATE DATA field contains the immediate data deposited to the adapter by the host using memory writes to XMIT AREA. This field may vary in length from 0 to 1,514 bytes. Immediate data is inserted into

a transmit frame between the preamble and the transmit buffer data (if any) by the adapter during transmission. Generally, immediate data is made up of the destination and source addresses and any protocol-specific header data. It is reasonable, however, for the entire transmit frame to be considered immediate data. This would make it unnecessary for the adapter to perform any bus master operations to fetch the remainder of the transmit frame. If XMIT IMMED LEN is zero, then this field is skipped and the entire frame is assumed to reside in host memory resident data buffers. If XMIT IMMED LEN does not specify an integral number of double words, then the host may round up to the nearest multiple of 4 and write up to that number of bytes. The extra bytes, beyond XMIT IMMED LEN, will be ignored and not included as part of the transmitted frame.

The XMIT DATA LEN field, one of two entries per host data buffer descriptor, defines the number of bytes in the associated host buffer. This value need not be a multiple of four.

The 32 bit XMIT DATA PTR value is the physical starting address of the associated host data buffer. This value need not be a multiple of four.

V. Transmission Process

FIG. **9** illustrates the network interface logic and host interface logic used in managing the transmit data buffer and transmit descriptor ring buffer in the independent memory on the adapter. On the host interface side, the logic includes host descriptor logic **150** and download DMA logic **151**. The host descriptor logic **150** and download DMA logic **151** are coupled to the host address space through the transmit "registers" including the XMIT AREA register, the XMIT COMPLETE THRESH register, the XMIT FAILURE register, the XMIT FRAME LENGTH register, the XMIT FRAME STATUS register, the XMIT PROTID register, the XMIT QUEUE STATUS register, and the XMIT START THRESH register. Details of these registers are described below.

The descriptors illustrated in FIG. **6** are stored in the transmit descriptor area of the host independent RAM on the adapter by host writes to the XMIT AREA address block. Three different processes act upon entries in the descriptor queue. The host writes to initially create transmit descriptors, bus master downloads to move buffer data from host memory to the transmit data buffer and transmission of described frames on the network. The first two processes occur within the download DMA logic **151** and the host descriptor logic **150**. Transmission is performed by the transmit DMA logic **155**. During the course of adapter operations, the number and status of descriptors in the transmit descriptor ring buffer **152** will vary according to the relative speeds of the host write, download, and transmission processes.

Two variables within the download DMA logic **151** helped describe the status of the transmit descriptor queue. ZERO DOWNLOADS PENDING indicates that there are no complete frame descriptors yet to be processed by the download DMA logic **151** for download. The ZERO FRAMES RESIDENT variable indicates that there are no descriptors which have been already processed by the download process, but are yet to be transmitted.

Each of the three processes that make up the transmission mechanism maintains its own set of pointers to the descriptor queue.

The host descriptor logic **150** generates pointers for the transmit descriptor ring buffer **152** on the adapter memory, which identify the current position of host accesses for writing descriptors in the transmit descriptor ring buffer **152**. These pointers are designated the current host descriptor pointer CHD, and the host write pointer HW. The current host descriptor pointer CHD points to the base address of a descriptor currently being written, or expected to be written by the host. The host write pointer HW points to the location within the current descriptor (pointed to by the CHD pointer) to which the host is expected to write next. That is, the HW pointer predicts the offset within the XMIT AREA address block at which the host will be writing to complete the current descriptor.

The download DMA logic **151** generates 3 pointers, and a buffer select signal, while managing downloads from the host system into the transmit data buffers XMIT DATA BUF**0** **153** and XMIT DATA BUF1 **154**. The pointers generated by the download DMA logic **151** include the current download descriptor pointer CDD which points to the base address of a descriptor that is currently being processed by the bus master download DMA logic **151**. The second pointer generated by the download DMA logic includes the download data pointer DD which points to the location within the current transmit data buffer (either XMIT. DATA BUF**0** or XMIT DATA BUF1) to which the download process is writing data. The third pointer generated by the download DMA logic **151** includes the current download buffer CDB pointer. The CDB pointer points to the buffer descriptor within the transmit descriptor pointed to by the CDD pointer, in which the specification of the buffer in host memory subject of a current download process resides.

The download DMA logic also selects the current transmit data buffer **153** and **154** to which the download DMA logic transfers data in the bus master operation as heuristically illustrated by signal BUF**1/0**.

The transmit DMA logic **155** generates three pointers for the transmission process. These pointers include the current transmit descriptor CXD pointer, which points to the base address of the descriptor in the transmit descriptor ring buffer **152** currently being processed by the transmit logic **155**. The transmit read XR pointer indicates the location within the current descriptor or current transmit data buffer (**153** or **154**) from which the transmission process is reading data to be transmitted.

The XMIT TAIL pointer (XT) points to the back end of the queue. The XT pointer points to an older descriptor in the transmit descriptor ring buffer **152** than the CXD pointer when there are frames that have completed transmission, but have not yet had their status information read by the host system.

The preferred system operates in two modes relative to generating indications to the host of the completion of a transmission. The preceding paragraphs described the data structures associated with host write, bus master download, and transmission processes when a XMIT COMPLETE ON DOWNLOAD variable is false and the indication is generated when the transmission is complete, or when XMIT COMPLETE THRESH (described below) is met. When XMIT COMPLETE ON DOWNLOAD is true, the download DMA logic **151** is also responsible for keeping track of the frames for which status has not been read. In this mode, the host receives the transmit complete indication for a frame upon download of the frame, so it is possible that the frame can be transmitted before the host is able to respond to the indication. This condition allows the XMIT TAIL pointer to no longer define the oldest useful entry in the descriptor ring buffer. Therefore, a frame status process is necessary. The frame status pointer FS points to the base address of the oldest descriptor for which status has not yet

been read by the host, in this mode of operation. Since the FS pointer is related to the transmit process, it is illustrated heuristically in the transmit DMA logic **155** in FIG. **9**. In the preferred system, however, it is logically coupled with the download DMA logic **151**.

The transmit DMA also selects the current transmit data buffer **103** or **104** from which a current transmit frame is being composed, as indicated heuristically by the signal **BUF0/1**.

The interface to the adapter is seen by the host as if it were a set of registers at a prespecified address block. The significant "registers" involved in transmission are outlined below.

A. XMIT AREA

The purpose of this register is to provide a mechanism for the host to supply the adapter with immediate data and a data structure that defines where in the host's memory system the remainder of the frame (if any) resides. The adapter stores this information in the transmit descriptor ring **152** for use when the frame being described can eventually be transmitted.

The adapter uses the address to which the data is written and the order in which it is written to determine what the data represents. The data must be written to the adapter using the structure and order described above with respect to FIG. **6**.

Bus master downloads begin after a descriptor has been written to XMIT AREA and XMIT QUEUE STATUS (described below) has been read by the host.

Once the host has completed the transfer of the transmit descriptor buffer structures to XMIT AREA, the host may read XMIT FRAME LENGTH to determine the number of bytes that the host has specified to be included in the transmit frame. XMIT QUEUE STATUS should then be read to advance the CHD pointer so that another frame may be written to this register. After reading XMIT QUEUE STATUS, the contents of XMIT FRAME LENGTH are undefined, until XMIT AREA is filled again.

The actual frame transmission onto the network will commence when two conditions are met: (1) the XMIT START THRESH (described below) condition has been met, or, if XMIT START THRESH is zero, when the entire frame has been copied to the adapter's RAM, and (2) when there are no previously queued transmit requests. If more than XMIT START THRESH bytes of immediate data are written to XMIT AREA, then network transmission may begin before XMIT QUEUE STATUS is read.

If the adapter runs out of XMIT AREA resources while the host is writing data to XMIT AREA, the host will be returned a value of 6 when it reads XMIT QUEUE STATUS. The writes that exceed the capacity of the descriptor buffer will not corrupt data already queued up on the adapter.

The transmit frame's destination and source addresses must be explicitly supplied to the adapter by the host for each frame transmitted. This information can be provided as part of the immediate data or, if there is no immediate data, as the first twelve bytes of the first data buffer specified in the descriptor.

Essentially, the host provides every byte of the frame between the start of frame delimiter and the frame check sequence (CRC) through the XMIT AREA register or download DMA operation.

B. XMIT COMPLETE THRESH

XMIT COMPLETE THRESH provides for an early indication of transmission completion.

The XMIT COMPLETE THRESH register is used to specify the number of transmit bytes that remain to be either transmitted or downloaded to the adapter (depending upon the adapter mode) before the adapter will issue a XMIT COMPLETE indication. Only bits **10** through **0** are implemented in this register. Values greater than the maximum frame length will prevent this function from operating properly. The method for disabling this function is to set the register to zero. The value in XMIT FRAME LENGTH (see below) is used to determine where the end of the transmit frame is.

If this threshold value is set too high, then the host-will respond to the indication before the adapter can provide a valid transmit status indication. If XMIT FRAME STATUS returns a ff (hex), then XMIT COMPLETE THRESH should be adjusted to delay the indication slightly. This is accomplished by reducing the value in the XMIT COMPLETE THRESH register. The function of this register is disabled during the transmission of the first 60 bytes of the frame. This register is cleared to 0 during a reset.

C. XMIT FAILURE

XMIT FAILURE returns the cause of a transmit failure.

This register returns the cause of the failure of the attempt(s) to transmit a queued frame. A non-zero value indicates that the frame encountered one or more errors during the transmission attempt.

bit 0	DMA UNDERRUN
bit 1	LOSS OF CARRIER SENSE
bit 2	MAX COLLISIONS
bit 3	SQE TEST FAILED

This register will contain valid data regardless of the success or failure of the attempt to transmit a frame. If there was no failure, then this register will contain a value of 0 (hex). The contents of this register are valid after the frame has completed transmission (low byte of XMIT FRAME STATUS not equal to ff (hex)) and before XMIT PROT ID is read.

If a data underrun occurs, the adapter will force a CRC error into the frame during transmission to assure that the frame is received as a bad frame and is discarded by the destination device.

D. XMIT FRAME LENGTH

XMIT FRAME LENGTH returns the number of bytes to be transmitted.

The XMIT FRAME LENGTH register returns the total number of bytes queued up for transmission by the current transmit frame descriptor identified by CXD pointer. This value is the total of the number of immediate data bytes and of all of the buffer length fields downloaded to the adapter for this frame. The value returned by this register does not reflect the effects of any padding of the frame that may be done by the adapter when the frame is less than 60 bytes in length.

The XMIT FRAME LENGTH register becomes valid immediately after the host writes the last byte to XMIT AREA and remains valid until the first write to XMIT AREA after a read of XMIT QUEUE STATUS.

E. XMIT FRAME STATUS

XMIT FRAME STATUS returns the results of a transmit attempt.

The least significant 16 bits of this register return the status of the attempt(s) to transmit a queued frame. The most significant 16 bits returns the XMIT REQ HANDLE for the frame. A value of XXXX0000 (hex) (XXXX is the XMIT REQ HANDLE for this particular frame) is returned for a successful transmission while XXXX000a (hex) is returned for a failed transmission. XXXX00fe (hex) is returned if the

adapter is in the process of retrying a transmission after a collision. If the transmission is still in progress, XMIT FRAME STATUS will return a xxxx00ff (hex).

If the frame was not transmitted successfully the specific cause of the transmit failure is available in XMIT FAILURE. Reading XMIT PROT ID advances XMIT FRAME STATUS to the status of the next transmitted frame, if any. If the "retry" status value is returned when XMIT FRAME STATUS is read, then reading XMIT FRAME STATUS will also clear the XMIT COMPLETE indication.

F. XMIT PROT ID

XMIT PROT ID returns the protocol ID of the transmit frame.

As soon as the adapter has completed its attempt(s) to transmit a queued frame and has posted its status, XMIT PROT ID can be read by the host as a method of identifying the frame. The value returned here is the same value that was written into the XMIT PROT ID field during the queuing of the frame via XMIT AREA.

Reading this register clears the XMIT COMPLETE indication except when the "retry" status value is read from XMIT FRAME STATUS. If "retry" was read, then reading XMIT FRAME STATUS will have cleared XMIT COMPLETE.

The XMIT PROT ID value resides in the upper 16 bits of the 32 bit register. The least significant 16 bits of this register will return the MACID value written to the MACID register. A double-word read will return both values simultaneously.

As multiple frames can be queued up for transmission, so can multiple transmission results be queued. Reading both words of XMIT PROT ID advances the completion status in XMIT FRAME STATUS, XMIT FAILURE, and XMIT PROT ID to the status of the next frame which has completed transmission, if any.

G. XMIT QUEUE STATUS

XMIT QUEUE STATUS returns the results of queuing a transmit frame.

A read of the XMIT QUEUE STATUS register returns the status of the host's attempt to queue up a transmit frame via XMIT AREA.

2 (hex)—Success: If the transmit request was successfully queued, this value is returned when XMIT QUEUE STATUS is read.

6 (hex)—Out of Resources: If the adapter runs out of queue storage RAM, then a status of 6 (hex) is returned.

7 (hex)—Frame Too Long: If the total number of bytes to be transmitted in a single frame exceeds the maximum frame length, this register will return a 7 (hex).

a (hex)—Order Violation: If the data written to XMIT AREA is written out of order, then this error code is returned.

ff Onex)—Adapter Not Ready: If XMIT QUEUE STATUS is read too quickly after the completion of the writes to XMIT AREA, it is possible to read the status value before the queuing process is complete.

Reading this register also advances XMIT AREA so that another transmit may be queued. This register MUST be read after all of the data has been written to XMIT AREA and before the next transmit request is written to XMIT AREA.

If an error code (6 (hex), 7 (hex), a (hex), or ff (hex)) is returned by XMIT QUEUE STATUS, then the frame was not queued and the host will have to attempt to queue it up another time. If the frame was flagged as being too long, it will have to be broken up into multiple frames before another queue attempt can be made.

If the code indicating success (2) is returned, then the host may immediately proceed to attempt to queue up an additional frame. The number of frames that can be queued depends on the amount of RAM on the adapter allocated for this purpose and the amount of immediate data included in each frame.

If the host attempts to queue a frame that is too large and also exceeds the available TRANSMIT DESCRIPTOR free space, the error that occurs first will take precedence and will be returned to the host.

H. XMIT START THRESH

XMIT START THRESH provides for an early start of transmission.

The XMIT START THRESH register is used to specify the number of transmit bytes that must reside on the adapter before it will start transmission. Values greater than the maximum frame length will prevent this function from operating properly. The method for disabling this function is to set the register to zero. Bytes are counted starting with the first byte of the destination field of the transmit frame.

The number of bytes considered to be available is the sum of the immediate data written to XMIT AREA by the host and those bytes transferred to the transmit data buffers in the adapter using bus master DMA operations. The transmit request will be posted immediately after XMIT START THRESH transmit frame bytes are made available from the immediate data or when the adapter has bus-mastered XMIT START THRESH—XMIT IMMED TEN bytes onto the adapter.

The number of bytes resident on the adapter must be equal to or greater than the value in XMIT START THRESH for the transmission to commence, unless the total frame size is less than XMIT START THRESH. In that case, the frame will begin transmission when the entire frame has been copied to the adapter. The actual transmission of the frame may be delayed by previous pending transmit frames and by deferrals to network traffic. This register is set to zero during a reset.

I. TRANSMISSION RING MANAGEMENT

FIGS. **10A–10E** illustrate the progression of the pointers used in the transmit operation. In FIGS. **10A–10E**, a portion of the transmit descriptor ring generally **200** and the transmit data buffers **201-0** and **201-1** are shown. Also, the pointers abbreviated as discussed above are indicated. One might also represent the transmit descriptor ring **200** in a circular fashion, as done below in FIGS. **12A–12F** for the receive ring.

In FIG. **10A**, the host descriptor logic is writing a first descriptor into the transmit descriptor ring **2130**. Thus, the CXD pointer points to the base address of the first descriptor, the HW pointer points to the offset from the base address to which the host is expected to write the next double word of the descriptor. The download pointers including CDD, CDB are also pointing to the base address of the first descriptor as no download operations have begun likewise, the transmit descriptors CXD and XR point to the same base address. Finally, the tail of the ring XT points to the beginning descriptor. The download data pointer DD points to the top of a first buffer, for instance, buffer zero.

As illustrated in FIG. **10B**, the first descriptor has completed writing and has begun the download process, and the host has begun writing a second descriptor. Thus, the host descriptor logic pointer CHD points to the base address of the next descriptor, and the HW pointer points to the expected address of the next byte. The download pointer CDD points to the base address of the first descriptor. The down DMA logic is assumed to be in the process of transferring buffers from host into the transmit data buffer. Thus, the CDB pointer points to a descriptor of a download

buffer in the first descriptor, and the DD pointer points to an offset within the transmit data buffer at which data is being downloaded. No transmit operations are yet to begin because the threshold has not been reached. Thus, the transmit pointers and the tail pointer still point to the first descriptor.

In FIG. **10**C, the host descriptor logic is working on a third descriptor, the download logic is working on the second descriptor, and the transmit logic is working on the first descriptor. The host descriptor logic pointers CXD and XR are working on the first descriptor. Thus, the CXD pointer points to the base address of the first descriptor, and the XR pointer points to immediate data being read for transmission by the transmit DMA logic.

The transmit read pointer XR will complete reading the immediate data and then move to the transmit data buffer filled by the download DMA logic when processing the first descriptor as. Illustrated in FIG. **10**B.

The transmit tail pointer XT still points to base address of the first descriptor.

The download logic is working on the second descriptor. Thus, the CDD pointer points to the base address of the second descriptor, the CDB pointer points to a buffer descriptor within the second descriptor, the DD pointer points to an offset within the second transmit data buffer to which the download DMA logic is transferring data from the host.

In FIG. **10**C, the host is writing a third descriptor into the XMIT AREA register. Thus, the CHD pointer points to the base address of the third descriptor, and the HW pointer points to the offset to which the next byte is expected to be written by the host.

In FIG. **10**D, the process has moved on so that the host is writing to a fourth descriptor. Thus, the CHD pointer points to the base address of the fourth descriptor, and the HW pointer points to the expected address of the next write to the XMIT AREA register.

The download logic is working on the third descriptor. Thus, the CDD pointer points to the base address of the third descriptor, the CDB pointer points to a buffer descriptor within the third descriptor, and the download data DD pointer points to a location in the first data buffer at which the download is occurring. This operation assumes that the transmit of the first descriptor has freed up the first data buffer for use by the download logic.

The transmit logic is working on the second descriptor. Thus, the CXD pointer points to the base address of the second descriptor, and the XR pointer points to a location in the transmit data buffer from which data is being read by the transmit logic. Since the status of the first descriptor is yet to be read, the transmit tail XT pointer still points to the base address of the first descriptor.

In FIG. **10**E, the process is assumed to have progressed so that the write of the fourth descriptor is completed, but the host has ceased writing new descriptors temporarily. In this case, the CHD pointer and the HW pointer point to the base address of a fifth descriptor waiting for further actions by the host. It is assumed that the download process has yet to complete downloading the third descriptor. Thus, the current CDD pointer points to the base address of the third descriptor, and the CDB pointer points to a buffer descriptor within the third descriptor. The DD pointer points to a location in the transmit data buffer to which the download process is downloading data.

In FIG. **10**E, it is also assumed that the transmission of the frame identified by the second descriptor is complete, and the transmit logic is waiting for the download operation on a third descriptor to either complete, or download sufficient data that the transmit logic may begin transmission. Thus, the CXD and the XR pointers point to the base address of the third descriptor.

It is assumed that the host has also read the transmit status of the first descriptor. This has the effect of moving the XT pointer to the base address of the second descriptor as illustrated in FIG. **11**E.

This process continues with automatic ring wraparound of the descriptors handled by the adapter. Also, overrun and underrun conditions are monitored and appropriate error signals indicated by the adapter.

VI. The Early Transmit Logic

The early transmit logic **6**A illustrated in FIG. **1** is implemented primarily in the download DMA logic **58** in the preferred implementation. The key components of a preferred implementation include data path arithmetic described with reference to FIG. **11**, and transmit start control logic illustrated with respect to FIGS. **12–17**.

FIG. **11** illustrates the basic logic of the data path arithmetic. This logic is founded primarily on 11 bit counter **300**. The 11 bit counter **300** includes a test input, a delta input, a counter enable CE input, a clock input, and a reset R input. The outputs of the 11 bit counter **300** include a carryout TC, and a counter output Q [**10:0**]. The test input is used for testing unrelated to the present invention. The delta input is driven by the download DMA byte enable [**3:0**], and sets an increment amount for the counter **300** between 1 and 4, based upon the number of bytes in the 32 bit bus that are active in the download process.

The counter enable CE input is driven by the output of AND gate **301**. The inputs to the AND gate **301** include the cyclone signal, and the output of delay circuit **302**. The delay circuit **302** receives as inputs the clock and the buffer download acknowledge bufferDwnAck signal. The cyclone signal is asserted during the last clock of a transfer cycle, and is used in concert with other control signals to generate read and write strobes within the download DMA. The bufferDwnAck signal is a cycle acknowledge signal for buffer download autonomous accesses on the adapter. It is asserted in response to a buffer download request. The delay circuit **302** operates to insure that it stays true for at least as long as the cyclone signal to insure proper incrementing of the counter.

The cyclone signal is generated during the last dock of a transfer cycle. It is the basis for incrementing the counter **300** by the amount indicated by data path byte enable [**3:0**].

The clock input **300** is driven by the system clock. The reset input is driven by OR gate **303**. The inputs to the OR gate include a newDescriptor signal indicating when a new descriptor is being processed by the download buffer, a downloadDmaReset signal, and a clearAllCounters signal for clearing counters on the adapter. The output of the counter **300** is supplied on an 11 bit line Q[**10:0**]. This output is supplied to adder **304**. The second input to the adder **304** is the immediate length value immedLenValue [**10:0**] which is read from the current descriptor. The sum of the immediate length value and the output of counter **300** is supplied as the downloadBytesResident value [**10:0**].

The data path arithmetic in FIG. **11** also illustrates generation of the bytesResidentValid signal. This signal is generated by a sequence of D-type flip-flops **305**, **306** which cause the bytesResidentValid signal, which is normally true, to go inactive for three clocks whenever the counter changes value. This allows for the download bytesResidentValue [**10:0**] to settle out of the adder and propagate through a comparator in the transmit start control logic described

below. Thus, the D-type flip-flops **305**, **306** have a reset input which is driven by the output of AND gate **301**, and a clock driven by the clock signal. Thus, these flip-flops are reset every time the counter **300** is incremented.

This logic is also used to generate a dataReady signal at the output of OR gate **307**. The inputs to this OR gate **307** are the dataPathByteEn signal [**3**:**0**]. Thus, the dataReady signal is asserted whenever a valid dataPathByteEn signal is generated.

A downloadCarryOut signal is generated for test purposes by the carryout signal TC of the counter **300**.

This logic is also used to generate a downloadDataPointer [**11**:**2**]. The downloadDataPointer is generated at the output of adder **308**. The inputs to the adder **308** include bits [**10**:**2**] of the output Q of counter **300**, and the output of the transmit buffer select MUX **309**. The inputs to the transmit buffer select MUX **309** include the offsets of the two transmit data buffers which are selected in response to transmit buffer select signal at the select input. Finally, this logic generates a dataPathOffset signal [**1**:**0**] at the output of the subtractor **310**. The subtractor generates a difference between signals A and B supplied at its inputs. Signal A is supplied at the output of latch **311**. The latch **311** latches the two least significant bits [**1**:**0**] of the output Q of the counter **300**. These two bits are latched in response to the doABuffer signal. The doABuffer signal is a command supplied to the buffer download logic to perform a bus master download of a buffer described in a transmit descriptor. Thus, these two bits are latched at the start of a download operation. The corresponding bits in the 32 bit download start address are subtracted from them to determine the data path offset for the buffer transfer.

Thus, this logic is coupled with the host address and monitors the downloading of data from the host into the transmit data buffer for use in making a threshold determination of the amount of data of a given frame resident in the adapter memory.

The downloadBytesResident [**10**:**0**] value is supplied to the transmit start control logic described with reference to FIGS. **12**–**15**. A block diagram of the transmit start control logic is provided in FIG. **12**. Each of the functional components of the block diagram in FIG. **12** are described with reference to FIGS. **13**–**17**.

FIG. **12** is a block diagram of the transmit start control logic. The transmit start control logic includes a start threshold register **320**, and a download compare block **321** which compares the value of the start threshold with the download bytes resident value generated in the data path arithmetic of FIG. **11**. Also, an immediate data comparator **322** is included to perform the comparison in the event that immediate data is present in the transmit descriptor. Finally, a data available control block **323** is included which generates the xmitDataAvailable signal for supply to the transmit DMA logic block of the adapter.

The transmit start control logic in FIG. **12** is responsible for generating the xmitDataAvailable signal, which is supplied to the transmit DMA module to indicate that there is data available to be transmitted onto the network. The xmitDataAvailable signal is asserted under three conditions: (1) there are one or more complete frames resident on the adapter ready to be transmitted; (2) there are no complete frames on the adapter, but the number of bytes resident for a frame currently being downloaded is greater than the value in the transmit start threshold register; and (3) there are no complete frames on the adapter, and there are no pending download operations, and the number of immediate bytes resident for the frame currently being written to the transmit area register is greater than the value of transmit start threshold.

The start threshold register **320** contains the physical XMIT START THRESH register, and logic to determine when the value in the register is valid. In particular, if the XMIT START THRESH value is written one byte at a time, then the START THRESH VALID signal is deasserted until the second byte is written. This keeps the download compare **321** and immediate compare **322** modules from making comparisons on transitional values. The inputs to the start thresh register **320** include the hostWriteData [**10**:**0**] which carries data through the start thresh register **320** during writes to the XMIT START THRESH register in the host interface. The startThreshWrite [**1**:**0**] signal are write strobes for the XMIT START THRESH register. The outputs of the start thresh register **320** include the startThreshValue [**10**:**0**] and the startThreshValid signal.

The inputs to the download compare module **321** include the startThreshValue [**10**:**0**] and the startThreshValid signal. Also, the downloadBytesResident [**10**:**0**] and the bytesResidentValid signals are supplied as inputs to the compare module **321**. The output of the download compare module **321** is the downloadThreshMet signal.

The inputs to the immediate compare module **322** include the startThreshValue [**10**:**0**] and the startThreshValid signal. Also, the immediate length value immedLengthValue [**10**:**0**] read from the transmit descriptor, the immedLengthValid signal. The immedLengthValid signal indicates that the value on immedLengthValue [**10**:**0**] is valid. Other inputs include the otherHostWrite [**3**:**0**] signal which are strobes indicating a host write of immediate data or buffer descriptor data. The four bits of the signal correspond to transfers on the four EISA bus byte lanes. The queueStatusRead signal indicating the host is reading status to prepare writing of a new transmit descriptor, the downloadDmaReset signal, which is a control signal from other portions of the adapter, a clock, a queueError signal, which indicates errors in host writes to the transmit area register; and a testAllCounters and clearAllCounters signals used for diagnostics and testing.

The output of the immediate compare module **322** is the immediateThreshMet signal. The downloadThreshMet and immediateThreshMet signals are supplied to the data available control block **322**. The other inputs include the downloadDmaReset signal, the clock signal, and the queueError signals. Also, the zeroDownloadsPending signal and zeroFramesResident signal are supplied to the data available control block **323**. These signals indicate when there are no complete transmit frames on the adapter, and when there are no pending download operations occurring, respectively. The output of the data available control block **323** is the xmitDataAvailable signal.

FIG. **13** illustrates the implementation of the start thresh register **320** of FIG. **12**. This register is composed of a 3 bit register **330** and an 8 bit register **331** to accommodate high and low bytes of the 11 bit threshold value. The 3 bit register stores bits **10**:**8** of the hostWriteData [**10**:**0**] bus. The 8 bit register **331** stores bits **7**:**0**. The 3 bit register **330** is clocked by the startThreshWrite [**1**] strobe and the 8 bit register **331** is clocked by the startThreshWrite [**0**] strobe. Both registers are reset on downloadDmaReset. The outputs of the registers **330** and **331** are supplied as the xmitStartData [**10**:**0**].

A start thresh valid state machine **332** is included. This state machine **322** receives as input the startThreshWrite strobes [**1**:**0**], the downloadDmaReset signal, and the clock. It contains a simple state machine as illustrated in FIG. **14** to monitor the startThreshWrite [**1**:**0**] and deassert the startThreshValid signal whenever one of the startThreshValue bytes is invalid. Thus, the state machine includes **3** states.

The first state INIT1 **335** is entered on downloadDmaReset. If the startThreshWrite [**0**] signal is true, and the startThreshWrite [**1**] signal is false, then the state machine goes to the LOW_BYTE_WRITTEN state **336**. When the startThreshWrite [**1**] signal is next asserted, then the state machine returns to the INIT1 state **335**. If, in the INIT1 state **335**, the startThreshWrite [**1**] signal is valid and the startThreshWrite [**0**] signal is not true, then the third state HIGH_BYTE_WRITTEN **337** is entered. When the startThreshWrite [**0**] signal is asserted, then the state machine returns to the INIT1 state **335**, during which the startThreshValid signal is asserted.

The download compare module **321** is illustrated in FIG. **15**. This logic includes a comparator **340** which receives a downloadBytesResident [**10**:**0**] value from the data path arithmetic of FIG. **11**, and the xmitStartData [**10**:**0**] from the start thresh register **320**. If the downloadBytesResident value is greater than or equal to the xmitStartData value, then the output of the comparator is asserted. The output of the comparator **340** is supplied to AND gate **341**. The AND gate **341** includes three inputs in addition to the output of the comparator **340**. The inputs include the startThreshValid signal from the start thresh register module **320**, and the output of logic **342** which tests whether the xmitStartData value is equal to zero. If it is not equal to zero, then the signal is asserted.

Finally, the bytesResidentValid signal from the data path arithmetic must be asserted before the downloadThreshMet signal is asserted through the AND gate **341**.

The immediate compare module **322** is illustrated in FIG. **16**. The immediate compare module **322** comprises an 11 bit counter **350**, a first comparator **351**, a second comparator **352**, and a third comparator **353**. Also, the output of the comparators is supplied through multiplexer **354**. The immedThreshMet signal is generated at the output of AND gate **355**. Also, an OR gate **356** and not equal to zero test logic **357** is included. The immediate compare module keeps track of the number of immediate data bytes resident using the 11 bit counter **350**. The inputs to the 11 bit counter include a test signal, otherHostWrite [**3**:**0**] signal, which indicates the delta or increment added by each dock, and a clock signal.

The 11 bit counter **350** is reset at the output of OR gate **356**. The inputs to the OR gate **356** include the downloadDmaReset signal, the queueStatusRead signal, the queueError signal, and the clearAllCounters signal. The outputs include an immedCarryout signal at the TC output, and the Q [**10**:**0**] bus which indicates the number of immediate bytes stored. The immediate bytes output is supplied as inputs to the first and second comparators **351** and **352**. The second input to the first comparator **351** is the immedLengthValue [**10**:**0**] which is retrieved from a descriptor currently being written. The second input to the second comparator **352** is the xmitStartData [**10**:**0**] from the start threshold register module **320**.

A third comparator **353** receives inputs including the immedLengthValue and the xmitStartData value.

The first comparator **351** generates a "clipped" signal in the event that the immediate bytes is greater than the immedLengthValue. The second comparator **352** generates the countCrossedThresh signal in the event that the immediate bytes is greater than or equal to the immedLengthValue. The third comparator **353** generates an immedCrossedThresh signal in the event that the immedLengthValue is greater than or equal to the xmitStartData value. The clipped signal is supplied at the select input of multiplexer **354**. This operates to cause the multiplexer **354** to select the countCrossedThresh signal as output Y so long as the number of immediate bytes does not exceed the immedLengthValue. When the number of immediate bytes exceeds the immedLengthValue, then the immedCrossedThresh signal is supplied at the output Y of the multiplexer.

The 11 bit counter **350** is incremented by 1, 2, 3, or 4, depending on the value of otherHostWrite [**3**:**0**].

The immediateThreshMet signal is supplied at the output of the AND gate **355**. The inputs to the AND gate **355** include the output of the logic **357** which is asserted when the xmitStartData value is non-zero. Also, the startThreshValid and immedLengthValid signals are supplied as inputs to the gate **355**. When all three of these signals are asserted, then the output of the multiplexer **354** is enabled for supply as the immediateThreshMet signal. When immediateBytes is less than immedLengthValue, clipped is inactive, and the output of comparator **352** is supplied as immediateThreshMet. When immediateBytes exceeds the immedLengthValue, then the value of immediate Bytes is disregarded, and the immedCrossedThresh comparator **353** takes over.

The assertion of queueStatusRead clears the counter, ensuring that it is always properly synchronized with the writing of a new frame descriptor to transmit area.

The data available control module **323** consists of a four-state state machine illustrated in FIG. **17**. The four states include an INIT2 state **370**, an IMMEDIATE_THRESH_MET state **371**, a DOWNLOAD_THRESH_MET state **372**, and a FRAME_RESIDENT state **373**. The INIT2 state **370** is entered upon downloadDmaReset. If the zeroFramesResident, and zeroDownloadsPending, and immediateThreshMet signals are true, then the IMMEDIATE_THRESH_MET state **371** is entered. In this state, the xmitDataAvailable signal is asserted. The IMMEDIATE_THRESH_MET state returns to the INIT2 state **370** upon the assertion of queueError. Alternatively, when the zeroDownloadsPending signal is deasserted and the queueError signal is not asserted, then the control proceeds from the IMMEDIATE_THRESH_MET state **371** to the DOWNLOAD_THRESH_MET state **372**. During this state, the xmitDataAvailable signal is true. If the zeroFramesResident signal is not true, then the state machine proceeds to the FRAME_RESIDENT state **373** from the DOWNLOAD_THRESH MET state **372**. In this state, the xmitDataAvailable signal remains asserted. Upon assertion of zeroFramesResident, the state machine returns to the INIT2 state **370** where xmitDataAvailable is false.

Alternatively, from the INIT2 state **370**, if the zeroFramesResident signal is deasserted, then the state machine proceeds directly to the FRAME_RESIDENT state **373**. Also, in the event that the zeroFramesResident signal is true, the zeroDownloadsPending signal is false, and the downloadThreshMet signal is true, then the state machine proceeds from the INIT2 state **370** to the DOWNLOAD_THRESH_MET state **372**. The state machine prevents transitions on the data available control input variables (downloadThreshMet, immediateThreshMet, zeroDownloadsPending, zeroFramesResident) from causing spudous indications on the xmitDataAvailable control line.

VII. Underrun Control Logic

The transmit logic **39** of FIG. **2** includes underrun control logic coupled to the data path as shown in the simplified block diagram of FIG. **18**. In FIG. **18**, the transmit data path **400** is illustrated receiving transmit bytes across line **401** from the transmit DMA logic. Also, a transmit write signal TXWR is supplied on line **402** to the transmit data path **400**.

The transmit data path receives the transmit bytes across line **401** and serializes them for supply on output line **403**. Also, data from the frame is supplied across line **404** to CRC logic **405** for generating error detection codes to be appended to the end of frames according to standard media access control functions. The output of the CRC logic **405** is supplied on line **406** to an exclusive OR gate **407**. Exclusive OR gate **407** passes the CRC code through to line **408** unless a bad frame signal on line **409** is asserted.

The data on line **403** and the CRC on line **408** are supplied to a multiplexer **410** which is controlled by transmit control logic **411** to supply the data of the frame followed by the CRC of the frame for valid frames to the serial output line **412**. Line **412** is then coupled to a transceiver for transmission on the network.

According to the present invention, this transmit data path includes an underrun detector **413** for detecting a condition in which the transferring of data into the transmit data buffer, or immediate data to the transmit descriptor buffer, by the host interface falls behind the transferring of data into the transmit data path **400** by the transmit DMA logic. The underrun detector **413** is controlled by the transmit control logic **411**. The transmit control logic **411** indicates intervals across line **414** during which a transmit write TXWR signal is expected on line **402**. The underrun detector determines that a transmit write TXWR signal is not present during an expected interval of the frame transmission, then a bad frame signal is generated on line **409**. In response to the bad frame signal, the CRC data is inverted by the exclusive OR gate **407** which causes a bad CRC to be generated for the already transmitted portions of the frame suffering the underrun. Transmit control logic **411** also responds to the bad frame signal on line **409** to select the bad CRC data through multiplexer **410**. Finally, the bad frame signal on line **409** is used for posting status information through the xrnitFailureRegister of an underrun condition.

VIII. Conclusion

Thus, the present invention allows for early transmission start in the system described above. By loading the XMIT START THRESH register, the system is enabled to begin transmission of a frame onto the network media prior to transfer of the entire frame into the adapter's buffer. The early transmission start is enabled by setting the XMIT START THRESH register to a non-zero value.

XMIT START THRESH is used to specify the number of bytes of the transmit frame that must reside on the adapter, either as immediate data or as data transferred onto the adapter via bus master data transfers, or as a combination of the two, before the adapter can commence with the media access control functions associated with transmitting the frame.

If the number of immediate data bytes expected by the adapter (as determined by the value written to the XMIT IMMEDIATE LENGTH field of the XMIT AREA) exceeds the value set in XMIT START THRESH, then the adapter will begin transmitting the frame immediately after sufficient writes to the XMIT AREA have occurred. Otherwise, the start of transmission will be delayed until, as a bus master, the adapter has downloaded enough data into its onboard buffer to satisfy the early transmit start threshold requirement. If, however, the threshold value is greater than or equal to the frame length, then transmission will commence once the entire frame is resident on the adapter. Of course, the start of the transmission may be delayed by other queued transmission, or by deferral to network traffic.

If this register set to zero, then the early transmit feature is disabled and the entire transmit frame must reside on the adapter before the adapter will begin to transmit it.

The value for this register may be programmed by the host to optimize performance. If set too low, system latencies or bandwidth limitations may cause the adapter to underrun the network during transmission, causing a partial frame with a guaranteed bad CRC to be transmitted. If the value is set too high, then unnecessary delays will be incurred before the start of transmission. The adapter generates an indication of an underrun condition which is made available to the host through the XMIT FAILURE register. If such an underrun indication occurs, then the host driver should increase the value on the XMIT START THRESH register. Further underrun indications should cause the driver to continually increase the XMIT START THRESH value. If the XMIT START THRESH value is increased to a value of greater than the maximum length expected by the system, then the early transmit start features should be disabled by writing a zero to the XMIT START THRESH register.

The foregoing description of preferred embodiments of the present invention has been provided for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Obviously, many modifications and variations will be apparent to practitioners skilled in this art. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, thereby enabling others skilled in the art to understand the invention for various embodiments and with various modifications as are suited to the particular use comemplated. It is intended that the scope of the invention be defined by the following claims and their equivalents.

What is claimed is:

1. A method for transmitting a frame of data from a host system through a network interface device to a network, comprising:

executing a frame transfer task initiated in the host system to transfer a frame to a buffer memory in the network interface device; and

executing a frame transmission task in the network interface device to initiate transmission of the frame from the buffer memory to the network in parallel with the frame transfer task before the frame is completely transferred to the buffer memory.

2. The method of claim **1**, wherein the frame transmission task includes executing a carrier sense, multiple access protocol.

3. The method of claim **1**, wherein the frame transmission task includes executing a carrier sense, multiple access with collision detection protocol.

4. The method as in claim **1**, further comprising:

detecting an underrun condition in which the transfer of the data of the frame into the buffer memory falls behind the transmission of the data from the buffer memory to the network; and

supplying a bad frame signal to the network un response to the underrun condition.

5. The method as in claim **1**, wherein the frame transmission task includes:

appending an error detection code to the frame of data to be transmitted to the network.

6. The method as in claim **5**, further comprising:

detecting an underrun condition in which the transfer of the data of the frame into the buffer memory falls behind the transmission of the data from the buffer memory to the network transceiver; and

corrupting the error detection code in response to the underrun condition.

7. The method as in claim 1, including making a threshold determination based on a comparison of a count of data transferred into the buffer memory with a threshold value.

8. The method as in claim 7, further comprising:

providing access to the threshold value so that it may be dynamically programmed by the host system; and

posting status information for use by the host system as feedback for optimizing the threshold value.

9. A method of transmitting frames of data from a host computer to a communications medium in a computer network, comprising:

transferring data of a frame from the host computer to a buffer memory;

monitoring the transfer of the data of the frame to the buffer memory to make a threshold determination of whether a threshold amount of data of the frame has been transferred to the buffer memory; and

initiating transmission of the frame of data from the buffer memory to the communications medium in response to the threshold determination prior to transfer of all the data of the frame from the host computer to the buffer memory.

10. The method as in claim 9, wherein transmission of the frame of data from the buffer memory to the communications medium is initiated as soon as the threshold amount of data has been transferred to the buffer memory.

11. The method as in claim 9, wherein the buffer memory is addressable by the host computer.

12. The method as in claim 9, wherein the threshold amount is dynamically programmable by the host computer.

13. The method as in claim 9, further comprising:

posting status information for use by the host computer as feedback for optimizing the threshold amount.

14. The method as in claim 9, wherein the initiating transmission of the frame step includes:

retrieving data from the buffer memory; and

supplying the retrieved data for transmission on the communications medium if no carrier is detected on the communications medium.

15. The method as in claim 9, wherein transmission of the frame onto the communications medium is initiated when the threshold amount is satisfied and no previously queued transmit requests exist.

16. The method as in claim 9, further comprising:

detecting an underrun condition in which the transfer of the data of the frame into the buffer memory falls behind the transmission of the data from the buffer memory to the communications medium; and

supplying a bad frame signal to the communications medium in response to the underrun condition.

17. The method as in claim 9, further comprising:

identifying the data of the frame to be transmitted on the communications medium including,

determining a frame length for the frame,

providing an indication field to selectively control whether an indication will be generated to notify the host computer upon completion of the transmission of the frame, and

providing a status field containing an indication of whether the frame to be transmitted has been transmitted.

18. The method as in claim 17, wherein the identifying step further includes reading a descriptor for the frame which includes a destination address, a source address, protocol-specific header data, and immediate data.

19. The method as in claim 9, further comprising:

appending an error detection code to the frame of data to be transmitted on the communications medium.

20. The method as in claim 19, further comprising:

detecting an underrun condition in which the transfer of the data of frames into the buffer memory falls behind the transmission of the data from the buffer memory to the communications medium; and

corrupting the error detection code in response to the underrun condition.

21. The method of transmitting frames of data from a host computer to a communications medium of a network, comprising:

receiving data of a frame from a host computer in a buffer memory;

monitoring the amount of the data of the frame in the buffer memory to determine whether a threshold amount of data of the frame has been received in the buffer memory;

initiating transmission of the frame of data from the buffer memory to the communications medium prior to receiving all the data of the frame from the host computer in the buffer memory in response to the threshold amount;

detecting an underrun condition in which the receiving of the data of the frame into the buffer memory falls behind the transmission of the data from the buffer memory to the communications medium;

supplying a bad frame signal to the communications medium in response to the underrun condition; and

posting status information for use by the host computer as feedback for optimizing the threshold amount.

22. The method as in claim 21, wherein the threshold amount is dynamically programmable by the host computer.

23. The method as in claim 21, wherein transmission of the frame onto the communications medium is initiated when the threshold amount is satisfied and no previously queued transmit requests exist.

24. The method as in claim 21, further comprising:

appending an error detection code to the frame of data to be transmitted on the communications medium.

25. The method as in claim 24, wherein the bad frame signal includes a corrupted error detection code.

26. The method as in claim 21, wherein the identifying step includes,

determining a frame length for the frame,

providing an indication field to selectively control whether an indication will be generated to notify the host computer upon completion of the transmission of the frame, and

providing a status field containing an indication of whether the frame to be transmitted has been transmitted.

27. The method as in claim 26, wherein the identifying step further includes reading a descriptor for the frame which includes a destination address, a source address, protocol-specific header data, and immediate data.

28. A method of controlling communication of data from a host system to a network transceiver coupled to a carrier sense, multiple access with collision detection (CSMA/CD) computer network, comprising:

transferring data of a frame from the host system to a buffer memory;

monitoring the transfer of the data of the frame to the buffer memory to make a threshold determination on an amount of data of the frame transferred to the buffer memory; and

initiating CSMA/CD transmission of the frame of data in the buffer memory to the network transceiver in response to the threshold determination prior to transfer of all the data of the frame from the host system to the buffer memory.

29. The method as in claim **28**, wherein the buffer memory is addressable by the host system.

30. The method as in claim **28**, wherein the threshold determination is based on a comparison of the amount of data transferred into the buffer memory with a threshold value, the threshold value being dynamically programmable by the host system.

31. The method as in claim **30**, further comprising:

posting status information for use by the host system as feedback for optimizing the threshold value.

32. The method as in claim **30**, wherein transmission of the frame to the network transceiver is initiated when the threshold value is satisfied and no previously queued transmit requests exist.

33. The method as in claim **28**, wherein the initiating transmission of the frame step includes:

retrieving data from the buffer memory; and

supplying the retrieved data for transmission to the network transceiver.

34. The method as in claim **28**, further comprising:

detecting an underrun condition in which the transfer of the data of the frame into the buffer memory falls behind the transmission of the data from the buffer memory to the network transceiver; and

supplying a bad frame signal to the network transceiver in response to the underrun condition.

35. The method as in claim **28**, further comprising

identifying the data of the frame to be transmitted to the network transceiver, including,

determining a frame length for the frame,

providing an indication field to selectively control whether an indication will be generated to notify the host computer upon completion of the transmission of the frame, and

providing a status field containing an indication of whether the frame to be transmitted has been transmitted.

36. The method as in claim **35**, wherein the identifying step further includes reading a descriptor for the frame which includes a destination address, a source address, protocol-specific header data, and immediate data.

37. The method as in claim **28**, further comprising:

appending an error detection code to the frame of data to be transmitted to the network transceiver.

38. The method as in claim **39**, further comprising:

detecting an underrun condition in which the transfer of the data of the frame into the buffer memory falls behind the transmission of the data from the buffer memory to the network transceiver; and

corrupting the error detection code in response to the underrun condition.

39. A method for transmitting a frame of data from a host system through a network interface device to a network, comprising:

initiating a frame transfer task in the host system to transfer a frame to a buffer memory in the network interface device;

monitoring the frame transfer task in the network interface device; and

executing a medium access task in the network interface device to initiate access to the network, and upon access transmitting the frame from the buffer memory to the network in parallel with the frame transfer task before the frame is completely transferred to the buffer memory.

40. The method of claim **39**, wherein the medium access task comprises a carrier sense, multiple access with collision detection protocol.

41. The method of claim **39**, further comprising:

detecting an underrun condition in which the transfer of the data of the frame into the buffer memory falls behind the transmission of the data from the buffer memory to the network; and

supplying a bad frame signal to the network in response to the underrun condition.

42. The method of claim **39**, wherein the communications medium compreses a random access communications medium.

43. The method of claim **39**, wherein the medium access task includes:

appending an error detection code to the frame of data to be transmitted to the network.

44. The method of claim **43**, further comprising:

detecting an underrun condition in which the transfer of the data of the frame into the buffer memory falls behind the transmission of the data from the buffer memory to the network transceiver; and

corrupting the error detection code in response to the underrun condition.

45. The method of claim **39**, including making a threshold determination based on a comparison of a count of data transferred into the buffer with a threshold value.

46. The method of claim **45**, further comprising:

providing access to the threshold value so that it may be dynamically programmed by the host system, and;

posting status information for use by the host system as feedback for optimizing the threshold value.

47. A method for transmitting a frame of data from a host system through a network interface device to a network, comprising:

executing a frame transfer task initiated in the host system to transfer a frame to a buffer memory in the network interface device;

making a threshold determination based on a comparison of a count of data transferred into the buffer memory with a threshold value;

executing a frame transmission task in the network interface device to initiate transmission of the frame from the buffer memory to the network in response to the threshold determination before the frame is completely transferred to the buffer memory;

posting status information for use by the host system as feedback for the threshold value; and

altering the threshold value based on the status information.

48. The method of claim **47**, including disabling the threshold determination during the step of altering the threshold value.

49. The method of claim **47**, including allowing alteration of the threshold value while data of a frame to be transmitted is stored in the buffer memory.

50. The method of claim **47**, wherein the step of altering the threshold value includes writing the threshold value in a register.

51. The method of claim 47, wherein the step of altering the threshold value includes using a driver in the host system to process the status information, and in response write a new threshold value in a register on the network interface device.

52. The method of claim 47, wherein the step of posting status information includes:

generating an indication of an underrun condition in the buffer memory.

53. The method of claim 52, wherein step of altering the threshold value includes increasing the threshold value in response to the indication of an underrun condition.

* * * * *

CERTIFICATE OF SERVICE

I certify that on this 14th day of July, 2015, I caused the foregoing document to be filed electronically with the Clerk of the Court using the CM/ECF System, which will send a notice of such filing to all counsel of record. Upon acceptance by the Court of the electronically filed document, six paper copies will be filed with the Court within the time provided in the Court's rules.

/s/ John C. Herman
John C. Herman

CERTIFICATE OF COMPLIANCE

I certify that this brief complies with the type-volume limitations of Fed. R. App. P. 32(a)(7)(B) because it contains 13,800 words, excluding those parts of the brief exempted by Fed. R. App. P. 32(a)(7)(B)(iii) and Federal Circuit Rule 32(b). I further certify that this brief complies with the typeface requirements of Fed. R. App. P. 32(a)(5) as well as the type style requirements of Fed. R. App. P. 32(a)(6).

Dated: July 14, 2015

/s/ John C. Herman
John C. Herman