US 20070064694A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2007/0064694 A1**

Ziedman (43) **Pub. Date: Mar. 22, 2007**

(54) **SYSTEM AND METHOD FOR CONNECTING A LOGIC CIRCUIT SIMULATION TO A NETWORK**

(76) Inventor: **Robert M. Ziedman**, Cupertino, CA (US)

Correspondence Address:
MACPHERSON KWOK CHEN & HEID LLP
2033 GATEWAY PLACE
SUITE 400
SAN JOSE, CA 95110 (US)

(21) Appl. No.: **11/557,053**

(22) Filed: **Nov. 6, 2006**

Related U.S. Application Data

(63) Continuation of application No. 10/044,217, filed on Nov. 19, 2001, which is a continuation-in-part of application No. 09/751,573, filed on Dec. 28, 2000, now Pat. No. 7,050,962.

(60) Provisional application No. 60/193,169, filed on Mar. 28, 2000.

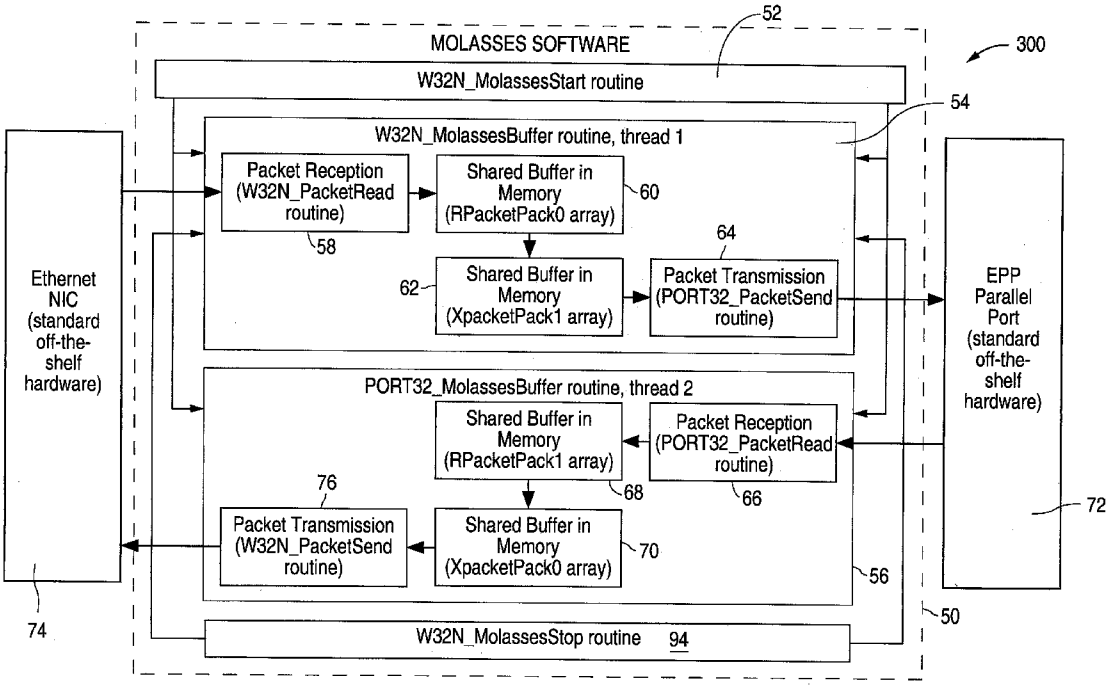
Publication Classification

(51) **Int. Cl.**
H04L 12/56 (2006.01)

(52) **U.S. Cl.** **370/389**; 370/466

(57) **ABSTRACT**

A system and method for connecting a running logic circuit simulation to a network running at a higher speed that includes a computer for receiving data packets from the network and storing the received data packets in a first buffer. The computer next transmits the received data packets to an electronic circuit in the logic circuit simulation at a slower speed. The computer also receives data packets from the electronic device under simulation, and stores the data packets received from the electronic device under simulation in a second buffer. The computer then transmits the data packets received from the electronic device under simulation to the network at a higher speed.



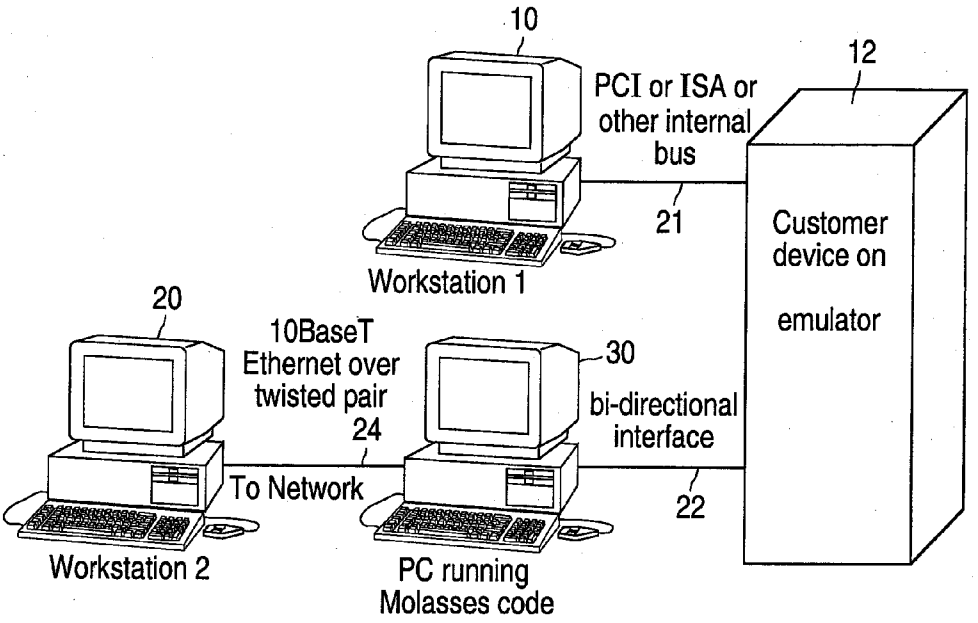

10
12
PCI or ISA or other internal bus
21
Customer device on emulator
Workstation 1
20
10BaseT Ethernet over twisted pair
24
30
bi-directional interface
To Network
22
Workstation 2
PC running Molasses code

FIG. 1

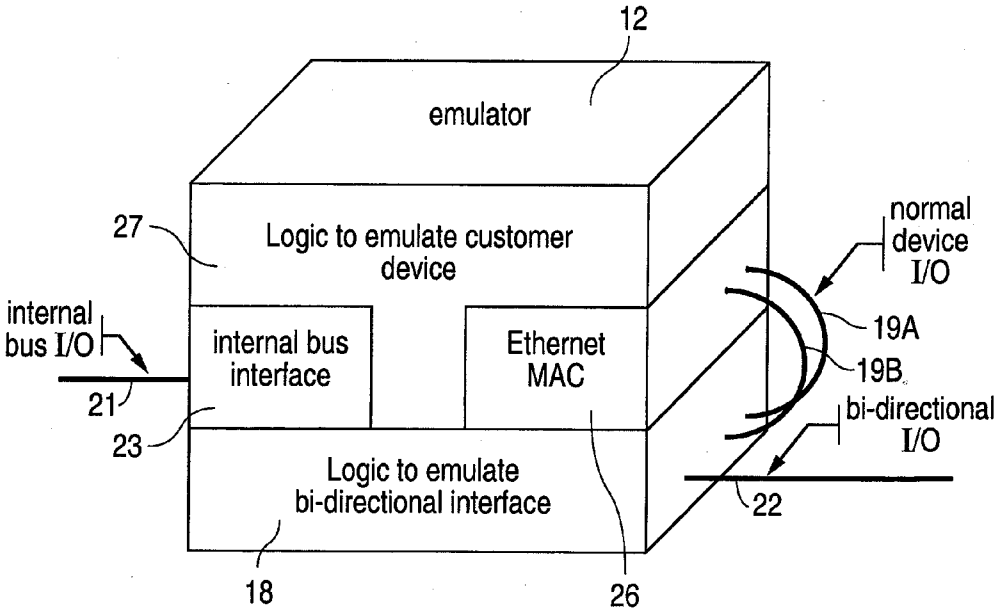

12
emulator
27
Logic to emulate customer device
normal device I/O
internal bus I/O
internal bus interface
Ethernet MAC
19A
19B
bi-directional I/O
21
23
Logic to emulate bi-directional interface
22
18
26

FIG. 2

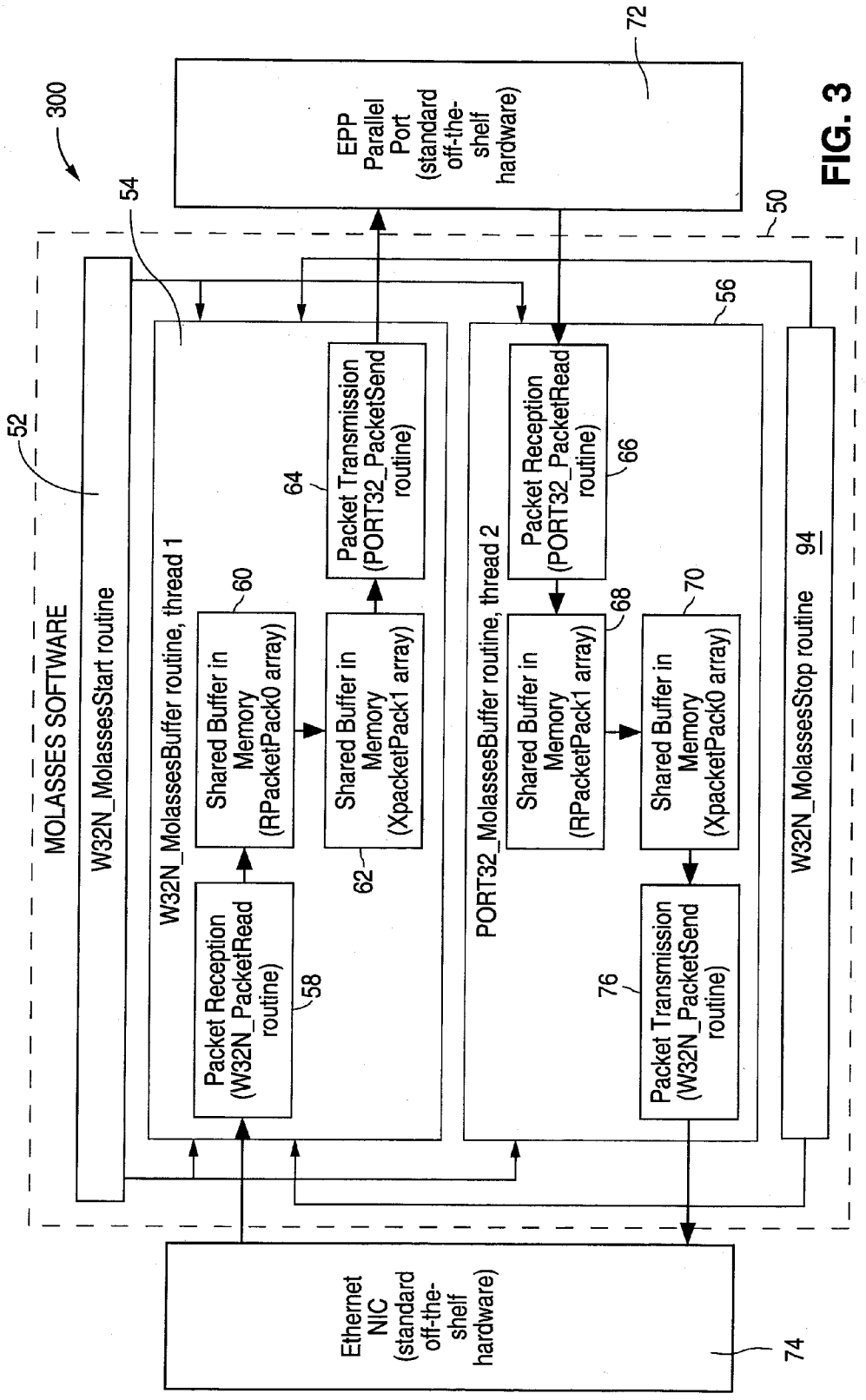

300
EPP Parallel Port (standard off-the-shelf hardware)
72
54
50
MOLASSES SOFTWARE
52
W32N_MolassesStart routine
W32N_MolassesBuffer routine, thread 1
Packet Transmission (PORT32_PacketSend routine)
64
Shared Buffer in Memory (RPacketPack0 array)
60
Shared Buffer in Memory (XpacketPack1 array)
62
Packet Reception (W32N_PacketRead routine)
58
PORT32_MolassesBuffer routine, thread 2
56
Packet Reception (PORT32_PacketRead routine)
66
Shared Buffer in Memory (RPacketPack1 array)
68
Shared Buffer in Memory (XpacketPack0 array)
70
Packet Transmission (W32N_PacketSend routine)
76
W32N_MolassesStop routine 94
Ethernet NIC (standard off-the-shelf hardware)
74

FIG. 3

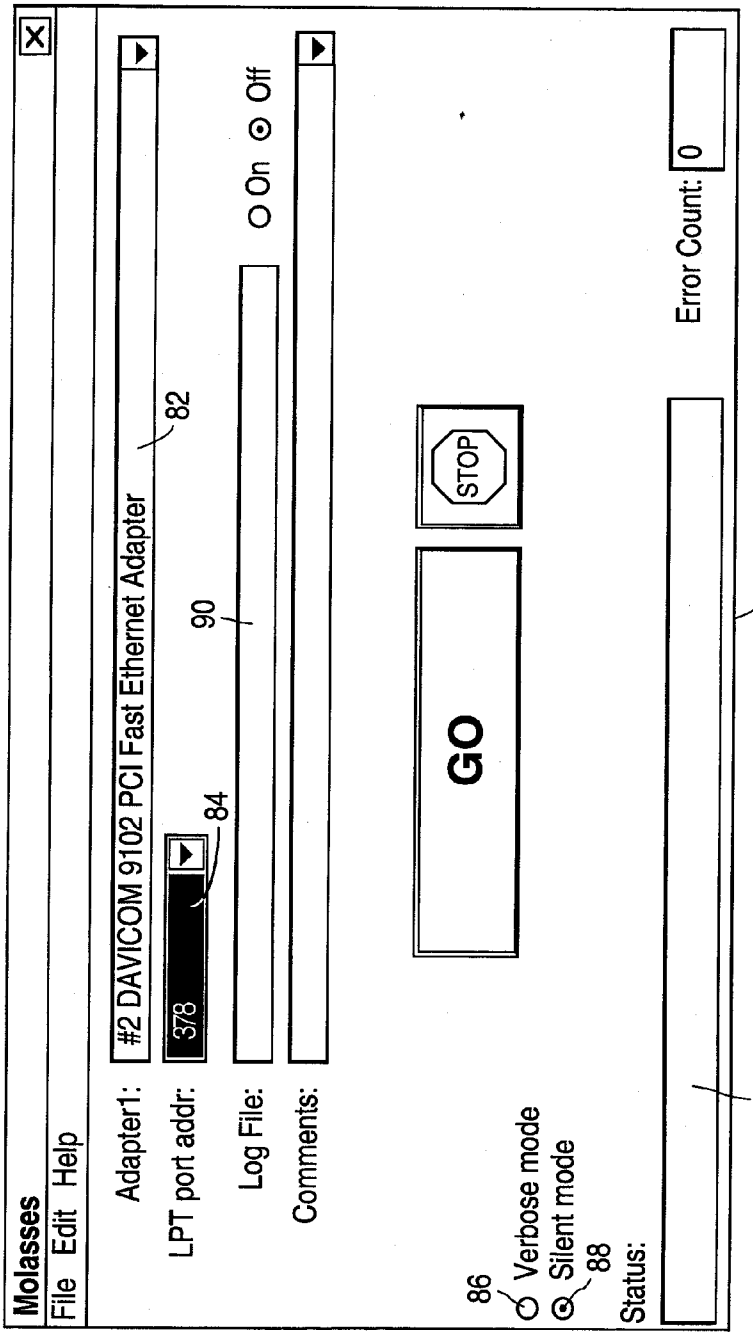

Molasses
File Edit Help
Adapter1:
#2 DAVICOM 9102 PCI Fast Ethernet Adapter
82
LPT port addr:
378
84
90
Log File:
On
Off
Comments:
GO
STOP
86
Verbose mode
Silent mode
88
Status:
Error Count: 0
80
92

FIG. 4

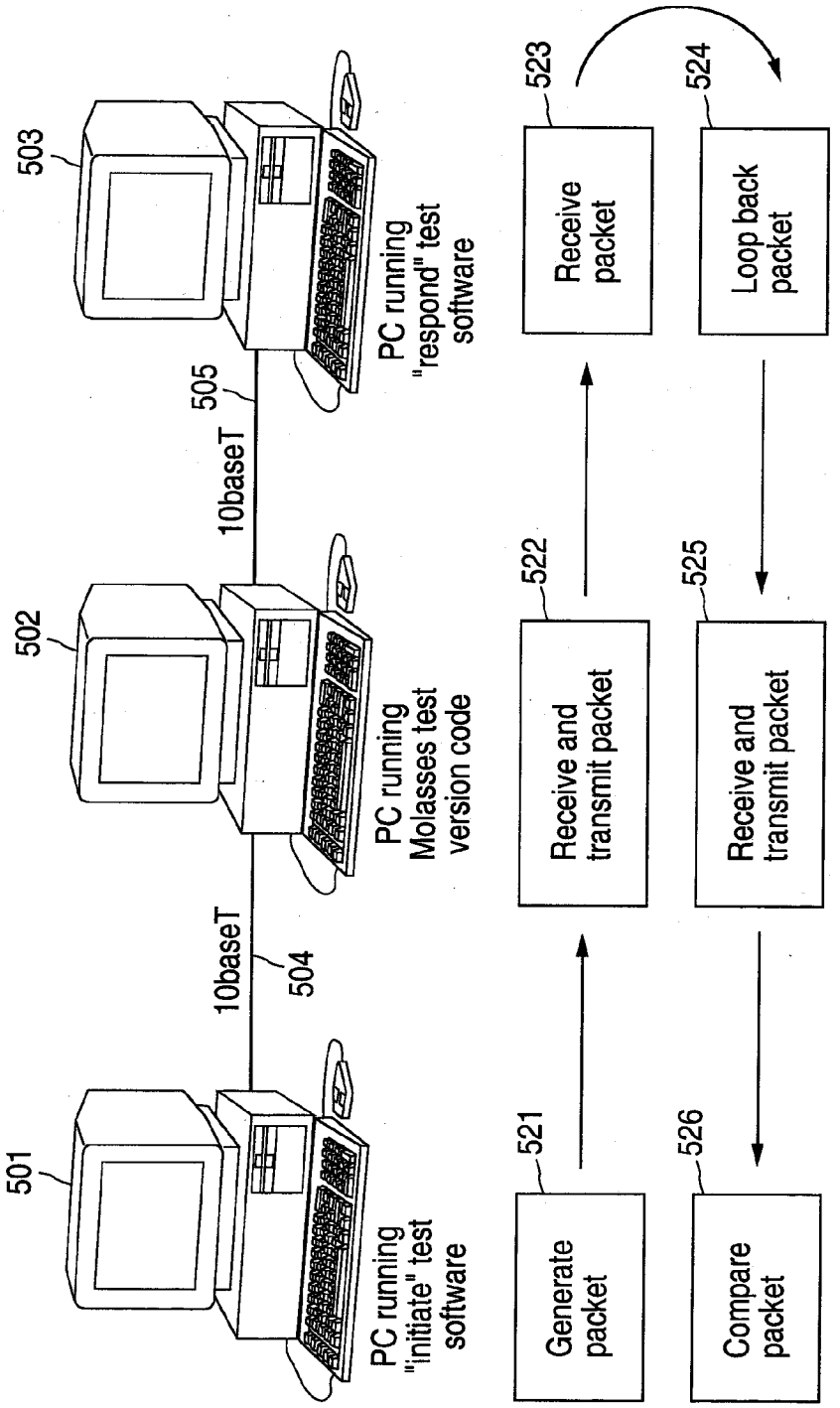

503
PC running "respond" test software
505
10baseT
502
PC running Molasses test version code
10baseT
504
501
PC running "initiate" test software
523
Receive packet
524
Loop back packet
522
Receive and transmit packet
525
Receive and transmit packet
521
Generate packet
526
Compare packet

FIG. 5

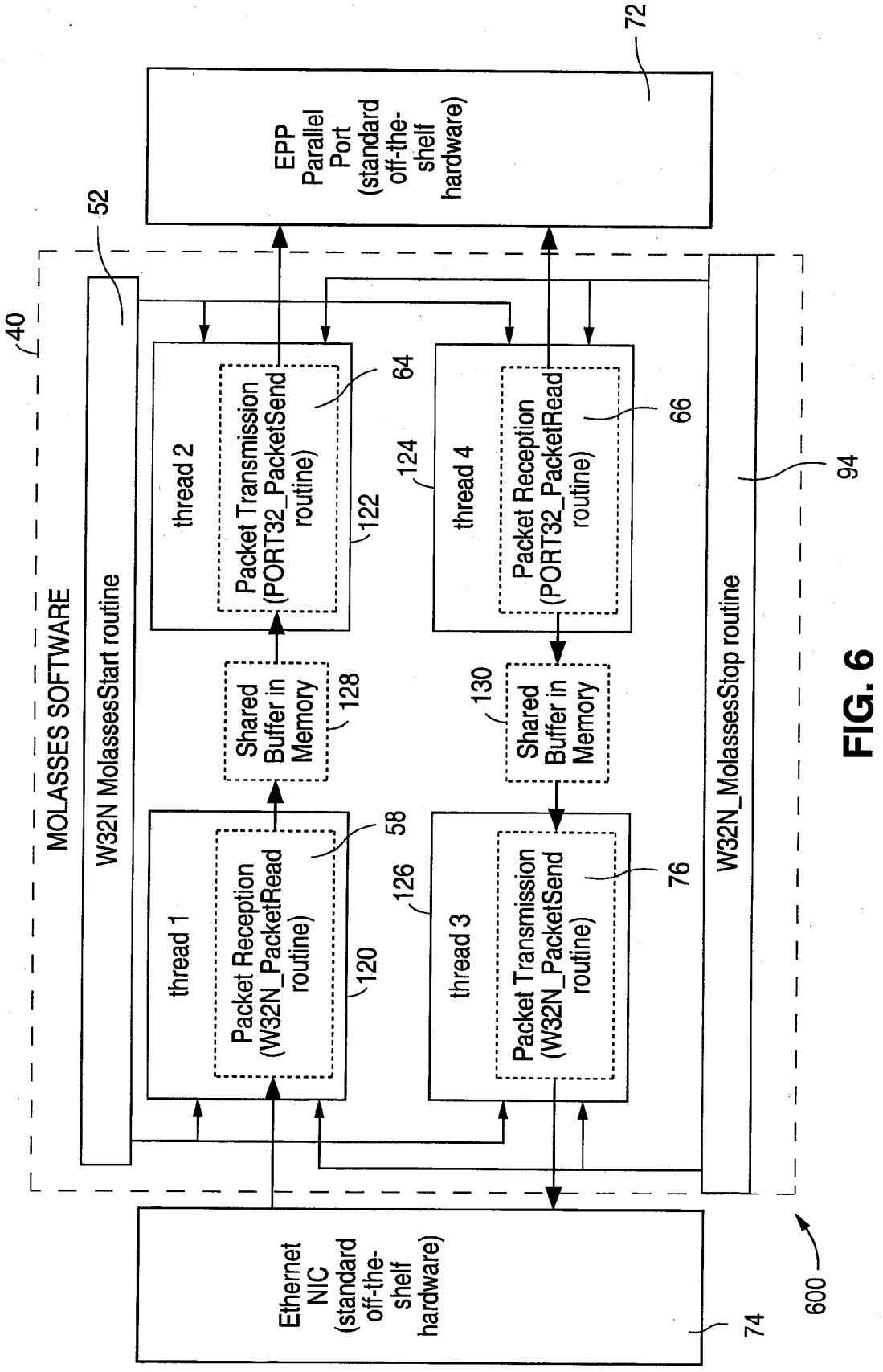

EPP Parallel Port (standard off-the-shelf hardware)
72
52
40
MOLASSES SOFTWARE
W32N MolassesStart routine
thread 2
Packet Transmission (PORT32_PacketSend routine)
64
122
thread 4
Packet Reception (PORT32_PacketRead routine)
66
124
94
W32N_MolassesStop routine
Shared Buffer in Memory
128
130
Shared Buffer in Memory
thread 1
Packet Reception (W32N_PacketRead routine)
58
120
thread 3
Packet Transmission (W32N_PacketSend routine)
76
126
Ethernet NIC (standard off-the-shelf hardware)
74
600

FIG. 6

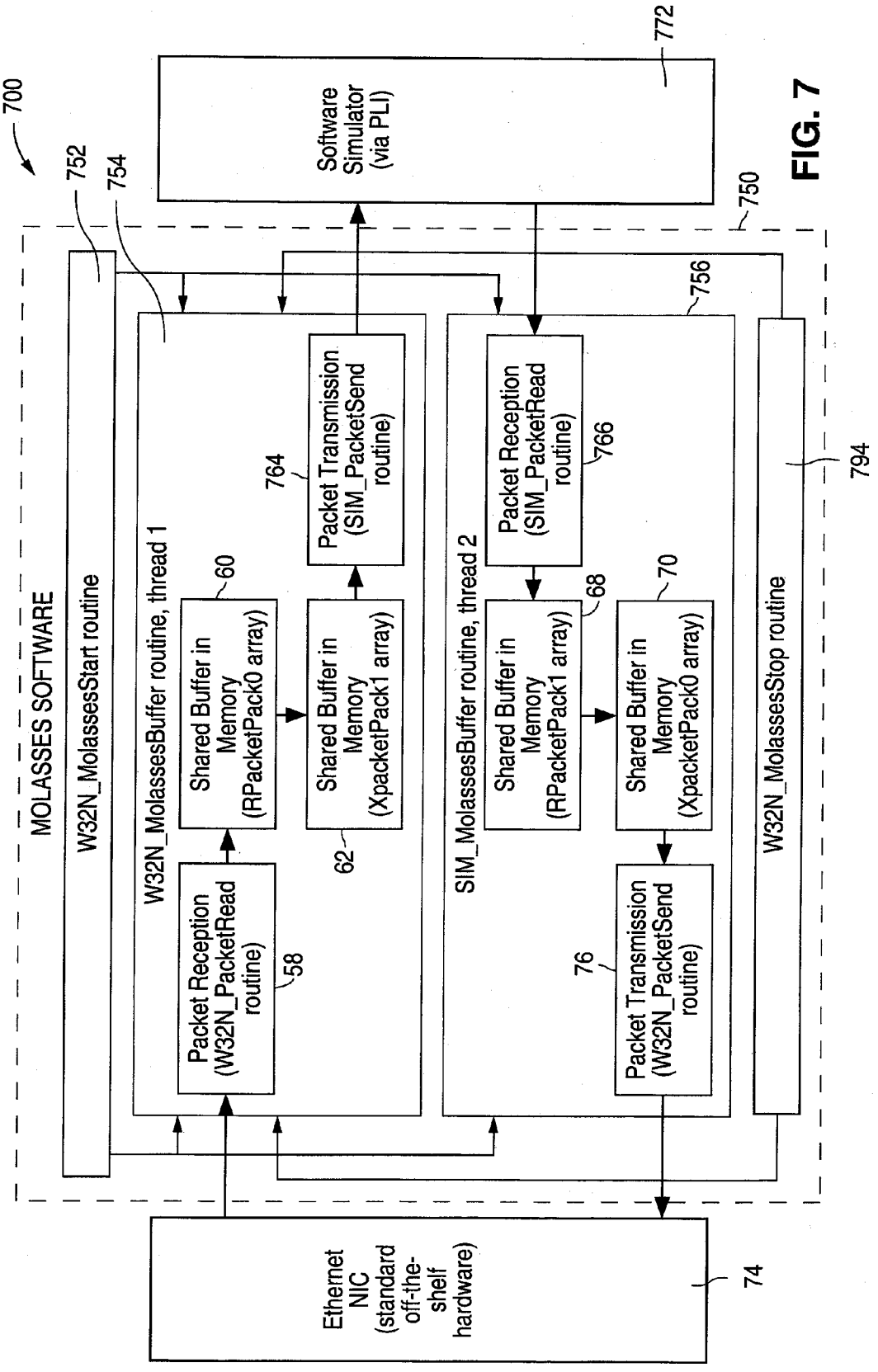

700
772
Software Simulator (via PLI)
752
754
750
MOLASSES SOFTWARE
W32N_MolassesStart routine
W32N_MolassesBuffer routine, thread 1
Shared Buffer in Memory (RPacketPack0 array)
60
Shared Buffer in Memory (XpacketPack1 array)
62
Packet Transmission (SIM_PacketSend routine)
764
Packet Reception (W32N_PacketRead routine)
58
SIM_MolassesBuffer routine, thread 2
Packet Reception (SIM_PacketRead routine)
766
Shared Buffer in Memory (RPacketPack1 array)
68
Shared Buffer in Memory (XpacketPack0 array)
70
Packet Transmission (W32N_PacketSend routine)
76
756
W32N_MolassesStop routine
794
Ethernet NIC (standard off-the-shelf hardware)
74

FIG. 7

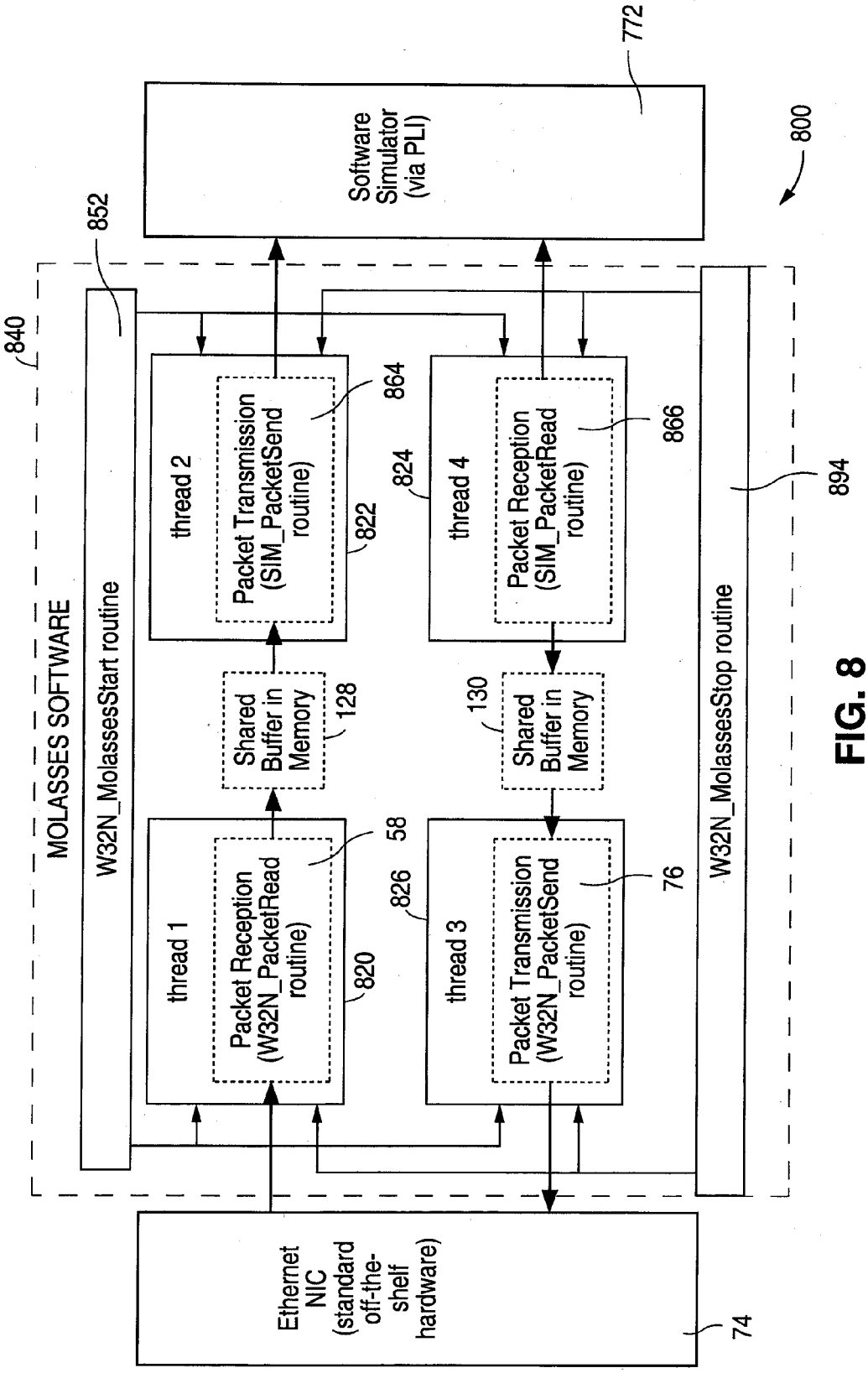

772
Software Simulator (via PLI)
800
852
840
MOLASSES SOFTWARE
W32N_MolassesStart routine
thread 2
Packet Transmission (SIM_PacketSend routine)
864
822
824
thread 4
Packet Reception (SIM_PacketRead routine)
866
894
W32N_MolassesStop routine
Shared Buffer in Memory
128
130
Shared Buffer in Memory
thread 1
Packet Reception (W32N_PacketRead routine)
58
820
826
thread 3
Packet Transmission (W32N_PacketSend routine)
76
Ethernet NIC (standard off-the-shelf hardware)
74

FIG. 8

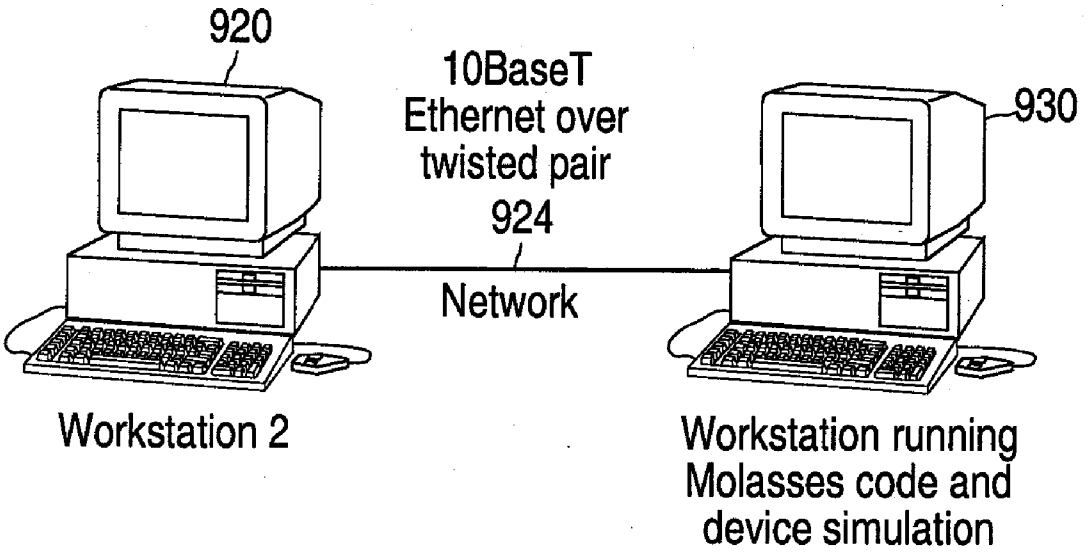

920
10BaseT
Ethernet over
twisted pair
924
Network
930
Workstation 2
Workstation running
Molasses code and
device simulation

FIG. 9

SYSTEM AND METHOD FOR CONNECTING A LOGIC CIRCUIT SIMULATION TO A NETWORK

CROSS-REFERENCE TO RELATED PATENT APPLICATION

[0001] The present application is a continuation-in-part application of copending U.S. patent application, entitled "METHOD FOR CONNECTING A HARDWARE EMULATOR TO A NETWORK," Ser. No. 09/751,573, filed on Dec. 28, 2000.

BACKGROUND OF THE INVENTION

[0002] Prior to reducing an integrated circuit design to a form suitable for fabrication, the integrated circuit design is often simulated in software on a computer, and emulated in hardware to allow the design to be optimized and debugged. Typically, using a hardware description language (e.g., VHDL), the circuit designer prepares a hardware description of the integrated circuit, which is then compiled into a software model to be simulated on a computer (e.g., an engineering workstation). Often, the hardware description can also be compiled into a hardware model that can be emulated in a hardware emulator. A hardware emulator suitable for such use typically includes field programmable gate arrays (FPGAs) that serve as a breadboard for implementing the integrated circuit design. Both the simulation and the emulator typically run at a slower speed than a computer network (e.g., an Ethernet network).

[0003] When an integrated circuit that has a computer network interface is simulated or emulated, network activities are usually simulated or emulated at the speed of the circuit emulator or the circuit simulation. When using a circuit emulator, a conventional network-emulation device is typically connected to a port of the circuit emulator. The circuit emulator receives data packets from the network-emulation device, re-packages the data and transmits the re-packaged data back at the speed of the circuit emulator. The re-packaged data is then received by the network-emulation device, which inspects the re-packaged data to determine if the integrated circuit under emulation in the circuit emulator correctly sends and receives data packets. However, on balance, such a conventional network-emulation device does not emulate network behavior accurately and correctly.

[0004] Alternatively, another conventional technique for connecting a circuit emulator to the network requires slowing down the network, receiving signals from the slowed network and translating the signals into suitable electrical signals in the form that the circuit emulator can accept. The circuit emulator, which typically operates at a slower speed than the network, can also send packets to the slowed network. However, because the network is designed to operate at a different speed, timing issues may arise in such a slowed network. These timing issues may require further modification to the network to resolve. Such modifications are undesirable because the modified network may not adequately represent network characteristics. Because not all network devices can be slowed to the circuit emulator speed, the circuit emulator is typically also limited to communication with a small subset of devices on the network.

[0005] Shortcomings of an emulation are typically also present in a logic circuit simulation.

SUMMARY OF THE INVENTION

[0006] The present invention allows a logic circuit simulator ("circuit simulation") running on a host processor to connect to a computer network at full network speed. The present invention provides a method and an apparatus for transferring data packets between a circuit simulation and the network. In one embodiment, an interface software program also installed on a host computer (e.g., a personal computer) is provided to handle communication between the network and the circuit simulator. The network can be, for example, an Ethernet network.

[0007] According to the present invention, data packets addressed to a device under simulation, or alternatively, addressed to a workstation connected to the network through the circuit simulation, is received and stored in buffers of the host computer. (In one example, a workstation is connected to a network through a simulated network interface.) The interface software in the host computer repackages the data packet into a second format for transmission to the simulated device at the speed of the circuit simulation. Under this arrangement, the interface software in the host computer need not send to the circuit simulation, for example, the preamble required to synchronize the clocks of the network and the circuit simulation, because the circuit simulation is usually not capable of providing an analog interface required to respond to the preamble. Similarly, the interface software in the host computer repackages the data packets received from the circuit simulation into proper format for transmission to the network at full network speed. Under this arrangement, the existing memory in the host computer is used to buffer data packets communicated between the circuit simulation and the network, so that data packets received from the network at network speed are transmitted to the circuit simulation at a slower speed, and data packets received from the circuit simulation at the slower speed is provided to the network at full network speed.

[0008] In one embodiment, the present invention allows the interface software of a host computer to individually examine a data packet of a conventional off-the-shelf interface card to identify the beginning and the end of the packet. When the beginning and the end of a data packet can be identified, the interface software of the host computer ignores data packets not addressed to the simulated circuit. Consequently, compared to the prior art, a much smaller amount of buffer memory is required. This arrangement loses data packets only in the occasional event of a buffer overflow, thus effectively preventing network connection loss.

[0009] In one embodiment, the interface software of the host computer is implemented as a multithreaded program having, in one instance, two executing threads. One thread is a task that receives data packets from the network interface card, stores the received data packets in a buffer, retrieves the stored data for repackaging, and sends the repackaged data over a simulation interface to the circuit simulation. Another thread is a task that receives packets from the simulation interface, repackages the data into a network data packet and sends the network data packet over the network interface card to the network.

[0010] In another embodiment, the interface software of the host computer is implemented as a multithread program including four executing threads. One thread is a task that

receives data packets from the network, and stores the received data packets in a buffer. A second thread is a task that polls the buffer for the received data packets. This second thread repackages the data packets and sends the repackaged packets over the simulation interface to the circuit simulation. A third thread is a task that receives data packets from the circuit simulation over the simulation interface and stores the received packets in a second buffer. A fourth thread is a task that polls the second buffer for the data packets received from the circuit simulation. This fourth thread repackages these data packets and sends the repackaged packets over the network interface to the network.

[0011] In yet another embodiment, the interface software of the host computer is also implemented as a multithread program, as in the previous embodiment, except that the second buffer is eliminated and the third and fourth tasks are combined into a single task executing as a single thread. In this embodiment, the single task receives data packets from the simulation interface from the circuit simulation, repackages the data packets received and sends the repackaged packets over the network interface to the network. This approach is possible because a circuit simulation runs at a much slower speed than the network, such that a data packet received from the circuit simulation can be repackaged and sent to the network before the next data packet's arrival from the circuit simulation.

[0012] Further features and advantages of various embodiments of the invention are described in the detailed description below, which is given by way of example only.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] FIG. **1** shows a configuration including first workstation **10**, second workstation **20**, host computer **30** and circuit emulator **12**, in accordance with the present invention.

[0014] FIG. **2** is a block diagram showing one configuration of circuit emulator **12**, during an emulation of a network interface card.

[0015] FIG. **3** is a block diagram **300** showing the functions performed by Molasses program **50**, in accordance with one embodiment of the present invention.

[0016] FIG. **4** shows a user interface Mainscreen **80** in Molasses program **50**.

[0017] FIG. **5** shows a test setup suitable for a self-test in Molasses program **50**, involving three personal computers (PC)

[0018] FIG. **6** is a block diagram **600** showing the functions performed by Molasses program **40**, in accordance with a second embodiment of the present invention.

[0019] FIG. **7** is a block diagram **700** showing the functions performed by Molasses program **750**, in accordance with a third embodiment of the present invention.

[0020] FIG. **8** is a block diagram **800** showing the functions performed by Molasses program **850**, in accordance with a fourth embodiment of the present invention.

[0021] FIG. **9** shows a configuration including first workstation **920** and host computer **930**.

[0022] In the following detailed description, to simplify the description, like elements are provided like reference numerals.

DETAILED DESCRIPTION

[0023] The present invention is illustrated in the following using first, as an example, an emulation of a network interface device. At a latter part of this description, the present invention is illustrated, by way of another example, a simulation of a network interface device. FIG. **1** shows a configuration including first workstation **10**, second workstation **20**, host computer **30** and circuit emulator **12**. In this embodiment, a network interface card is emulated in circuit emulator **12**, which interfaces with first workstation **10** over a conventional internal bus (e.g., a PCI bus). The network interface card emulated is intended to operate as a network interface of a workstation, such as workstation **10**. However, circuit emulator **12** does not operate at the full network speed the network interface card is intended. Circuit emulator **12** is connected to host computer **30** over bidirectional interface **22**, such as a conventional personal computer (PC) parallel port. Host computer **30** runs an interface program "Molasses" which is discussed in further detail below in conjunction with FIG. **4**. Host computer **30** connects to conventional computer network **24** (e.g., 10BaseT Ethernet over twisted pair) using a conventional network interface. Workstation **20** communicates with host computer **30** over computer network **24** using conventional network protocols. Host computer **30** can be, for example, a desktop PC running Windows 95 or Windows 98 and equipped with a 10baseT Ethernet controller card and two parallel ports. A proprietary parallel port interface can be used for faster transfer speeds or easier connection to circuit emulator **12**. In one embodiment, host computer **30** includes an Intel Pentium class processor and is equipped with 32 Mbytes of DRAM and 500 Mbytes of hard disk space. Host computer **30** also includes Media Access Controller ("MAC") drivers, parallel port drivers and the NDIS API (Network Driver Interface Specification—Application Program Interface). MAC drivers interface the operating system to the Ethernet card hardware. The parallel port drivers allow the operating system to interact with the parallel port hardware, and the NDIS API allows functional enhancements to network drivers in the operating system. Host computer **30** also includes a Graphical Users Interface ("GUI") and a packet capture, buffering and transmission application program as part of the "Molasses" program, which will be described in further detail later. Workstations **10** and **20** can each be any conventional workstation, including a PC running the Windows 98 operating system.

[0024] Alternatively, the network interface card can also be simulated in a circuit simulation running on host computer **30**. Host computer **30** also runs a version of the interface program "Molasses" (e.g., Molasses program **750** or **850**) which is discussed in further detail below in conjunction with FIG. **7** or FIG. **8**.

[0025] FIG. **2** is a block diagram of one configuration of circuit emulator **12** during an emulation of the network interface card. As shown in FIG. **2**, circuit emulator **12** provides logic circuit **18** that couples circuit emulator **12**'s circuit to bi-directional interface **22**, an emulated Ethernet MAC **26**, interface **23** to internal bus **21**, and logic circuit **27**, which is the remainder of the emulated network interface

card. Cable assemblies **19**A and **19**B connect the input terminals and the output terminals ("I/O terminals") of Ethernet MAC **26** and logic circuit **18**. Logic circuit **18** translates the signals of Ethernet MAC **26** (communicated over cable assemblies **19**A) into the signals of bi-directional interface **22** for transmitting to host computer **30**. In one embodiment, computer network **24** includes a conventional hub providing 10baseT connections. Alternatively, computer network **24** can also include a switch, which selectively transmits data packet based on destination addresses. Use of a switch can reduce packet traffic at a particular connection, and thus reduce the buffer requirements at the connected devices (e.g., host computer **30**). Providing a switch at host computer **30**'s connection to computer network **24** also simplifies the Molasses program **50** running on host computer **30**. In this configuration, emulator **12**'s connection to workstation **10** over internal bus **21** allows examination of signals in logic circuits **18** and **27** for debugging purpose.

[**0026**] FIG. **3** is a block diagram **300** showing the functions performed by Molasses program **50**, in accordance with one embodiment of the present invention. Molasses program **50** includes a graphical user interface illustrated in FIG. **4** by Mainscreen **80**. As shown in FIG. **4**, Mainscreen **80** allows the user to specify an Ethernet NIC (act **82**) and a port address for bidirectional interface **22** connected to emulator **12** (act **84**). Status line **92** displays continuously information about packets being processed by Molasses program **50**. The amount of information to be shown on status line **92** can be selected using verbose mode option **86** and silent mode option **88**. The user can also specify a log file (act **90**) to record information such as the value of each byte in each data packet, a count or the nature of errors that occur, or comments that the user may wish to add. The log file can be used for future reference and debugging purposes.

[**0027**] Referring back to FIG. **3**, Molasses program **50** interfaces with network interface card (NIC) **74**, which provides host computer **30**'s access to computer network **24**, and interface **72**, which couples host computer **30** through bidirectional interface **22** to circuit emulator **12**. In this embodiment, interface **72** can be a conventional parallel port operating under the conventional EPP standard. Once the parameters of Mainscreen **80** are set, Mainscreen **80** calls "W32N_MolassesStart" routine **52**. Routine **52** creates simultaneous threads running "W32N_MolassesBuffer" routine **54** and "PORT32_MolassesBuffer" routine **56**, respectively. W32N_MolassesBuffer routine **54** receives packets from the Ethernet NIC **74** (via a W32N_PacketRead routine **58**), stores the received packets into receive buffer **60** ("RPacketPack0") in host computer **30**'s main memory. Subsequently, the received packets in buffer **60** are transferred to transmit buffer **62** ("XPacketPack1"), from which they are then transmitted to interface **72** via "PORT32_PacketSend" routine **64**. PORT32_MolassesBuffer routine **56** receives packets from interface **72** (via "PORT32_PacketRead" routine **66**), stores the packets into receive buffer **68** ("RPacketPack1"). Subsequently, routine **56** then transfers the data packets in buffer **68** to a transmit buffer **70** ("XPacketPack0"), which are then transmitted to the Ethernet NIC **74** via "W32N_PacketSend" routine **76**. Molasses program **50** converts data packet formats, when necessary. For example, the preamble that is used in a packet for synchronizing the clock signals of the network and the emulated device is removed before being forwarded to circuit emulator **12** over interface **72**.

[**0028**] Mainscreen **80** calls "W32N_MolassesStop" routine **94** to terminate execution of both threads **54** and **56**.

[**0029**] FIG. **6** is a block diagram **600** showing the functions performed by Molasses program **40**, in accordance with a second embodiment of the present invention. As in Molasses program **50** of FIG. **3**, Molasses program **40** of FIG. **6** interfaces with network interface **74**, which provides host computer **30**'s access to computer network **24**, and interface **72**, which couples host computer **30** to bidirectional interface **22** to circuit emulator **12**. Interface **72** can be implemented by a conventional parallel port operating under, for example, the EPP standard. Once the parameters of Mainscreen **80** (FIG. **4**) are set, Mainscreen **80** calls "W32N_MolassesStart" routine **52**, which creates four threads **120**, **122**, **124** and **126**. Thread **120** executes "W32_PacketRead" routine **58**, which receives data packets from Ethernet NIC **74** and stores the received data packet into shared buffer **128** in the main memory of the host computer **30**. Thread **122** executes "Port32_PacketSend" routine **64**, which polls shared buffer **128** for the received data packets, repackages these data packets and sends them to circuit emulator **12** over emulation interface (parallel port) **72**. Thread **124** executes "Port32_PacketRead" routine **66**, which receives data packets from circuit emulator **12** over parallel port **72** and stores the received data packet into shared buffer **130**. Thread **126** executes a "W32N_PacketSend" routine **76**, which polls shared buffer **130** for data packets, repackages the data packets and sends them into network **24** over Ethernet NIC **74**.

[**0030**] Because circuit emulator **12** typically runs at a speed much slower than devices on network **24**, an alternative embodiment combines threads **124** and **126** and eliminates shared buffer **130**, taking advantage that W32N_PacketSend routine **76** can complete repackaging and sending out a data packet to network **24** before arrival of the next data packet from circuit emulator **12** over parallel port **72**.

[**0031**] Mainscreen **80** calls "W32N_MolassesStop" routine **94** to terminate execution of both threads **120**, **122**, **124** and **126**.

[**0032**] The size of each of buffers **60**, **62**, **68** and **70**, **128** and **130** can be changed dynamically. Even then, a buffer overflow condition can occasionally occur, resulting in data packets being discarded. Typically, discarding an incomplete packet risks losing a network connection. However, there is no risk of losing a network connection under the present invention, because only whole packets are discarded.

[**0033**] As mentioned above, the present invention is also applicable to a circuit simulation of the network interface device. FIG. **9** shows a configuration including first workstation **920** and host computer **930**. In this embodiment, a network interface card is simulated in a circuit simulation program running on host computer **930**. Circuit simulation does not operate at the full network speed the network interface card is intended. Host computer **930** runs the interface program "Molasses" concurrently with the circuit simulation program. Host computer **930** connects to conventional computer network **924** (e.g., 10BaseT Ethernet over twisted pair) using a conventional network interface. Workstation **920** communicates with host computer **930** over computer network **924** using conventional network protocols. Host computer **930** can be, for example, a desktop

PC running Windows 95 or Windows 98 and equipped with a 10baseT Ethernet controller card. In one embodiment, host computer **930** includes an Intel Pentium class processor and is equipped with 32 Mbytes of DRAM and 500 Mbytes of hard disk space. Host computer **930** also includes Media Access Controller ("MAC") drivers, parallel port drivers and the NDIS API (Network Driver Interface Specification—Application Program Interface). MAC drivers interface the operating system to the Ethernet card hardware. The NDIS API allows functional enhancements to network drivers in the operating system. A programming language interface (PLI) supported by the circuit simulation program allows the circuit simulation program to interact with the Molasses software. Workstation **920** can each be any conventional workstation, including a PC running the Windows 98 operating system.

[**0034**] FIG. **7** is a block diagram **700** showing the functions performed by Molasses program **750**, in accordance with a third embodiment of the present invention that can be implemented in the configuration of FIG. **9** Molasses program **750** can include a graphical user interface similar to that illustrated in FIG. **4** by Mainscreen **80**. As in Molasses program **50**, Molasses program **750** interfaces with network interface card (NIC) **74**, which provides host computer **930**'s access to computer network **24**, and simulation interface **772**, which provides a program interface in host computer **930** to a circuit simulation running on host computer **930** concurrently with Molasses program **50**. In this embodiment, simulation interface **772** can be a conventional programming language interface (PLI), an interprocess communication mechanism (e.g., a socket), a client-server type interface, or any other suitable software interface. For example, hardware description languages, such as Verilog or VHDL, provide support for programming interfaces. In particular, under Verilog, which is standardized by the Institute of Electrical and Electronics Engineers (IEEE), defines a PLI interface through which a compiled C Language program can communicate with a running circuit simulation.

[**0035**] Once the parameters of Mainscreen **80** are set, Mainscreen **80** calls "W32N_MolassesStart" routine **752**. Routine **752** creates simultaneous threads running "W32N_MolassesBuffer" routine **754** and "SIM_MolassesBuffer" routine **756**, respectively. As in W32N_MolassesBuffer routine **54** discussed above, W32N_MolassesBuffer routine **754** receives packets from the Ethernet NIC **74** (via the same W32N_PacketRead routine **58** discussed above), stores the received packets into receive buffer **60** in host computer **930**'s main memory. Subsequently, the received packets in buffer **60** are transferred to transmit buffer **62**, from which they are then transmitted to simulation PLI **772** via "SIM_PacketSend" routine **764**. SIM_MolassesBuffer routine **756** receives packets from simulation PLI **772** (via "SIM_PacketRead" routine **766**), stores the packets into receive buffer **68**. Subsequently, routine **56** then transfers the data packets in buffer **68** to a transmit buffer **70**, which are then transmitted to the Ethernet NIC **74** via "W32N_PacketSend" routine **76**. As in Molasses program **50** above, Molasses program **750** converts data packet formats, when necessary. For example, the preamble that is used in a packet for synchronizing the clock signals of the network and the emulated device is removed before being forwarded to the circuit simulation over simulation PLI **772**. Mainscreen **80** calls "W32N_MolassesStop" routine **794** to terminate execution of both threads **754** and **756**.

[**0036**] FIG. **8** is a block diagram **800** showing the functions performed by Molasses program **840**, in accordance with a fourth embodiment of the present invention that can also be implemented in the configuration of FIG. **9**. As in Molasses program **750** of FIG. **7**, Molasses program **840** of FIG. **8** interfaces with network interface **74**, which provides host computer **930**'s access to computer network **24**, and simulation PLI **772**, which couples host computer **930** to the running circuit simulation. Once the parameters of Mainscreen **80** (FIG. **4**) are set, Mainscreen **80** calls "W32N_MolassesStart" routine **852**, which creates four threads **820**, **822**, **824** and **826**. Thread **820** executes "W32_PacketRead" routine **58**, which receives data packets from Ethernet NIC **74** and stores the received data packet into shared buffer **128** in the main memory of the host computer **930**. Thread **822** executes "SIM_PacketSend" routine **864**, which polls shared buffer **128** for the received data packets, repackages these data packets and sends them to the running circuit simulation over simulation PLI **772**. Thread **824** executes "SIM_PacketRead" routine **866**, which receives data packets from the running circuit simulation over simulation PLI **772** and stores the received data packet into shared buffer **130**. Thread **826** executes a "W32N_PacketSend" routine **76**, which polls shared buffer **130** for data packets, repackages the data packets and sends them into network **24** over Ethernet NIC **74**.

[**0037**] As in circuit emulator **12**, because a circuit simulation typically runs at a speed much slower than devices on network **24**, an alternative embodiment combines threads **824** and **826** and eliminates shared buffer **130**, taking advantage that W32N_PacketSend routine **76** can complete repackaging and sending out a data packet to network **24** before arrival of the next data packet from the circuit simulation emulator **12** over simulation PLI **772**.

[**0038**] Mainscreen **80** calls "W32N_MolassesStop" routine **894** to terminate execution of threads **820**, **822**, **824** and **826**.

[**0039**] Molasses programs **50**, **40**, **750** and **840** each include a test program for self-test. FIG. **5** shows a test setup suitable for use with the self-test involving PCs **501**, **502** and **503**. For brevity, this test program is described with respect to Molasses program **50**. Description herein regarding Molasses program **50** is equally applicable to Molasses program **40**. The test program has two modes of operation—"initiate" and "respond". PC **502** runs Molasses program **50**, configured to address two Ethernet network interface cards, rather than the bi-directional interface, as in FIG. **1**. PC **501** runs the test software in initiate mode, generating and sending Ethernet packets of varying size to PC **502** via local area network **504** (e.g., Ethernet with 10BaseT connections) at step **521**. At step **522**, Molasses program **50** of PC **502** receives the packets from PC **501** using one of its two network interface cards, and then forwards the received packets to PC **503** over network **505**, which is coupled to the other one of its network interface cards. PC **503** runs the test software in respond mode, taking each packet received from PC **502** (step **523**) and re-transmitting it back to PC **502** over local area network **505** (step **524**). At step **525**, Molasses software of PC **502** then forwards the received packet from

PC **503** to PC **501** over local area network **504**. There, at step **526**, under initiate mode, the test software on PC **501** compares the returned packet to the packet it transmitted at step **521**. Any mismatch of these packets is reported as an error. In one embodiment, a timer can be set in 1/18-second increments to specify the frequency of packet generation in PC **102**.

[**0040**] A user interface is provided by the test program to self-test Molasses program **50**. The user interface displays an appropriate amount of information, based on a user's selection of silent mode or verbose mode. Through this user interface, a user can vary a packet throughput rate, effectuate an overflow condition in any of buffers **60**, **62**, **68**, **70**, **128** and **130**, or test for timing and throughput problems. A status line is provided in the user interface to continuously update information about packets processed by Molasses program **50**, such as the number of packets sent and received, the current value of the timer, an error count, and other status information desired. In addition, a log file can be specified to record the value of each byte of each packet, errors that occurred, and comments that the user may wish to add using a comment line in the user interface. The recorded information may be used for future reference and debugging.

[**0041**] The test program also provides a test for accessing circuit emulator **12** through a bidirectional interface (e.g., a parallel port). In one embodiment, an industry standard parallel port conforming to the enhanced parallel port (EPP) standard is provided. The test program allows a user to read and write 8-bit addresses and 8-bit data patterns via the parallel port to circuit emulator **12**. In one test, data is continuously written to and read back from circuit emulator **12** and compared. Any mismatch between the written data and the read back data is reported as an error.

[**0042**] Various modifications and adaptations of the operations described here would be apparent to those skilled in the art based on the above disclosure. Many variations and modifications within the scope of the present invention are therefore possible. The present invention is set forth by the following claims.

1-24. (canceled)

25. A method for simulating an electronic device that interacts with a network, the simulation being carried out by a program executing in a host computer, the simulation includes simulating the electronic device's interaction with the network, the method comprising executing instructions on the host computer for:

(a) receiving data packets designating the electronic device as recipient from the network through a network interface; and

(b) transmitting the data packets to the simulation through a software interface to provide data packets for simulating the electronic device's interaction with the network.

26. The method of claim 25, the instructions further comprising instructions for storing the data packets received from the network in a buffer allocated in the memory of the host computer.

27. The method of claim 26, the instructions further comprising instructions for changing the size of the buffer at run time.

28. The method of claim 26, the instructions further comprising instructions to cause to be discarded packets of data when the buffer is full.

29. The method of claim 26, the instructions further comprising instructions for keeping a record of the data packets received from the network.

30. The method of claim 29, the instructions further comprising instructions for displaying the record.

31. The method of claim 29, the instructions further comprising instructions for storing the record in a file.

32. The method of claim 25, the instructions further comprising instructions for recording the throughput of the data packets.

33. The method of claim 25, the instructions further comprising instructions for modifying the packets to make the packets suitable for receipt by the simulation.

34. The method of claim 33, wherein modifying includes removing a preamble from a data packet.

35. The method of claim 25, wherein the instructions for receiving data packets from the network, and the instructions for transmitting the data packets to the simulation are executed in a single thread.

36. The method of claim 25, wherein the instructions for receiving data packets from the network is executed in a first thread, and the instructions for transmitting the data packets to the simulation is executed in a second thread.

37. A method for simulating an electronic device that interacts with a network, the simulation of the electronic device being carried out by a program executing in a host computer, the simulation including simulating the electronic device's interaction with the network, the method comprising executing on the host computer instructions for:

(a) receiving data packets designating the electronic device as a source through a software interface from the simulation of the electronic device's interaction with the network; and

(b) transmitting the data packets to the network through a network interface.

38. The method of claim 37, the instructions further comprising instructions for storing the data packets received from the simulation in a buffer allocated in the memory of the host computer.

39. The method of claim 38, the instructions further comprising instructions for changing the size of the buffer at run time.

40. The method of claim 38, the instructions further comprising instructions to cause to be discarded packets of data when the buffer is full.

41. The method of claim 37, the instructions further comprising instruction for keeping a record of the data packets received from the simulation.

42. The method of claim 41, the instructions further comprising instructions for displaying the record.

43. The method of claim 41, the instructions further comprising instructions for storing the record in a file.

44. The method of claim 37, the instructions further comprising instructions for recording the throughput of the data packets.

45. The method of claim 37, the instructions further comprising instructions for modifying the data packets to make the packets suitable for receipt by the network.

46. The method of claim 45, wherein modifying includes inserting a preamble in a data packet.

47. The method of claim 37, wherein the instructions for receiving data packets from the simulation and the instructions for transmitting the data packets received from the simulation to the network are executed in a single thread.

48. The method of claim 37, wherein the receiving data packets from the simulation is executed in a first thread and the transmitting the data packets to the network is executed in a second thread.

49. A computer-readable medium for use in a simulation of an electronic device, wherein the simulation is to be carried out by a program executing in a host computer, and wherein the simulation of the electronic device includes simulating the electronic device's interaction with the network; the computer-readable medium comprising computer-executable instructions to be executed on the host computer for:

(a) receiving data packets designating the electronic device as a recipient from the network through a network interface; and

(b) transmitting the data packets to the simulation through a software interface to provide data packets for simulating the electronic device's interaction with the network.

50. The computer-readable medium of claim 49, further comprising computer instructions for storing the data packets received from the network in a buffer allocated in the memory of the host computer.

51. The computer-readable medium of claim 50, further comprising computer-executable instructions for changing the size of the buffer at run time.

52. The computer-readable of medium claim 50, further comprising computer-executable instructions for discarding packets of data when the buffer is full.

53. The computer-readable medium of claim 49, further comprising computer instructions for keeping a record of the data packets.

54. The computer-readable medium of claim 53, further comprising computer-executable instructions for displaying the record.

55. The computer-readable medium of claim 53, further comprising computer-executable instructions for storing the record in the storage medium.

56. The computer-readable medium of claim 49, further comprising computer-executable instructions for recording the throughput of the data packets.

57. The computer-readable medium of claim 49, further comprising computer-executable instructions for modifying the data packets for receipt by the simulation.

58. The computer-readable medium of claim 57, wherein the computer-executable instructions for modifying includes computer-executable instructions for removing a preamble from a data packet.

59. A computer-readable medium for use in a simulation of an electronic device, wherein the simulation is to be carried out by a program executing in a host computer, and wherein the simulation of the electronic device includes simulating the electronic device's interaction with the network; the computer-readable medium comprising computer-executable instructions to be executed on the host computer for:

(a) receiving data packets designating the electronic device as a source through a software interface from the simulation of the electronic device's interaction with the network; and

(b) transmitting the data packets received from the simulation to the network through a network interface.

60. The computer-readable medium of claim 59, further comprising computer-executable instructions for storing the data packets received from the simulation in a buffer allocated in the memory of the host computer.

61. The computer-readable medium of claim 60, further comprising computer-executable instructions for changing the size of the buffer at run time.

62. The computer-readable medium of claim 60, further comprising computer-executable instructions for discarding packets of data when the buffer is full.

63. The computer-readable medium of claim 59, further comprising computer-executable instructions for keeping a record of the data packets.

64. The computer-readable medium of claim 63, further comprising computer-executable instructions for displaying the record.

65. The computer-readable medium of claim 63, further comprising computer-executable instructions for storing the record in the storage medium.

66. The computer-readable medium of claim 59, further comprising computer-executable instructions for recording the throughput of the data packets.

67. The computer-readable medium of claim 59, further comprising computer-executable instructions for modifying the data packets to make the packets suitable for receipt by the network.

68. The computer-readable medium of claim 67, wherein computer-executable instructions for modifying includes computer-executable instructions for inserting a preamble in a data packet.

* * * * *