(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 November 2001 (08.11.2001)

PCT

(10) International Publication Number
WO 01/84272 A2

(51) International Patent Classification[7]: G06F

(21) International Application Number: PCT/US01/13350

(22) International Filing Date: 25 April 2001 (25.04.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/560,436 28 April 2000 (28.04.2000) US

(71) Applicant: **PEAK AUDIO, INC.** [US/US]; Suite 414, 1790 30th Street, Boulder, CO 80301 (US).

(72) Inventors: **ANDERSON, Charles, William**; 4809 Sugarload Road, Boulder, CO 80301 (US). **BRITTON, John, Bayard**; 8981 North 55th Street, Longmont, CO 80503 (US). **LIEB, Derek, Wearin**; 4718 Ipswich, Boulder, CO 80301 (US). **GROSS, Kevin, Paul**; 2370 Glenwood Drive, Boulder, CO 80304 (US).

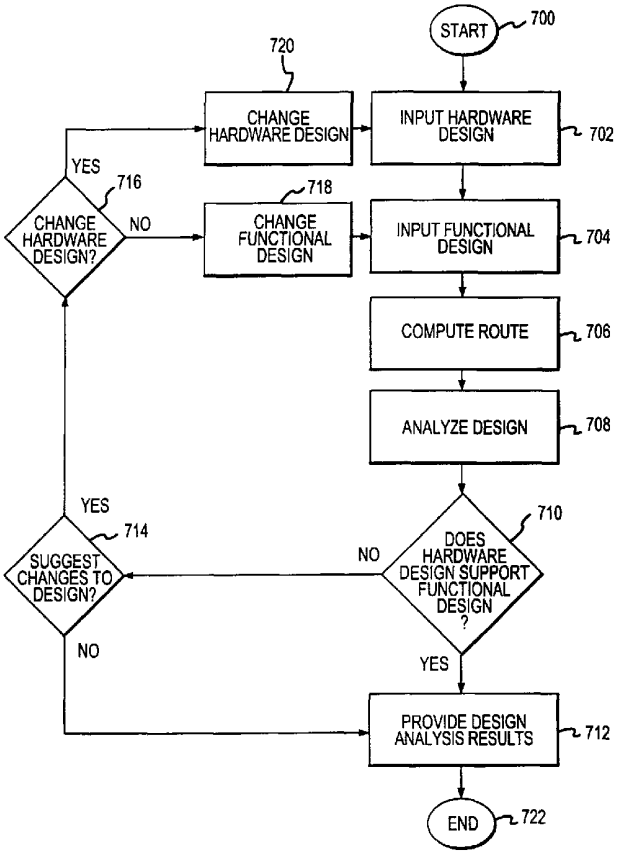
(74) Agent: **KENNEDY, John, T.**; Dorsey & Whitney LLP, Suite 4700, Republic Plaza Building, 370 Seventeenth Street, Denver, CO 80202-5647 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE,

[Continued on next page]

(54) Title: SYSTEM AND METHOD FOR CHECKING A PHYSICAL NETWORK DESIGN AGAINST A FUNCTIONAL CONNECTION OF DATA STREAMS



(57) Abstract: In a preferred embodiment, the present invention provides a method and a system for designing and verifying a physical network configuration (Fig. 3B) is capable of supporting a functional network configuration (Fig. 4B) without having to actually establish such networks. The present invention allows a user to input a hardware network design and a functional network design into a computer system (900). Utilizing the functional network design, the computer system then models the route taken by the data to travel from a first transceiver to a second transceiver over the network, while also determining those resource requirements, such as bandwidth, needed to support such data flow through the hardware network design. Upon determining routes for each connection specified in the functional network design, and determining resource requirements associated therewith at each hardware devices effected, the present invention analyzes the maximum resource capabilities of each hardware device with respect to those resources needed at such device and determines whether such hardware configuration can support the functional design. In a second embodiment, the present invention recommends changes to the hardware necessary to support the functional design and vice versa. In a third embodiment, a computer system (1600) implementing the present invention configures the various network devices specified in the hardware network design upon verifying the hardware design supports the functional design.

WO 01/84272 A2

IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii)) for the following designationsAE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG)*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii)) for all designations*

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

SYSTEM AND METHOD FOR CHECKING A PHYSICAL NETWORK DESIGN AGAINST A FUNCTIONAL CONNECTION OF DATA STREAMS

FIELD OF THE INVENTION

The present invention relates to the field of data network design and verifying whether a specified hardware configuration will support a functional description of a network. More specifically, the present invention provides a method and system for determining whether a hardware description of an Ethernet network utilized to distribute audio, video or control data streams can support a functional description of a network.

BACKGROUND OF THE INVENTION

As multimedia presentation systems have become more commonplace in arenas, theaters, board rooms, and even homes, a need has arisen for the deployment of computing systems, networks, and various other data processing systems which reliably transmit data to numerous locations. Often the data desired to be transmitted is both audio, video or control data which may be obtained from numerous sources and is often characterized as being isochronous (i.e., the data is clocked, continuous, and delivered with a determinant latency). For example, a presentation of a football game might include multiple camera angles, and audio data from microphones on announcers, coaches, and the field. The audio data might be presented at many hundred speakers distributed throughout the stands, luxury suites, coaches boxes, and press areas in the stadium. Similarly, the video data might be distributed to a comparable number of display devices. Statistics, public address information, and other textual data might also be presented on multiple devices, while control data for the various systems might also be needed. As such, a network engineer might be

faced with the task of determining the data routing for hundreds of input devices to thousands of output devices. Currently, planning a network to address these numerous data needs is quite time consuming and labor intensive.

Further compounding the obstacles faced by network system designers is the fact that numerous networks and other mechanisms are currently available for transmitting data. One such network configuration is a Token Ring which provides a 4/16 Mbit multiple access network. A Token Ring network commonly utilizes a ring topology to interconnect end stations, as shown in **Figure 1A**. The features and limitations of Token Ring networks are well known in the art.

Another network which utilizes a configuration similar to a Token Ring, is a Fiber Digital Data Interface (FDDI). A FDDI network also utilizes a token passing scheme to control network access. However, a FDDI network uses intelligent hubs (which are often quite expensive) to overcome many of the shortcomings of a Token Ring Network and is physically wired in a star configuration using a central concentrator to improve the network's reliability. FDDI networks are also commonly known in the art and, for purposes of the present invention, are not discussed in detail herein.

A third network configuration is an ATM network, as shown in **Figure 1B**. An ATM network generally relies on complex and expensive central switches to route data between various destinations. ATM networks are also commonly known in the art. Additionally, various other network types may be utilized including Localtalk, High Performance Parallel Interface (HPPI), Synchronous Optical Network (SONET), Small Computer System Interface (SCSI), Fibre channel, IEEE-1394 (a.k.a. FireWire), RS-485, Universal Serial Bus (USB), and Arcnet. The

features and functions of these networks are also well know in the art and are not the subject of further discussion herein.

In addition to the before mentioned network technologies, perhaps the most popular network technology is Ethernet. As is commonly known, Ethernet networks are defined by the Institute of Electrical and Electronic Engineers (IEEE) Standard 802.3. This standard defines the rules for configuring Ethernet networks as well as the protocol that allows the various network devices to communicate.

Network protocols provide those standards which allow computers to communicate with each other. It is commonly appreciated that a network may be configured to support various other protocols, including, but not limited to, IPX, TCP/IP, DECnet, AppleTalk, LAT, SMB, DLC, and NetBEUI. Thus, a systems engineer is commonly faced with deciding which network topology to utilize, which protocol(s) to use and, as is discussed further herein, which system components and wiring or wireless connections to utilize in configuring a network. As is generally appreciated, designing a network for a large system is often a daunting task.

Additionally, each type of network has numerous valid configurations. For example, an Ethernet network might be configured in a repeater hub configuration, as shown in **Figure 1C**, or in a switched network, as shown in **Figure 1D**. Additionally, multiple Ethernet networks might be connected to each other via a bridge or a repeater, while a router might divide a given network into various sub-nets such that only traffic designated for particular devices can pass between segments.

In addition to determining which type of network to utilize and which network protocol to use, the network systems engineer also has to decide upon a cabling medium and wiring topology. For example, for an Ethernet network alone, numerous types of cabling are available including, but

not limited to: 10BASE-5 (or "Thickwire"); 10BASE-2 (or "Thinwire"); 10BASE-T (or Unshielded Twisted Pair (UTP)) which itself has rating categories 1-5; 100BASE-T (or "Fast Ethernet") which further includes 100BASE-TX (for category 5 UTP cable), 100BASE-FX (for fiber-optic cable), 100BASE-T4 (which uses category 3 UTP cable with two extra wires); and Gigabit Ethernet cabling, which is available in both copper and fiber-optic cabling. Similarly, numerous wireless configurations are available, all of which are commonly known in the art.

Further complicating the network system designers task is the fact that many networks, and especially an Ethernet network, may utilize various components manufactured by various companies - each of which has unique capabilities, specifications, and configuration information. For example, a short listing of manufacturers of Ethernet switches might include the following manufacturers: 3COM, CISCO, Hewlett Packard, Extreme Networks, Fore Systems, and Linksys. Each of these manufacturers often offer numerous types of Ethernet switches, with varying features, functions, and prices.

In addition to considering a network's protocols, cabling, topology, and devices, the network designer must also consider other "resources" associated with a network including, but not limited to, bandwidth, latency (i.e., will the data arrive at the desired time at each location), hop count (i.e., how many times data passes through devices), and jitter (i.e., variations in clocking data - which, for example, is essential to synchronizing video and audio). Those skilled in the art readily appreciate the various resources which may be provided by network devices and needed for a network to operate as desired.

Once a network system designer has selected all the various features, functions, and components for a network, a system is needed which allows the designer to verify the design before

the network is actually constructed. Currently, there are no automated systems available which enable a network system designer to verify a network's hardware design based upon a functional design.

Therefore, a system is needed which allows a network system designer to input a desired hardware configuration and a functional configuration into a computer, which then determines whether the configurations will function as desired. Additionally, a system which recommends changes, additions, and/or deletions to system components, is needed, such that a system designer would not have to perform numerous iterations in obtaining a hardware design which supports a given functional description and vice versa.

Additionally, networked systems are often utilized in applications where network system engineers are not readily accessible including, for example, office configurations of computer equipment, the interconnection of audio and video equipment in corporate boardrooms, video conferencing applications, paging systems, lighting systems, and even systems used in homes (for example, intercoms and audio distribution systems). Users of such systems often do not possess the technical skills necessary to add devices to networks or to reconfigure a network. Thus, a system is needed which allows a person not skilled in network system design to specify a functional description of a desired data distribution system and in response be provided with a hardware configuration which is capable of implementing the desired functional system.

Additionally, after a network system has been designed and connected, network system engineers commonly spend significant amounts of time configuring each device in a network. As can be appreciated, configuring a network distributed across large geographic expanses (for example, a multi-national video conferencing network) is a daunting task since each device may

have to be uniquely configured in the system. Therefore, a system is needed which automatically configures the various devices utilized in a network.

SUMMARY OF THE INVENTION

The present invention provides a method and system for inputting and verifying a hardware network design against a functional network design to determine whether the hardware network design is capable of supporting the functional network design. More specifically, the present invention allows a user of a computer application to input a hardware design and a functional design for a network, analyze the hardware design relative to the requirements of the functional design, and determine whether the hardware design is capable of supporting the functional design.

In a second embodiment, the present invention also receives the results of the analysis of the hardware design relative to the requirements of the functional design and recommends changes to either the hardware and/or the functional design, as necessary.

In a third embodiment, the present invention enables a user to input a functional design into a system implementing the present invention. The system then utilizes the functional design to recommend a hardware design which supports the functional design. Additionally, in this embodiment, the present invention may also recommend specific devices, cabling, protocols, and other components necessary to implement the desired functional design.

In a fourth embodiment, a system implementing the present invention is suitably connected into a physical embodiment of a desired hardware and functional design. Upon user direction, the system automatically configures each of the various devices provided in the network and verifies the physical network connections actually exist and operate as provided for in the functional design.

The aforementioned embodiments, features and functions of the process and system of the present invention are more fully disclosed with reference to the drawing figures, the following detailed description, and the claims.

BRIEF DESCRIPTION OF THE DRAWING FIGURES

Figure 1A is a schematic representation of a prior art Token Ring network;

Figure 1B is a schematic representation of a prior art ATM network;

Figure 1C is a schematic representation of a prior art Ethernet network which utilizes a repeater hub configuration;

Figure 1D is a schematic representation of a prior art Ethernet network which utilizes a switched configuration;

Figure 2 is a flow chart depicting the process by which a hardware network design and a functional network design are verified by a preferred embodiment of the present invention;

Figure 3A is a flow chart depicting the process by which a user inputs a hardware network design for the preferred embodiment of the present invention;

Figure 3B is a representative schematic diagram of a hardware network design which has been input into a system implementing the process shown in **Figure 3A** for a preferred embodiment of the present invention;

Figure 4A is a flow chart depicting the process by which a user inputs a functional network design for the preferred embodiment of the present invention;

Figure 4B is a representative schematic diagram of a functional network design which has been input into a system implementing the process shown in **Figure 4A** for a preferred embodiment of the present invention;

Figure 5A is a flow chart depicting the process by which the hardware network design is verified in view of the functional network design for a preferred embodiment of the present invention;

Figure 5B is a representation of a connection table data array generated by the process of the present invention;

Figure 5C is a representation of a node results table generated by the process of the present invention for the example presented herein after the bandwidth resources needed for the first connection have been determined;

Figure 5D is a representation of an edge results table generated by the process of the present invention for the example presented herein after the bandwidth resources needed for the first connection have been determined;

Figure 5E is a representation of a node results table generated by the process of the present invention for the example presented herein after the bandwidth resources needed for the second connection have been determined;

Figure 5F is a representation of an edge results table generated by the process of the present invention for the example presented herein after the bandwidth resources needed for the second connection have been determined;

Figure 5G is a representation of a node results table generated by the process of the present invention for the example presented herein after the bandwidth resources needed for all the connections have been determined;

Figure 5H is a representation of an edge results table generated by the process of the present invention for the example presented herein after the bandwidth resources needed for all the connections have been determined;

Figure 6 is schematic representation of an output result of the process of verifying a hardware network design in view of a functional network design for a preferred embodiment of the present invention;

Figure 7 is a flow chart depicting the process by which the present invention, upon receiving a hardware network design and a functional network design, verifies a design and recommends changes to the design, as necessary, for a second embodiment of the present invention;

Figure 8 is a flow chart depicting the process by which the present invention upon receiving a functional network design determines and suggests the various components necessary in a hardware network design to accomplish the functional network design for a third embodiment of the present invention;

Figure 9 is a schematic representation of a preferred embodiment of a system for implementing the present invention;

Figure 10A is a screen shot from a preferred embodiment of the system of the present invention wherein drop down menus for selecting hardware devices are displayed;

Figure 10B is a screen shot from a preferred embodiment of the system of the present invention wherein switches have been entered into a hardware design and drop down menus for selecting transceivers are displayed;

Figure 10C is a screen shot from a preferred embodiment of the system of the present invention showing switches and transceivers which have been input into a hardware design for the illustrative example discussed herein;

Figure 10D is a screen shot from a preferred embodiment of the system of the present invention showing the various hardware devices interconnected via cabling at specific device ports for the illustrative example discussed herein;

Figure 10E is a screen shot from a preferred embodiment of the system of the present invention showing the various hardware devices interconnected via cabling at specific device ports and the data information provided by the present invention for the illustrative example discussed herein;

Figure 10F is a screen shot from a preferred embodiment of the system of the present invention showing the hardware design constraints provided by the present invention which prohibit a user from designing a system with mismatched data ports for the illustrative example discussed herein;

Figure 11A is a screen shot from a preferred embodiment of the system of the present invention showing an inputted functional design for the illustrative example discussed herein;

Figure 11B is a screen shot from a preferred embodiment of the system of the present invention showing various functional connections between various devices for the illustrative example discussed herein;

Figure 12 is a screen shot from a preferred embodiment of the system of the present invention after an analysis of the hardware design has been accomplished in view of the functional design and showing that the hardware design supports the functional design;

Figure 13 is a screen shot from a preferred embodiment of the system of the present invention after an analysis of the hardware design has been accomplished in view of the functional design and showing that the hardware design does not support the functional design because a cable connection is overloaded;

Figure 14A is a screen shot from a second embodiment of the system of the present invention after an analysis of the hardware design has been accomplished in view of the functional design and showing that the hardware design does not support the functional design because a connection between two devices has not been provided;

Figure 14B is a screen shot from a second embodiment of the system of the present invention after an analysis of the hardware design has been accomplished in view of the functional design and showing that the hardware design does not support the functional design because a connection between two devices has not been provided and providing a hardware solution to the identified design defect;

Figure 14C is a screen shot from a second embodiment of the system of the present invention after an analysis of the hardware design has been accomplished in view of the functional design and showing that the hardware design does not support the functional design because a connection between two devices has not been provided and providing a functional solution to the identified design defect;

Figure 15A is a screen shot from a third embodiment of the present invention showing the display of functional devices containing functional data ports;

Figure 15B is a screen shot from a third embodiment of the present invention showing the functional interconnection of data ports to functional network ports, as specified by a user;

Figure 15C is a screen shot from a third embodiment of the present invention showing the functional interconnection of functional network ports, as specified by a user;

Figure 15D is a schematic representation of an intermediate processing step utilized by the third embodiment of the present invention to generate a hardware network design based upon an inputted functional network design showing the interconnection of hardware transceivers with switches;

Figure 15E is a schematic representation of the result of an intermediate processing step utilized by the third embodiment of the present invention to reduce the number of switches utilized in a hardware design supporting an inputted functional design;

Figure 15F is a schematic representation of a result of the processing provided in a third embodiment of the present invention wherein a hardware design is specified which supports an inputted functional design; and

Figure 16 is a schematic representation of a fourth embodiment of the present invention in which a system for automatically configuring network devices utilized in a hardware design is presented.

DETAILED DESCRIPTION OF THE INVENTION

In a preferred embodiment, the present invention provides a process and system for inputting a hardware design and a functional design and verifying the hardware design is capable supporting the functional design. For purposes of simplicity, the process and systems by which the present invention accomplishes these objectives is discussed with reference to an Ethernet network configuration. However, it is to be appreciated any network configuration may be utilized by the present invention in lieu of or in cooperation with an Ethernet network. Additionally, those skilled in the art appreciate that the rules and/or protocols utilized by the present invention to design and verify Ethernet configurations may be modified, as necessary, to configure other devices which are governed by other protocols, rules, or guidelines.

The process by which the present invention, in a preferred embodiment, receives a hardware design and a functional design and determines whether such designs are operable and compatible is shown in **Figure 2**. The process preferably begins (Block 200) with a user inputting a hardware design into a system implementing the process of the present invention (Block 202). Once the hardware design has been entered, the process continues with the user inputting a functional design (Block 204). The processes by which the user inputs the hardware design and the functional design are explained below in greater detail with reference to **Figures 3A** and **3B**, and **Figures 4A** and **4B**, respectively. Utilizing the hardware design and the functional design, the process then analyzes the hardware design in view of the functional design to determine whether the hardware design can support the functional design while also verifying the various data routes from any first network device to a second network device are supported (Block 206). The results of this

functional and hardware analysis are provided preferably on a computer monitor and/or hard copy output to the user (Block 208).

The before mentioned process is preferably implemented on a computer workstation 900, as shown in **Figure 9**. The computer workstation 900 preferably contains a data input device 902 such as a keyboard, voice recognition software module, mouse, or other similar devices. The computer workstation 900 also includes a display monitor 904, and is connected to an output device, such as a printer 906. The computer workstation 900 may also be connected via a suitable communications link 908 to a server 910 having access to a database 912 from which information on devices, cabling, protocols, network configurations, design rules and other information may be obtained. The communications link 908 may be provided via the Internet, dial-up connections, other network configurations (for example, a computer LAN or WAN), or any other connection scheme between a computer workstation and a database of information. As is commonly known, the database 912 may also be provided within the computer workstation 900, for example, in a hard drive, RAM, ROM, compact disc, digital versatile disc, or similar device.

Additionally, the computer workstation 900 contains those data storage, manipulation, and control features commonly provided by such devices. The computer workstation 900 preferably utilizes a Pentium III®, but any controller providing the processing speed and capabilities necessary to run a software program implementing the process of the present invention may be utilized. Additionally, throughout this discussion of the process of the present invention, reference is made to the system shown in **Figure 9** and screen displays provided thereby. It is to be appreciated, however, that the process of the present invention is not limited to an implementation only on the system shown in **Figure 9** nor the various screen displays shown in **Figures 10-13**. As such, the

present invention should not be construed as being limited to any specific system configuration or screen displays. Any system which supports the processes, features, and functions of the present invention may be utilized.

With reference to **Figure 3A**, the process for the preferred embodiment by which a user inputs a hardware design is shown. First, the user selects the type of network (Block 302). For the preferred embodiment, an Ethernet network is selected; however, any network configuration may be selected for the present invention. The process of the present invention may be suitably modified to support various network configurations by appropriately modifying devices, cabling, protocols, resources, and other factors as is necessary based upon design concepts well known in the art.

Once the type of network has been selected, (for purposes of this discussion an Ethernet network is selected) the process of the present invention preferably continues with the user selecting and/or inputting a device for a specific location in the network. One example of such a device may be a datastream transceiver, switch, repeater hub, repeater, and/or bridge. In the preferred embodiment, the user suitably selects, from a drop down menu or other data presentation technique, one of a plurality of devices to be utilized in a specific location. For example, as shown in **Figure 10A**, a user selecting an Ethernet switch is preferably provided on a drop-down menu a listing of the various switches 1002 provided by manufacturers. The user may suitably select any of these devices to use in the hardware network design. While the present invention preferably utilizes drop down menus, it is to be appreciated that input fields and various other known in the art techniques for inputting data into a computer system in order to identify hardware devices provided in a network may also be utilized. For example, the computer workstation 900 may be connected to a pre-existing network. Utilizing processes and programs known in the art, the computer

workstation 900 may then determine the hardware design for the pre-existing network, including the types of devices utilized, the location of devices, and various other network configuration information. Thus, the present invention is not to be construed as being limited to any specific methods or systems for inputting a hardware network design.

Technical specifications for each device provided on the drop down menus are preferably stored in data files associated with the process of the present invention. By double clicking, selecting, highlighting, or using various other techniques known in the art, a user may retrieve technical specifications for any device listed. Similarly, for a device is not provided on the drop down menu, data fields are suitably provided which allow a user to specify the characteristics, as necessary, of the device. Those skilled in the art appreciate the various parameters which may need to be specified for a network device.

Additionally, the process of the present invention preferably allows the user to modify any of the specifications for a device that has been selected or inputted (Block 306). Such configuration changes might be necessary, for example, when a specially configured device is being utilized, when a user modifies a device itself, or when a device provides programmable features which the user specifies based upon system constraints. For example, a user might modify a 3COM SuperStack II Switch 3300® such that the switch utilizes a software patch downloaded from the Internet. In such a situation, the operating specifications for the switch might need to be modified (Block 308) by inputting new parameters and/or downloading new parameters. The present invention supports updates to device characteristics, the inputting of new devices into a device library, and other

operations commonly performed on computer based systems. As is well known, updates to such device files may be accomplished via the Internet, CD-ROMs, and other known techniques.

Figure 10B shows a representative example of a user entering a hardware network design by first entering three switches 1004 into the design view screen 1006. As shown, various transceivers may be connected to an Ethernet switch. Preferably such transceivers are CobraNet® compatible devices and are identified on a drop down menu 1008. However, the present invention is not limited to only using CobraNet transceivers and may be configured to support any hardware network devices.

Upon repeating the process flow of Blocks 306-310 until the various devices desired in a hardware design have been inputted into the hardware design view field 1006, the user obtains a layout of the various hardware devices desired in a specific network, for example the network shown in **Figure 10C**. As shown, the illustrative example provides three switches 1004 and five transceivers 1010, however, it is to be appreciated that various other Ethernet devices, including additional transceivers and switches and other devices such as audio devices, repeater hubs, routers, and bridges may be added to this layout as desired. The present invention is not limited in size or complexity and may support any network design.

After all of the devices have been entered into the network design, the process continues with the user specifying connections between the various network devices (Block 311). As shown in **Figure 10D**, preferably a system implementing the present invention utilizes a graphical user interface which enables a user to establish network connections by merely dragging a mouse from a first port to a second port. However, other techniques such as specifying ports in a table may also be utilized, as desired, by the present invention. Additionally, the present invention allows a user to

specify each connection as being wired or wireless. Wireless network connections are well known in the art, the present invention supports both wired networks, wireless networks, and combinations thereof. Those skilled in the art appreciate the various transmitters, receivers, and other devices commonly utilized in providing a wired and/or a wireless network connection. The present invention may be suitably modified to support any network configuration desired.

As provided in the illustrative example shown in **Figure 3B**, the user preferably configures a network which provides five transceivers: datastream transceiver node 1 (250), datastream transceiver node 2 (252), datastream transceiver node 3 (254), datastream transceiver node 4 (256), datastream transceiver node 5 (258); and three switches: network switch node 6 (260), network switch node 7 (262), and network switch node 8 (264). Each of these nodes (1-8) may be connected in any desired configuration provided for within the Ethernet standards. However, for purposes of the present example, it is assumed that the network hardware design provides the following connections: node 1 (250) to node 6 (260) via cable 266; node 2 (252) to node 6 (260) via cable 268; node 3 (254) to node 7 (262) via cable 270; node 4 (256) to node 8 (264) via cable 272; and node 5 (258) to node 8 (264) via cable 274. When a user is connecting devices, the present invention preferably provides the user with a data port 251 which represents the actual port on a specified transceiver 250 by which hardware devices would actually be connected. Additionally, specifications for each data port 251 are generally provided by the manufacturer of the specific device and are accessible to a user, for example, by right "clicking" on the data port 251. For example, these specifications 1012 might include a description of the type of cabling to be attached thereto, as shown in **Figure 10E**.

As shown in **Figure 10E**, for purposes of the present example, each of the Network switches (i.e., 260, 262, 264) include numerous data ports 1014 which preferably provide 100Mbit/sec connections to transceivers and/or other switches. Additionally, each switch preferably includes a high data rate port 1016 which provides 1 Gbit/sec connections between switches. Preferably, the present invention prevents a user from connecting, for example, 1 Gbit port 1016 with a 100 Mbit port 1014, as shown by the attempted connection 1018 in **Figure 10F**. As such, the present invention provides configuration expertise which allows persons other than network designers to configure a hardware system. Such users might include a school custodian who is configuring an Ethernet based public address system and is not a skilled network engineer. Similarly, the present invention may be suitably modified such that extremely low-tech individuals may access the feature and functions provided herein and design a networked system.

Referring again to **Figure 10F**, for purposes of the present example, the user designing the networked system is assumed to have selected 100BASE-T cabling to connect the various nodes. The user however, could have selected any cabling type supported by the appropriate devices, including for example 1 Gbit cabling between the switch nodes 260/262/264. By identifying specific devices and providing connections between ports on such devices, the present invention simplifies many of these design decisions by automatically identifying the recommended type of cabling based upon manufacturer specifications.

Additionally, it is well known in the art that various types of cables may be utilized to connect various devices within a network. Many of these various cabling types have unique data transfer capabilities and distance requirements which may dictate whether or not a particular cable is recommended to connect any two nodes within a network. As mentioned previously, the process

preferably selects compatible cabling types whenever possible. However, the present invention also allows a user to modify cabling types. Such modifications may be entered manually or might be selected from a drop-down menu (Block 312). Preferably, a user will select a cabling type which is supported by the devices to which specific cabling is to be connected. However, the present invention is not to be construed as preventing a user from using a cable which is not identified by a specific device or may be new on the market.

Once the cabling between the various network devices has been specified (or, in the case of a wireless network, the wireless connections), the user is then queried as to whether the user desires to specify a distance between connections (Block 314 -316). These distance specifications may be important when a user desires to run a cable a length greater than one which is supported by a cable. For example, a user may find it necessary to run a cable two miles from one office location to another. As is known in the art, some cables are capable of accurately transmitting data only up to distances of 100 meters without requiring additional signal processing. The process of the present invention preferably checks cable lengths to ensure the cable can support the desired length (Block 316). When the cable length is not supportable by the current network hardware design, the process preferably prompts the user to either modify the cable distance (Block 317), the cable type (Block 319), or the hardware configuration, for example, by adding devices to the network design (Block 321).

The process by which the user inputs the hardware design continues after all the cabling has been verified. At this point, a query is issued to the user as to whether any more devices need to be connected to each other (Block 320). If additional devices need to be connected within the

network, the process continues with modifying the cabling, as necessary (Block 312). When all devices have been connected, the network hardware design is then preferably saved.

Once the user has finished inputting a hardware design, the user preferably inputs a functional design for the network (as shown in **Figure 2**, Block 204). As shown in **Figure 4A**, the process allows a user to input a functional design based upon the previously entered hardware design by presenting those devices (in this example, transceivers) which input and output the appropriate data to other non-network devices (for example, microphones, speakers, and display monitors). As shown in **Figure 11A**, the process for the present example displays nodes 1-5 (250, 252, 254, 256, and 258, respectively), each of which are transceivers. The process preferably does not display those network switches or similar devices which merely route data.

Once the previously input hardware design is presented in the functional design view field 1102, as shown in the View 2 screen 1100, the process preferably continues with the user selecting a first device from which data will be transmitted, i.e., a source (Block 402, **Figure 4A**), and a second device which will receive the data, i.e., a sink (Block 404). In the present example, for each device 250 a set of virtual input data ports 1104 and virtual output data ports 1106 are designated. These ports do not actually physically exist within a device since the data between any two devices is actually transmitted via the data ports 251 (as shown in **Figure 10D** and discussed previously). However, for purposes of data flow, the input data ports 1104 and output data ports 1106 provide graphical representations of how a user desires data to actually be transmitted.

Upon selecting a source and a sink, the process continues with the user selecting the type of data to be sent between the source and the sink (Block 406). As is commonly known in the art, various data types may be transmitted by a network including, but not limited to, audio data, video

data, and telemetry data. For example, the user might specify that a 50 Mbit/sec data rate 276 is required between node 1 (250) and node 4 (256) while making a connection between a virtual output data port 1110 on node 1 (250) and a virtual input data port 1112 on node 4 (256). In one embodiment of this process step, the user may specify the maximum data rate for a connection by double "clicking" on the connection and inputting data rate information into an appropriate data field. In addition to specifying a resource such as the data rate, the present invention also suitably allows a user to specify other resource requirements, for example, latency. However, in the preferred embodiment, the resources (for example, the data rate) required between two devices is preferably determined based upon the various input ports selected to be transmitted on a network (for example, whether an audio input port or a video input port is desired to be connected to the network). Additionally, in the preferred embodiment, the user may optionally specify the type of data, for example, MPEG3, and various other data characteristics which are known in the art.

After a source, a sink, a data connection and the resource requirements (inlcuding the maximum data rate) has been provided (by either user entry or based upon port connections), the process continues with querying the user as to whether another functional data connection between two devices is desired to be entered into the functional design (Block 416). When another data connection is to be specified, the user repeats Blocks 402 through 414 in specifying a source, a sink, a data link and resource requirements. **Figure 11B** provides a representation of the various functional connections which are desired for the present example. Similarly, **Figure 11B** provides a screen display of how the various data entries in **Figure 4B** might appear on a display for a system implementing the present invention.

When all functional data connections between the various device have been specified, the process produces a functional design as shown in **Figure 4B** (Block 418). As shown, the network functional design preferably does not include components such as switches which are utilized to transmit and transfer data from a first transceiver to a second transceiver. Instead, the network functional design preferably provides a graphical representation of a source of data, a sink of data, and the resources required (for example, an amount of data) to transfer data from the source to the sink at any one particular instance in time. However, the present invention may be configured to provide the network functional design in a non-graphical manner, for example, by providing a connection table, a data array or information in a similar format.

Additionally, as is commonly known in the art, a data stream transceiver may be both a source for some data and a sink for other data. For example, node 3 (254) is a source of the 50 Mbit/sec data on link 286 sent to node 5 (258) while also being a sink for the 25 Mbits/sec data on link 288 from node 4 (256) and also the 25 Mbit/sec data on link 284 from node 2 (252). The present invention suitably displays these data rates and flow directions by providing designations such as "out" and "in" and/or by utilizing various other data presentation techniques, such as directional arrows and color schemes. Thus, the process of the present invention allows the user to input a hardware design and based upon that hardware design specify a functional interconnection of data between various devices.

Referring again to **Figure 2**, once the user has input the hardware design and the functional design, the process for the preferred embodiment preferably continues with analyzing the hardware design in view of the functional design. In the preferred embodiment, this process step is accomplished after the complete functional design has been input into the system. However, it is to

be appreciated that design checks may be accomplished as the user specifies either or both the hardware design and the functional design. In large scale projects, such real-time verifications may be highly beneficial in locating network deficiencies before a network is completely designed. When determining whether a hardware design supports a functional design, the present invention basically verifies the hardware design is capable of support the various interconnections and data rate specified in the functional design, in addition to verifying the hardware design complies with the various networking configuration rules and protocols. In the preferred embodiment, the present invention assumes that devices selected from drop down menus, and interconnections there between, are compatible in terms of latency, hop count, jitter, and various other network parameters. However, the present invention may be suitably modified to verify latency and other variables, as desired.

Figure 5A provides a flowchart representative of the process by which the present invention verifies a hardware design in view of the functional design. The process preferably begins with the generation of a connection table based upon the functional design (Block 502). **Figure 5B** provides an example of a connection table 500 for the network functional design shown in **Figure 4B** for the bandwidth resource. For purposes of the present example, one resource, bandwidth, associated with a network is analyzed. It is to be appreciated, however, that various other resources, including latency, hop count and jitter may also be similarly analyzed by the process of the present invention.

The connection table 500 provides four columns: a connection column 501, a source node column 503, a sink node column 505, and a bandwidth column 507. The connection node column 501 identifies the various source to sink connections shown in **Figure 4B**. For example, the first

connection 276 is provided from node 1 (250) to node 4 (256). As shown, the source node column 503 identifies a source of data for a specific functional connection while the sink node column 505 identifies a sink for the data upon the functional connection and the bandwidth column 507 identifies the bandwidth needed for the connection.

After the connection table has been generated for the functional design specified by the user, the process analyzes the first connection (Block 504) and sets a data variable "CONNECTION" = 1. In the preferred embodiment, CONNECTION is a marker utilized by the process to identify from which row in the connection table to obtain source/sink information. However, the process of the present invention may utilize any scheme for designating connections, provided the process considers each connection in determining whether the hardware design supports a given functional design.

The process continues with determining whether "CONNECTION" = LAST (Block 506). Since at this instance, "CONNECTION" = 1 the process continues with retrieving from the connection table for the connection row specified (i.e., in this case connection 1) the source, sink, and bandwidth data provided in the table (Block 510). Also, the process determines the desired route from the source to the sink. For the preferred embodiment, the process determines the route by first modeling the network as a graph. Graph modeling of connected elements, for example, the hardware design or the functional design, is accomplished utilizing commonly known in the art techniques such as graph transveral algorithms. Commonly, such algorithms model a physical network by simulating the algorithms employed by the actual network devices in routing data. For example, Ethernet switches commonly use the Spanning Tree algorithm (IEEE 802.1d bridge protocol) or the Link Aggregation Algorithm (IEEE 802.3ad) to choose how data is routed across

the connections and devices in a given network configuration. After the data-flow within a physical network has been modeled, other graph transversal algorithms commonly known in the art, such as breadth-first search, are preferably employed by the present invention to determine the route that each functional source to sink connection will utilize across the network.

Upon determining a route from the source to the sink, the process preferably stores each route between a source and sink provided in the connection table in appropriate data files. These "route" data files may be recalled as necessary. The process also preferably entails determining whether the route for the connection at issue is empty (Block 512). If the route is empty, an error condition is generated (Block 514). As is commonly known in the art, the before mentioned route determining algorithms may return a result that indicates a route is "empty" (i.e., a connection between a source and a sink node is not possible) because, for example, the hardware design lacks a physical connection between two devices, as specified in a functional design. Such an empty route condition would exist in the present example if connection 266 was not specified by the user during the hardware design phase. In such a situation, data could not be communicated between nodes 7 and 8 and the various nodes connected to those switches.

When the route is not empty, the process obtains bandwidth data from the connection table for the source node (node 1 for the first connection example) (Block 516). The process then generates the Node Results table 523 which is preferably a data array that identifies the Maximum Bandwidth 511 available (or the Maximum Resources available) and the Bandwidth Needed 513 (or the Resources Needed) at each Node 509 (see **Figure 5C**). Similarly, an Edge Table 515 is generated which provides the Maximum Bandwidth 519 available and Bandwidth Needed 521 for each Edge 517. The values in the Maximum Bandwidth columns 511 and 519 are preferably

obtained from the specification provided with each hardware device and/or cabling selected by the user while inputting the hardware design into a system implementing the present invention. As shown in **Figure 5C,** for purposes of the present example, nodes 1-8 are specified as each having a maximum bandwidth of 100 Mbits/sec. Additionally, as is commonly used in the art, an "edge" is a connection between two devices (or "nodes") utilized in a network configuration.

In Block 516, the process also adds the bandwidth data obtained from the Bandwidth column 507 in the Connection Table 500 to the resulting "Bandwidth Needed" column 513 of the "Node Results" table 523 for the first portion of the route from the source (node 1) to the sink (node 4). In this instance, 50 Mbits is added to the value (initially set to "0") in the Bandwidth Needed column for node 1 resulting in a needed bandwidth of 50Mbits/sec.

After computing the bandwidth needed for the first node of the route for the first connection, i.e., from source node 1 to sink node 4, the process queries whether the route data array is empty (Block 518). As is commonly appreciated, a route data array is empty when a user has traced a path from a source to a sink. At this instant, for the present example, the route data array is not empty. Therefore, the process preferably proceeds to the next entry in the route data array which is an "edge". As is commonly known, a route data array generally comprises a source node followed by a series of edge, node, edge, node, etc. until the sink node is reached. In this example, the next edge is edge 1. For the route from source node 1 to sink node 4, a route data array would consist of the following entries, in sequence: node 1, edge 1, node 6, edge 6, node 7, edge 7, node 8, edge 4, node 4.

As shown in Block 520, the process then obtains the bandwidth data for the edge from the connection table and adds the resulting bandwidth to the Bandwidth Needed column 521 of the

edge results table shown in **Figure 5D**. The process then continues with proceeding to the next entry in the route data array, obtaining the bandwidth data for the next node in the queue from the connection table and entering the bandwidth data into the appropriate row in the Bandwidth Needed column 513 (Block 522). In the present example, the next node is node 6. As such, the process provides in the node results table shown in **Figure 5C** the bandwidth needed of 50 Mbits/sec for the node 6 row. The process then inquires whether the route is empty (Block 518). At this stage of the present example, the route is not empty and contains the value for edge 6, which connects node 6 with node 7, as shown in **Figure 3B**. The processing then completes Blocks 520-522-518 until the entire route data array from a source node of 1 to the sink node 4 has been entered into the Node Results table 523 and the Edge Results table 515. As shown in **Figures 5C** and **5D**, the results of entering the connection data for a first connection through a complete route results in 50 Mbits/sec of bandwidth being needed in nodes 1, 4, 6, 7, and 8 and edges 1, 4, 6, and 7. Similar calculations could be accomplished by the present invention for other resource requirements.

At this point, the route is empty and the process continues with incrementing the value in "CONNECTION" by 1 (Block 524). In the present example, at this point CONNECTION = 2. As such, in Block 506 since CONNECTION is not equal to LAST, the processing continues through Blocks 510 through 522 up through 518 until the node results and edge results tables have been completely filled out. As shown in **Figures 5E** for nodes 1 and 6, the second connection results in an additional 25 Mbits/sec of bandwidth being added to the 50 Mbits/sec of bandwidth previously needed for the first connection. Also, node 2 needs 25 Mbits/sec of bandwidth

Similarly, in the Edge Results table 525 (**Figure 5F**), an additional 25 Mbits/sec of bandwidth are required for edge 1 and an initial 25 Mbits/sec are required for edge 2.

By repeating this process for each of the connections provided in the connection table 500 (see **Figure 5B**) the present invention generates the Node Results 529 and Edge Results 531 tables shown in **Figures 5G** and **5h**, respectively.

Upon proceeding through each of the connections provided in the Connection table 500, the process flow reaches the end of the data array and sets CONNECTION equal to LAST. Since CONNECTION = LAST at Block 506, the process proceeds to Block 508.

In Block 508, the process compares the Bandwidth Needed column 533 against the Maximum Bandwidth 511 available for each Node 509. Similarly, the Bandwidth Needed column 535 is compared against the Maximum Bandwidth 519 for each Edge 517. The result of these comparisons are preferably designated as a network design analysis results as shown in **Figure 6**.

As shown, the network design analysis result output identifies the maximum data capacity needed for each edge and device/node specified in the hardware design. Preferably, the present invention displays this data graphically and utilizes color-coding and various other schemes to indicate the amount of resource needed by an edge and/or a device/node to support a given network functional design in light of the hardware design. However, the present invention may be configured such that the Node Results and Edge Results tables are provided to the user, or the information contained therein is provided in another format.

Similarly, when implementing the before mentioned design analysis functions, the computer workstation 900 might return a display as shown in **Figure 12**. As shown, both the hardware connections 1202 and the functional connections 1204 between the various network devices 1206

are depicted by a hollow line indicating the hardware connections can support the resource requirements specified in the functional design. Alternatively, color coding may be utilized by the present invention to indicate the level of available resources (or of a given resource, for example, bandwidth) being utilized by a given device or connection. For example, a "green" coded device/connection might represent a device/connection which is not using greater than 50% of its resources. Similarly, a "yellow" coded device/connection might represent a device/connection which is using between 50% and 100% of its resources. Lastly, a "red" coded device/connection might indicate a device/connection which is overloaded. Those skilled in the art appreciate that various other indicating schemes may be utilized to convey such resource utilization information.

Figure 13 illustrates a scenario in which too much data is trying to be passed upon a given hardware design. As shown by the dark cabling 1302 between node 7 (262) and node 8 (264), too much data is being attempted to be passed through the edge 1302. As shown, the process of the present invention preferably annotates those functional data links 1304 which are not supported by the hardware design of the present invention. However, the present invention may also be configured such that hardware devices which can not support a functional design may be suitably annotated. For example, by designating a cable connection as lacking sufficient capacity.

In an alternative embodiment of the present invention, in addition to providing the before mentioned features and functions, a system implementing the present invention recommends changes to a design based upon the results of the analysis function. **Figure 7** provides a flowchart representing one embodiment of a process which provides design input and analysis functions while also recommending changes to a design. As shown, the process for blocks 702-708 mirrors the process provided for the preferred embodiment. As provided above, the process enables a user to

input a hardware design and a functional design, and verify the designs. In addition, based upon the results of the design analysis (Block 708), the present invention determines whether the hardware design supports the functional design by determining whether any functional data connection can not be accomplished by the hardware design (Block 710). When a functional design is not supported, the process queries the user as to whether the process should suggest changes to the design (Block 714).

At this point, the user may decide to implement their own changes to the functional and/or hardware design. In such a situation, the process provides the user with the design analysis results (Block 712) and the process terminates (Block 722). When the user desires for the process to suggest changes to the design, the process queries the user as to whether to changes should be made to the hardware design or the functional design (Block 716). When changes to the hardware design are desired, the process suitably adds/modifies/deletes those hardware configurations necessary to support the functional design (Block 720). Similarly, when changes to the functional design are desired, the process add/modifies/deletes those functional connections which can not be supported by the specified hardware design (Block 718).

Figure 14A provides an example of a screen display from a system implementing this second embodiment. As shown, a network connection 1400 between node 7 (262) and node 8 (264) is missing. As such, the functional data connections 1402 are not supported by the hardware design. As shown, the present invention preferably provides a textual description 1404 identifying the error condition. **Figure 14B** provides one solution to the deficient design in which the hardware is changed. As shown, a data connection 1406 is added by the process of the present invention between node 7 (262) and node 8 (264). In contrast, **Figure 14C** provides a solution to the

deficient design in which the functional design is changed by deleting the functional data connections which can not be supported by the hardware design. Upon reviewing the results of modifying the functional design, the user might decide that such modifications are undesirable and decide to add/modify the hardware design.

Referring now to **Figure 8**, in a third embodiment, the present invention determines a hardware network design based upon a functional network design input by a user. As shown in **Figure 15A**, this process begins with a user inputting functional transceiver devices into a design view field 1500 (Block 802). Each of the functional transceiver devices 1502, 1504, 1506, 1508, and 1510 identify data input ports 1512, data output ports 1514, functional network input ports 1516, and functional network output ports 1518. Those skilled in the art appreciate that the data input ports 1512 and data output ports 1514 may be suitably connected to various devices including, but not limited to, microphones, sound processors, speakers, video feeds, video displays, and computer data terminals. For this process, the user preferably designates data types reaching each of the various ports (Block 804). Additionally, those skilled in the art appreciate that the functional transceivers specified by the user may include any level of details, including intended use (for example, audio processing), desired number of ports, and similar parameters. Likewise, the transceivers may contain no specifics whatsoever and may constitute mere place holders until more information is available for determining suitable transceivers.

Once the data types being received and processed by the present invention have been identified, data connections between the input ports 1512 and network output ports 1518 and between the network input ports 1516 and the data output ports 1514 are specified (Block 806). As shown in **Figure 15B**, these connections are preferably made by connecting each data input

port 1514 with at least one network output port 1518 in a data connection field 1520, as necessary, until all the functional data connections are designated.

As shown in **Figure 15C**, once the functional data connections within a functional transceiver device are specified, the user provides the various functional interconnections desired between the functional devices by providing the appropriate functional wiring connections 1522 (Block 808). These functional connections 1522 preferably provide data rates and data direction flows by connecting output ports 1518 to input ports 1516. Those skilled in the art, however, appreciate that functional connections between transceivers may be designated before connections between data input/output ports and functional network ports are accomplished. As such, the process of the present invention may be suitably modified as necessary to reflect a design approach desired by a user.

Upon establishing the data connections and flows, the process preferably continues with determining the resources (for example, bandwidth) required from each functional device (Block 810). In this step, the process calculates how much data capacity a specific functional device will need to provide. Additionally, the process preferably characterizes data types by function, i.e., audio channels, video channels, telemetry channels, etc. Once the resources and channel types are determined for a functional device, the process identifies a hardware device which provides the data ports and processing capabilities needed (Block 812). Preferably, the process determines hardware devices by utilizing lookup tables and identifying in a list those devices which meet certain specific requirements. By utilizing known in the art optimization algorithms, the process then selects from the list the optimal actual device for each functional device.

The process of determining resources and selecting hardware devices to provide a given functional device continues until each functional device has been identified as a hardware device (Block 814). At this instant, the process generates a connection table based upon the functional description (Block 816). As explained previously for the other embodiments, the connection table is preferably utilized to identify sources, sinks, and resources (in this case, bandwidth) requirements. Additionally, the process determines the total resources (bandwidth) into and out of a hardware device (Block 818). In determining the resource (bandwidth) requirements, the process functionally generates a data table which identifies total needs for any switch to be connected.

The process then virtually connects each device to a switch (Block 820), as such, if five transceivers were specified, then a separate switch1524 is connected, via a suitable cable 1526, to each transceiver 1528, as shown in **Figure 15D**. Based upon the data requirements of each switch-transceiver pairing, the process then determines whether any switch has excess data capacity (Block 822). If a switch has excess data capacity, the process generates a listing showing which other switches in the hardware design the present switch sends data through to reach a given transceiver.

Based upon this listing, the process identifies those switches, if any, which when combined optimize data usage and routing. In performing this optimization, the present invention may suitably configure factors such as component costs, cabling requirements, and other resources including, bandwidth, latency, excess capacity requirements, hop counts, distances, whether excess capacity is desired for future additions, and various other factors. Based upon these optimization requirements, the process might return a result, as shown in **Figure 15E**, wherein switches 1 and 2

have been combined into a single switch 1530 and switches 4 and 5 have been combined into a single switch 1532.

After the appropriate switches, if any, have been combined, the process verifies the minimum desired number of switches have been designated (Block 824). This minimization process is preferably based upon pre-established rules and/or user input. Next, the process completes the hardware design by connecting the switches to each other, as appropriate, with the appropriate cabling (Block 826) or wireless connections. **Figure 15F** provides a representation of a hardware configuration which has been generated based upon a functional design. As such, the present invention provides a process and system for designing a network based upon a hardware design and a functional design, or only upon a functional design.

In a fourth embodiment, the present invention, upon user direction, will automatically configure the various devices specified in a hardware network design. As shown in **Figure 16**, a configuring system 1602, which is preferably the same system as that used to design and verify a network configuration, is suitably connected to the network 1604. The configuring system 1602 may be connected to the network 1604 as a component on the network via a connection 1606 to a switch or each device in the network 1604 might be independently connected 1608 to the configuring system. Those skilled in the art appreciate the various configurations by which a configuring system 1602 might be connected to the various devices in a network 1604 such that the devices may be configured and/or monitored by the configuring system 1602. The present invention is not limited to any specific embodiment. Additionally, is it understood by those skilled in the art, that the present invention may only configure those devices which are remotely configurable. Devices which do not possess remote configuring capacities are not considered to fall within the

purview of this fourth embodiment of the present invention, unless modifications to such devices are accomplished which allow remote communications between a configuring system and such a device. Additionally, configuring devices remotely is well known in the art. The present invention may be suitably modified to support any configuration scheme for networked devices.

While the present invention has been described and illustrated with reference to a preferred embodiment and various other embodiments, it will be appreciated by those skilled in the art that changes to the above descriptions or illustrations may be made with respect to form or detail without departing from the spirit and scope of the present invention.

CLAIMS

1. A method for designing and verifying a network based upon a specified hardware design and a specified functional design, said method comprising the steps of:

a. specifying a hardware design for a network, wherein said hardware design specifies at least two nodes and one edge, said edge establishing a physical network connection between said nodes;

b. specifying a functional design for said network, wherein said functional design specifies at least one functional network connection between any two nodes in said network;

c. analyzing said hardware design in view of said functional design to determine whether said hardware design can support said functional design; and

d. providing a result of said analysis.

2. The method of claim 1 wherein said step of specifying a hardware design for a network further comprises the steps of:

a. specifying a type of network;

b. specifying a hardware device for said network;

c. repeating step b. until every hardware device desired in said network has been specified;

d. specifying a connection between a first hardware device and a second hardware device;

e. repeating steps b-d until all devices within said network have been connected to at least one other device.

3. The method of claim 2 wherein said type of network is selected from the group consisting of an Ethernet network, a Token Ring network, a Fiber Digital Data Interface Network, an ATM network, a Localtalk network, a FireWire network, and an USB network.

4. The method of claim 2 wherein said hardware devices is one selected from the group consisting of a transceiver, a repeater hub, and a switch.

5. The method of claim 4 wherein said method further comprises the step of modifying at least one characteristic associated with said hardware device.

6. The method of claim 2 wherein said connection is established using Ethernet compatible cabling.

7. The method of claim 2 wherein said connection is a wireless connection.

8. The method of claim 2 wherein said connection is established using IEEE-1394 compatible cabling.

9. The method of claim 2 wherein said connection is established using USB compatible cabling.

10. The method of claim 2 wherein said method further comprises the step of verifying a cable connecting a first hardware device and a second hardware device is rated for said distance specified between said first and said second devices.

11. The method of claim 1 wherein said step of specifying a hardware design for a network further comprises the steps of:

a. connecting a system to an existing network, wherein said network has at least one device connected to a second device;

b. determining, via said system, a configuration for said network, wherein said configuration includes an indication of a device type for each device provided in said system, and any cabling connecting a first device with a second device within said network; and

c. specifying a result of said determination as said hardware design.

12. The method of claim 1 wherein said step of specifying a functional design for said network further comprises the steps of:

a. selecting a first device from which data will be sent;

b. selecting a second device at which said data will arrive;

c. specifying a data type for said data; and

d. repeating steps a-c when another connection between two devices is specified in said functional design.

13. The method of claim 1 wherein said step of analyzing said hardware design in view of said functional design to determine whether said hardware design supports said functional design is accomplished on a computer system.

14. The method of claim 1 wherein said step of analyzing said hardware design in view of said functional design to determine whether said hardware design can support said functional design further comprises the steps of:

a. generating a connection table based upon said functional design, said connection table identifying a source node, a sink node, and a resources for each functional connection;

b. determining a route for each functional connection, said route identifying a physical connection over which data travels from said source node via at least one edge to said sink node;

c. determining at least one resource needed for each node and edge utilized in said route for said connection;

d. determining for each node and for each edge a total resources needed to support each connection;

e comparing for each node said total resources needed against a maximum resources provided by said node;

f. comparing for each edge said total resources needed against a maximum resources provided by said edge; and

g. annotating each node and each edge identified in said hardware design based upon a result of said comparisons.

15. The method of claim 1 wherein said result indicates a level of utilization of a maximum resource for a node identified in said hardware design by said functional design.

16. The method of claim 1 wherein said result indicates a level of utilization of a maximum resource for an edge identified in said hardware design by said functional design.

17. The method of claim 1 wherein said result indicates whether a functional network connection is not supported by said hardware design.

18. The method of claim 1 wherein said method further comprises the steps of determining whether said hardware design supports said functional design.

19. The method of claim 18 wherein said step of determining whether said hardware design supports said functional design further comprises the steps of:

a. determining a maximum resource available for each node and each edge provided in said hardware design;

b. determining a quantity of resource needed at each node and at each edge based upon said functional design;

c. comparing said resource needed against said maximum resource available for each node and for each edge; and

d. designating nodes or edges as lacking sufficient capacity where a result of said comparison indicates said resource needed at said node or edge is greater than said maximum resource available.

20. The method of claim 18 wherein said step further comprises the step of recommending changes to said hardware design so that said hardware design can support said functional design.

21. The method of claim 19 wherein said step further comprises the step of recommending changes to said functional design so that said functional design can be supported by said hardware design.

22. The method of claim 1 wherein each of said steps of specifying a hardware design for a network and specifying a functional design for a network are accomplished via a graphical user interface.

23. A system for checking a hardware network design against a functional network design, said system comprising:

a means for entering a hardware network design, wherein said hardware network design comprises a network of connected devices including a first transceiver physically connected to a second transceiver via said network;

a means for entering a functional network design, wherein said functional network design includes at least one functional connection between said first transceiver and said second transceiver;

a means for verifying said hardware network design can support said functional network design; and

a means for providing an output indicative of a result of said verification.

24. A computer program for designing a data network and verifying a hardware configuration of said network can support a functional design of said network, said computer program comprising:

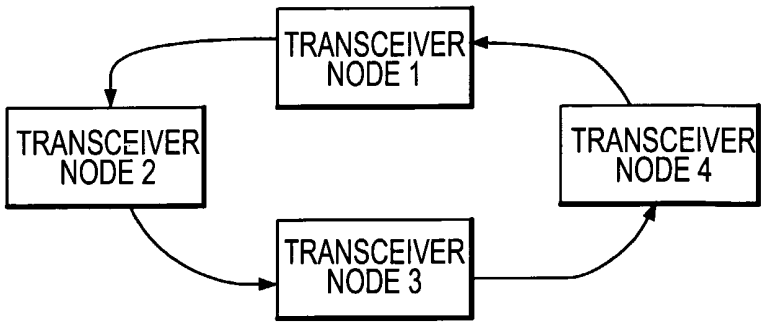
a first data structure representing a hardware network design comprising at least one transceiver connected to a second transceiver via a network;

a second data structure representing a functional network design comprising a functional connection between a first transceiver and a second transceiver;

an algorithm which determines whether aid first data structure can support said second data structure.

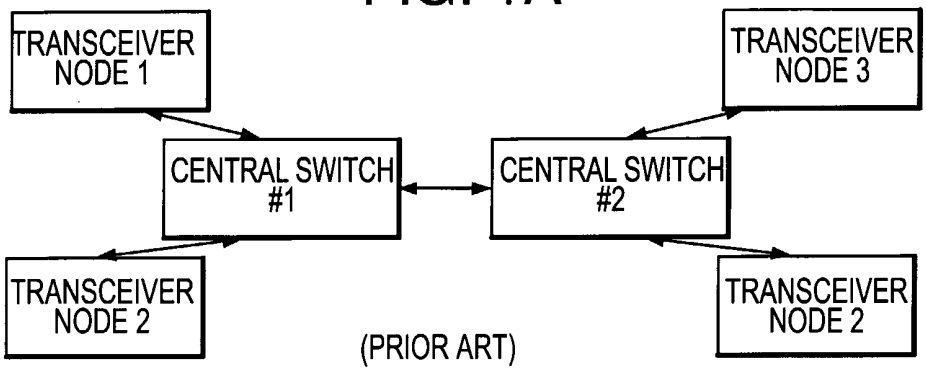
25. The computer program of claim 24 wherein said computer program further comprises a graphical user interface for entering said first data structure, said second data structure, and displaying a result of said determinations by said algorithm.

26. The computer system of claim 21 wherein said system further comprises a means for remotely configuring each of said devices within said hardware network design to support said functional network design.



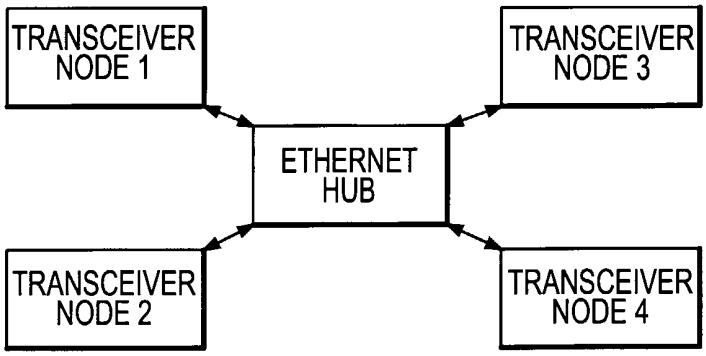
(PRIOR ART)

FIG. 1A



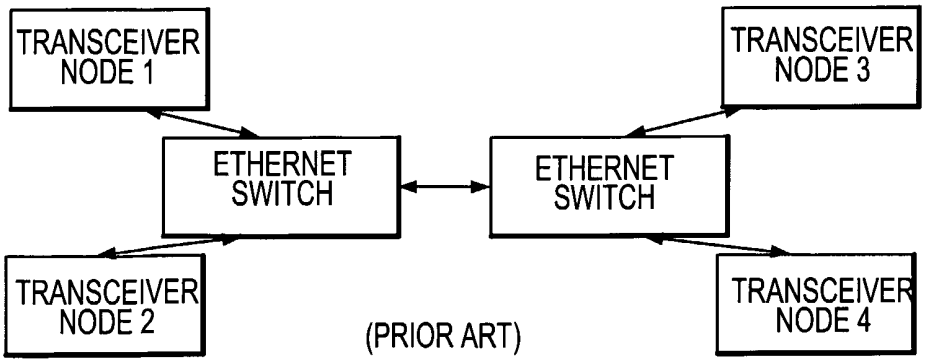
(PRIOR ART)

FIG. 1B



(PRIOR ART)

FIG. 1C



(PRIOR ART)

FIG. 1D

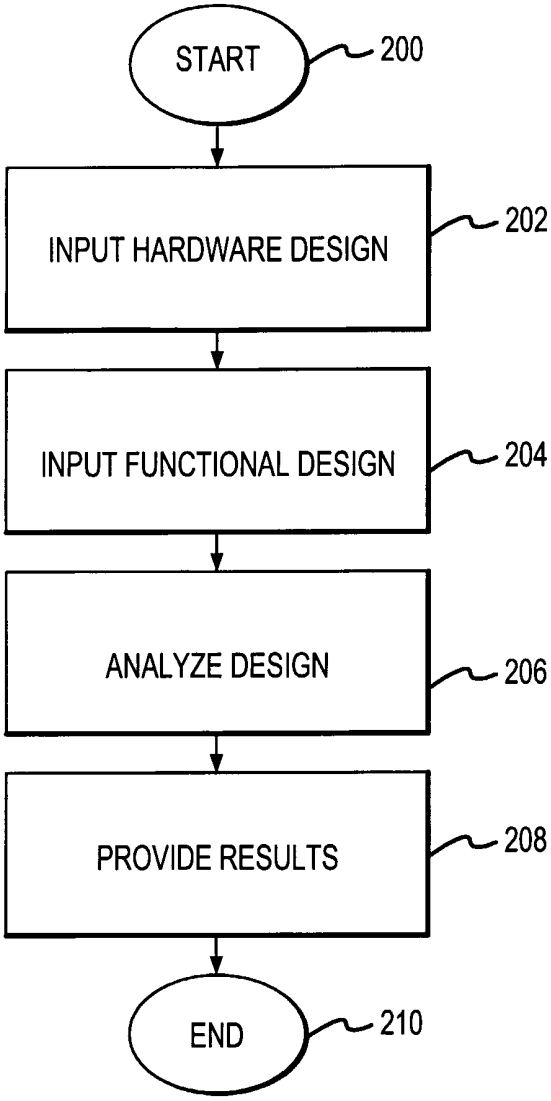

START
200
INPUT HARDWARE DESIGN
202
INPUT FUNCTIONAL DESIGN
204
ANALYZE DESIGN
206
PROVIDE RESULTS
208
END
210

FIG. 2

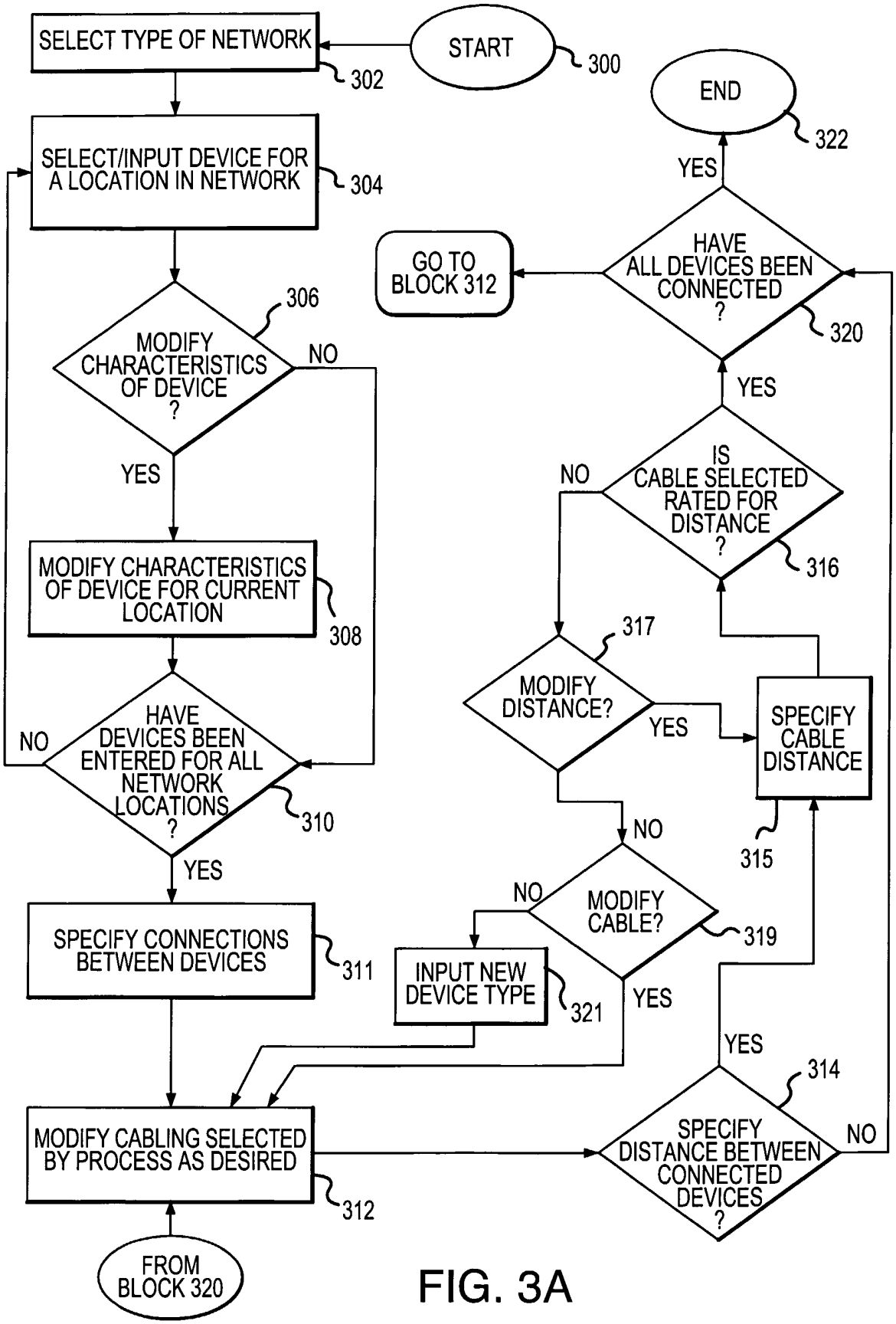

START
300
SELECT TYPE OF NETWORK
302
SELECT/INPUT DEVICE FOR A LOCATION IN NETWORK
304
MODIFY CHARACTERISTICS OF DEVICE ?
306
NO
YES
MODIFY CHARACTERISTICS OF DEVICE FOR CURRENT LOCATION
308
HAVE DEVICES BEEN ENTERED FOR ALL NETWORK LOCATIONS ?
310
NO
YES
SPECIFY CONNECTIONS BETWEEN DEVICES
311
MODIFY CABLING SELECTED BY PROCESS AS DESIRED
312
FROM BLOCK 320
END
322
YES
HAVE ALL DEVICES BEEN CONNECTED ?
320
GO TO BLOCK 312
YES
IS CABLE SELECTED RATED FOR DISTANCE ?
316
NO
317
MODIFY DISTANCE?
YES
SPECIFY CABLE DISTANCE
315
NO
NO
MODIFY CABLE?
319
INPUT NEW DEVICE TYPE
321
YES
YES
314
SPECIFY DISTANCE BETWEEN CONNECTED DEVICES ?
NO

FIG. 3A

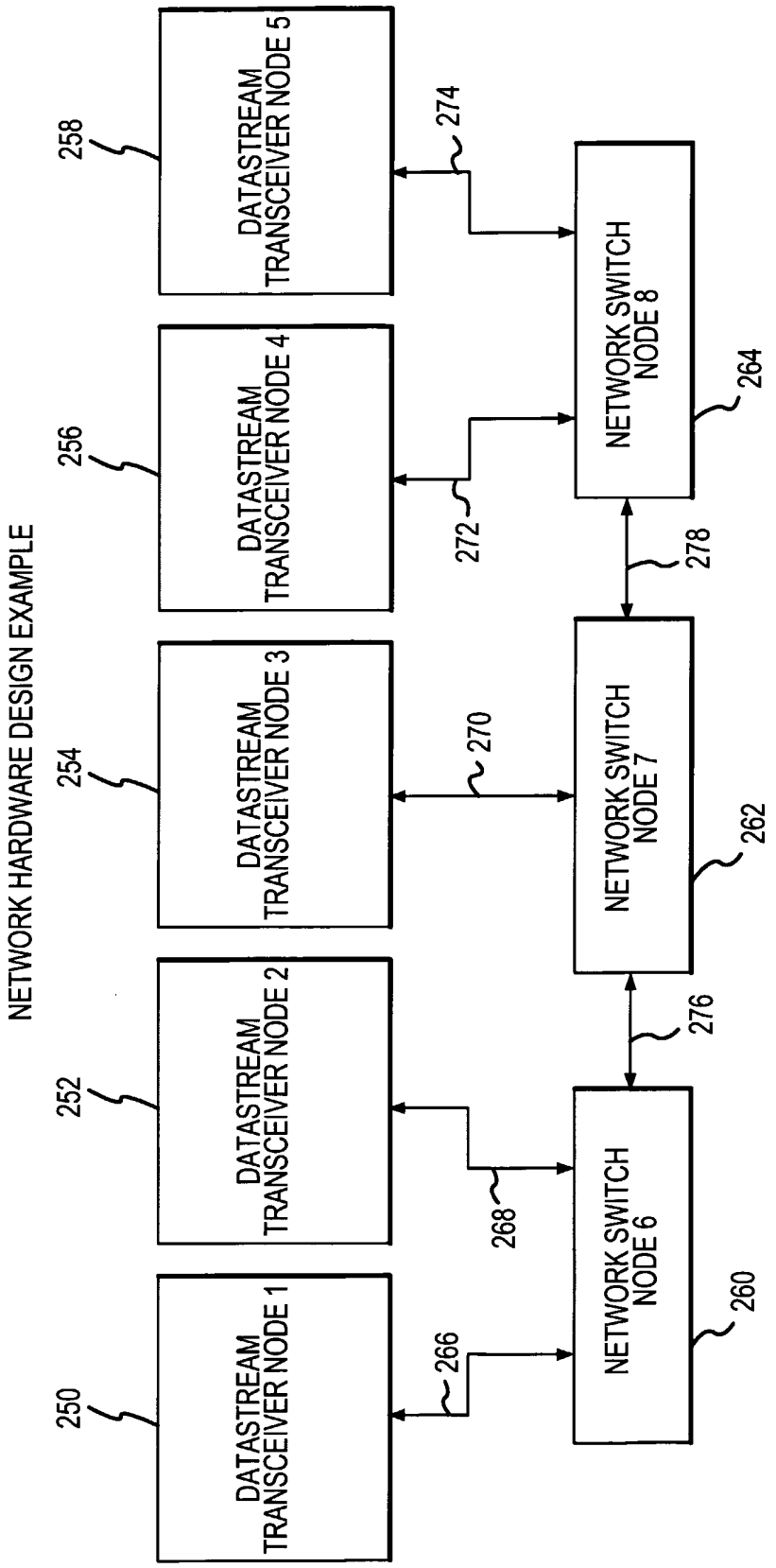

NETWORK HARDWARE DESIGN EXAMPLE
DATASTREAM TRANSCEIVER NODE 1
DATASTREAM TRANSCEIVER NODE 2
DATASTREAM TRANSCEIVER NODE 3
DATASTREAM TRANSCEIVER NODE 4
DATASTREAM TRANSCEIVER NODE 5
NETWORK SWITCH NODE 6
NETWORK SWITCH NODE 7
NETWORK SWITCH NODE 8
250
252
254
256
258
260
262
264
266
268
270
272
274
276
278

FIG. 3B

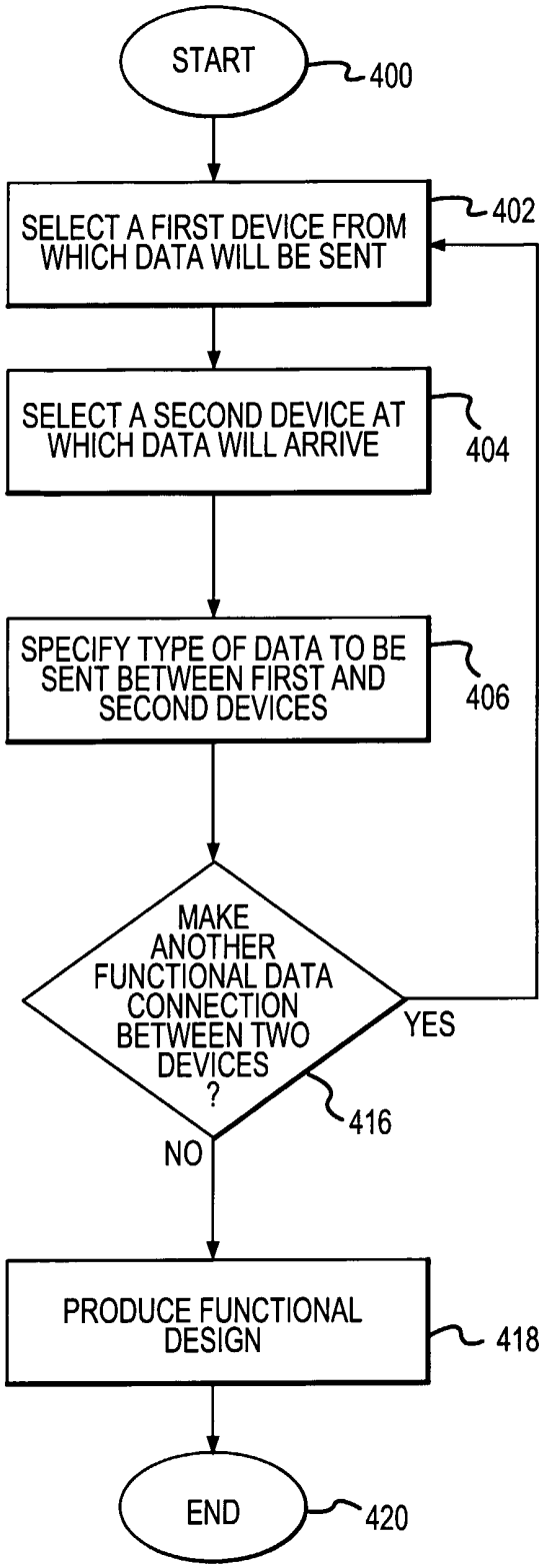

START
400
SELECT A FIRST DEVICE FROM WHICH DATA WILL BE SENT
402
SELECT A SECOND DEVICE AT WHICH DATA WILL ARRIVE
404
SPECIFY TYPE OF DATA TO BE SENT BETWEEN FIRST AND SECOND DEVICES
406
MAKE ANOTHER FUNCTIONAL DATA CONNECTION BETWEEN TWO DEVICES ?
YES
416
NO
PRODUCE FUNCTIONAL DESIGN
418
END
420

FIG. 4A

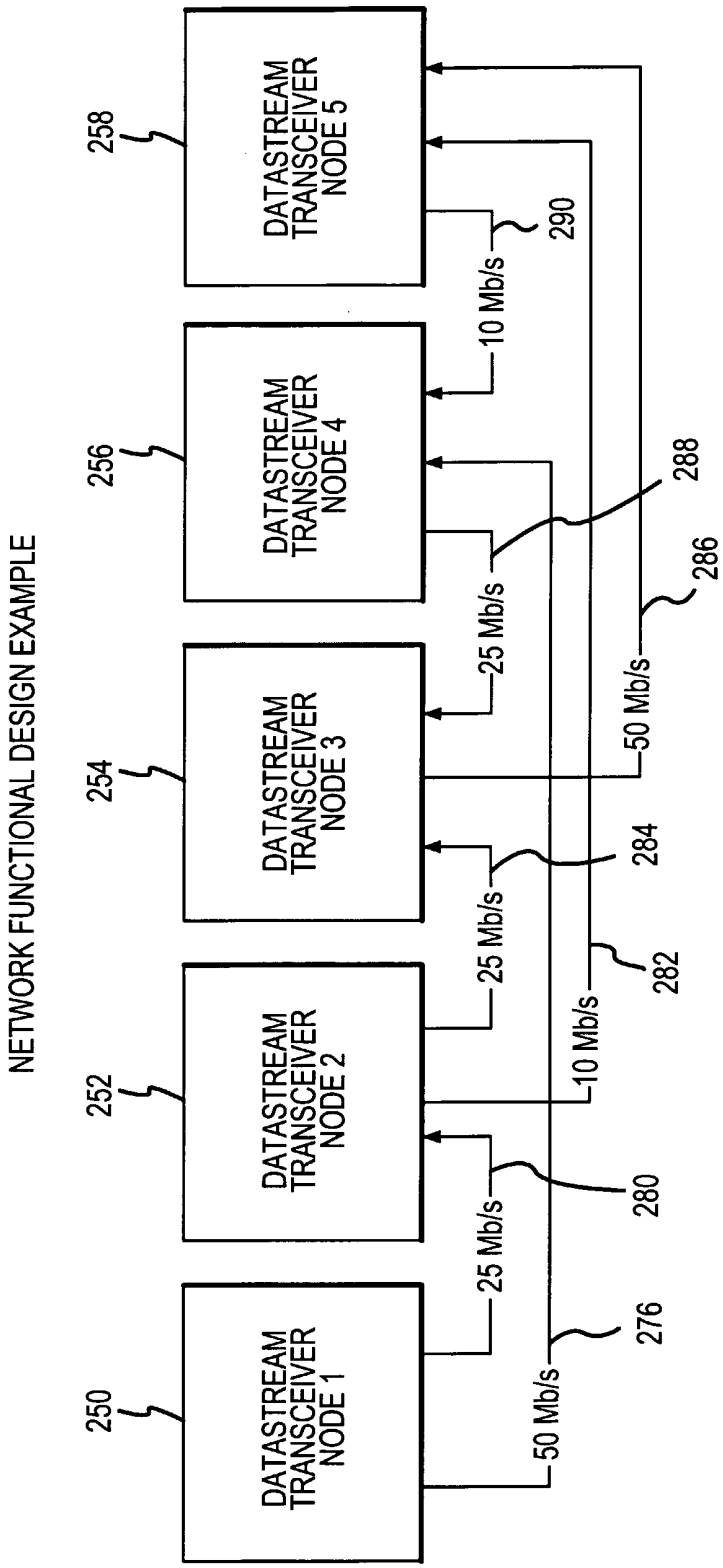

NETWORK FUNCTIONAL DESIGN EXAMPLE
DATASTREAM TRANSCEIVER NODE 1
DATASTREAM TRANSCEIVER NODE 2
DATASTREAM TRANSCEIVER NODE 3
DATASTREAM TRANSCEIVER NODE 4
DATASTREAM TRANSCEIVER NODE 5
250
252
254
256
258
25 Mb/s
10 Mb/s
50 Mb/s
276
280
282
284
286
288
290

FIG. 4B

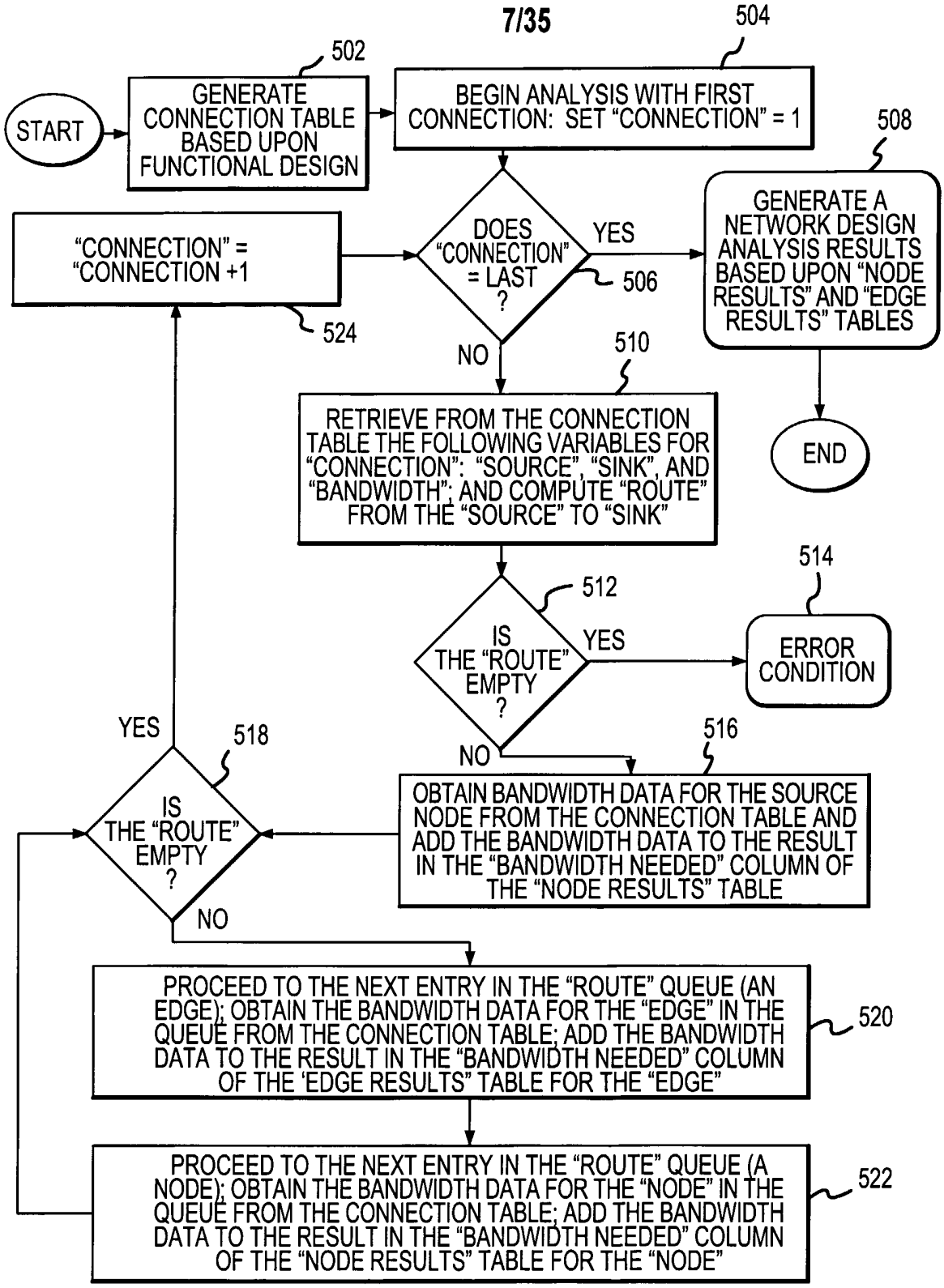

START
GENERATE CONNECTION TABLE BASED UPON FUNCTIONAL DESIGN
502
BEGIN ANALYSIS WITH FIRST CONNECTION: SET "CONNECTION" = 1
504
DOES "CONNECTION" = LAST ?
YES
506
NO
"CONNECTION" = "CONNECTION +1
524
GENERATE A NETWORK DESIGN ANALYSIS RESULTS BASED UPON "NODE RESULTS" AND "EDGE RESULTS" TABLES
508
END
RETRIEVE FROM THE CONNECTION TABLE THE FOLLOWING VARIABLES FOR "CONNECTION": "SOURCE", "SINK", AND "BANDWIDTH"; AND COMPUTE "ROUTE" FROM THE "SOURCE" TO "SINK"
510
IS THE "ROUTE" EMPTY ?
512
YES
ERROR CONDITION
514
NO
OBTAIN BANDWIDTH DATA FOR THE SOURCE NODE FROM THE CONNECTION TABLE AND ADD THE BANDWIDTH DATA TO THE RESULT IN THE "BANDWIDTH NEEDED" COLUMN OF THE "NODE RESULTS" TABLE
516
IS THE "ROUTE" EMPTY ?
518
YES
NO
PROCEED TO THE NEXT ENTRY IN THE "ROUTE" QUEUE (AN EDGE); OBTAIN THE BANDWIDTH DATA FOR THE "EDGE" IN THE QUEUE FROM THE CONNECTION TABLE; ADD THE BANDWIDTH DATA TO THE RESULT IN THE "BANDWIDTH NEEDED" COLUMN OF THE 'EDGE RESULTS" TABLE FOR THE "EDGE"
520
PROCEED TO THE NEXT ENTRY IN THE "ROUTE" QUEUE (A NODE); OBTAIN THE BANDWIDTH DATA FOR THE "NODE" IN THE QUEUE FROM THE CONNECTION TABLE; ADD THE BANDWIDTH DATA TO THE RESULT IN THE "BANDWIDTH NEEDED" COLUMN OF THE "NODE RESULTS" TABLE FOR THE "NODE"
522

FIG. 5A

500

CONNECTION TABLE
[WITH IDENTIFYING NUMBERS IN FIGURE 3B SHOWN IN "( )"]

CONNECTION		SOURCE NODE		SINK NODE		BANDWIDTH
1 (276)		1 (250)		4 (256)		50 Mb/s
2 (280)		1 (250)		2 (252)		25 Mb/s
3 (282)	501	2 (252)	503	5 (258)	505	10 Mb/s
4 (284)		2 (252)		3 (254)		25 Mb/s
5 (286)		3 (254)		5 (258)		50 Mb/s
6 (288)		4 (256)		3 (254)		25 Mb/s
7 (290)		5 (258)		4 (256)		10 Mb/s

507

FIG. 5B

513

NODE RESULTS				
NODE [WITH IDENTIFYING NUMBERS IN FIGURE 2B SHOWN IN "( )"]	MAXIMUM BANDWITH			BANDWIDTH NEEDED
1 (250)	100 Mb/s			50 Mb/s
2 (252)	100 Mb/s			0 Mb/s
3 (254)	100 Mb/s	514	513	0 Mb/s
4 (256)	100 Mb/s			50 Mb/s
5 (258)	100 Mb/s			0 Mb/s
6 (260)	100 Mb/s			50 Mb/s
7 (262)	100 Mb/s			50 Mb/s
8 (264)	100 Mb/s			50 Mb/s

509

FIG. 5C

515

EDGE RESULTS				
EDGE [WITH IDENTIFYING NUMBERS IN FIGURE 2B SHOWN IN "( )"]		MAXIMUM BANDWITH		BANDWIDTH NEEDED
1 (266)		100 Mb/s		50 Mb/s
2 (268)	519	100 Mb/s	521	0 Mb/s
3 (270)		100 Mb/s		0 Mb/s
4 (272)		100 Mb/s		50 Mb/s
5 (274)		100 Mb/s		0 Mb/s
6 (276)		100 Mb/s		50 Mb/s
7 (278)		100 Mb/s		50 Mb/s

517

FIG. 5D

525

NODE RESULTS		
NODE [WITH IDENTIFYING NUMBERS IN FIGURE 2B SHOWN IN "( )"]	MAXIMUM BANDWIDTH	BANDWIDTH NEEDED
1 (250)	100 Mb/s	50+25=75 Mb/s
2 (252)	100 Mb/s	25 Mb/s
3 (254)	100 Mb/s	0 Mb/s
4 (256)	100 Mb/s	50 Mb/s
5 (258)	100 Mb/s	0 Mb/s
6 (260)	100 Mb/s	50+25=75 Mb/s
7 (262)	100 Mb/s	50 Mb/s
8 (264)	100 Mb/s	50 Mb/s

FIG. 5E

527

EDGE RESULTS		
EDGE [WITH IDENTIFYING NUMBERS IN FIGURE 2B SHOWN IN "( )"]	MAXIMUM BANDWIDTH	BANDWIDTH NEEDED
1 (266)	100 Mb/s	50+25=75 Mb/s
2 (268)	100 Mb/s	25 Mb/s
3 (270)	100 Mb/s	0 Mb/s
4 (272)	100 Mb/s	50 Mb/s
5 (274)	100 Mb/s	0 Mb/s
6 (276)	100 Mb/s	50 Mb/s
7 (278)	100 Mb/s	50 Mb/s

FIG. 5F

529

NODE RESULTS			
NODE [WITH IDENTIFYING NUMBERS IN FIGURE 2B SHOWN IN "( )"]	MAXIMUM BANDWITH		BANDWIDTH NEEDED
1 (250)	100 Mb/s		75 Mb/s
2 (252)	100 Mb/s		60 Mb/s
3 (254)	100 Mb/s	511	533 100 Mb/s
4 (256)	100 Mb/s		85 Mb/s
5 (258)	100 Mb/s		70 Mb/s
6 (260)	100 Mb/s		110 Mb/s
7 (262)	100 Mb/s		160 Mb/s
8 (264)	100 Mb/s		145 Mb/s

509

FIG. 5G

531

EDGE RESULTS			
EDGE [WITH IDENTIFYING NUMBERS IN FIGURE 2B SHOWN IN "( )"]		MAXIMUM BANDWITH	BANDWIDTH NEEDED
1 (266)		100 Mb/s	75 Mb/s
2 (268)	519	100 Mb/s	535 60 Mb/s
3 (270)		100 Mb/s	100 Mb/s
4 (272)		100 Mb/s	85 Mb/s
5 (274)		100 Mb/s	70 Mb/s
6 (276)		100 Mb/s	85 Mb/s
7 (278)		100 Mb/s	135 Mb/s

517

FIG. 5H

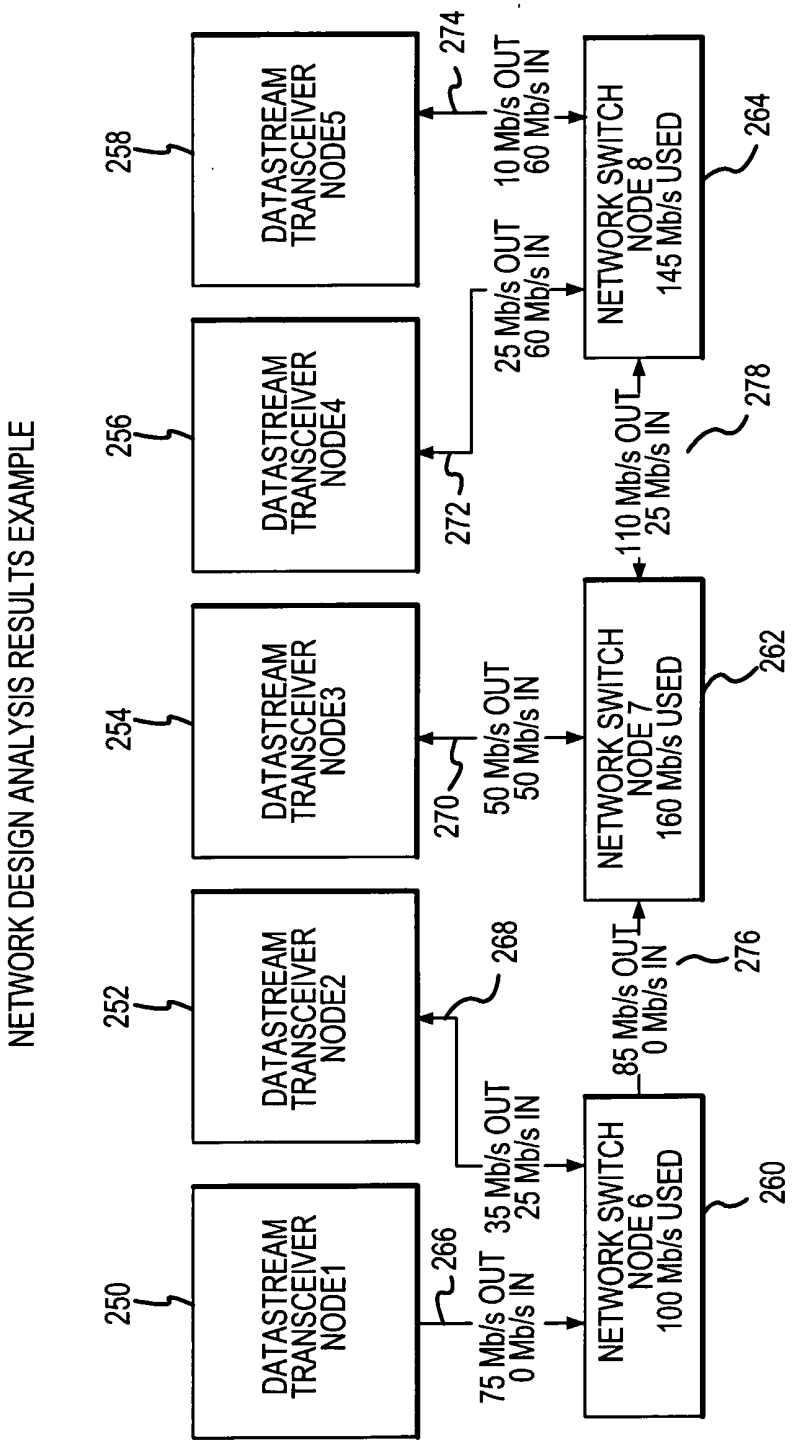

NETWORK DESIGN ANALYSIS RESULTS EXAMPLE
DATASTREAM TRANSCEIVER NODE1
DATASTREAM TRANSCEIVER NODE2
DATASTREAM TRANSCEIVER NODE3
DATASTREAM TRANSCEIVER NODE4
DATASTREAM TRANSCEIVER NODE5
NETWORK SWITCH NODE 6 100 Mb/s USED
NETWORK SWITCH NODE 7 160 Mb/s USED
NETWORK SWITCH NODE 8 145 Mb/s USED
250
252
254
256
258
260
262
264
266
268
270
272
274
276
278
75 Mb/s OUT 0 Mb/s IN
35 Mb/s OUT 25 Mb/s IN
50 Mb/s OUT 50 Mb/s IN
25 Mb/s OUT 60 Mb/s IN
10 Mb/s OUT 60 Mb/s IN
85 Mb/s OUT 0 Mb/s IN
110 Mb/s OUT 25 Mb/s IN

FIG. 6

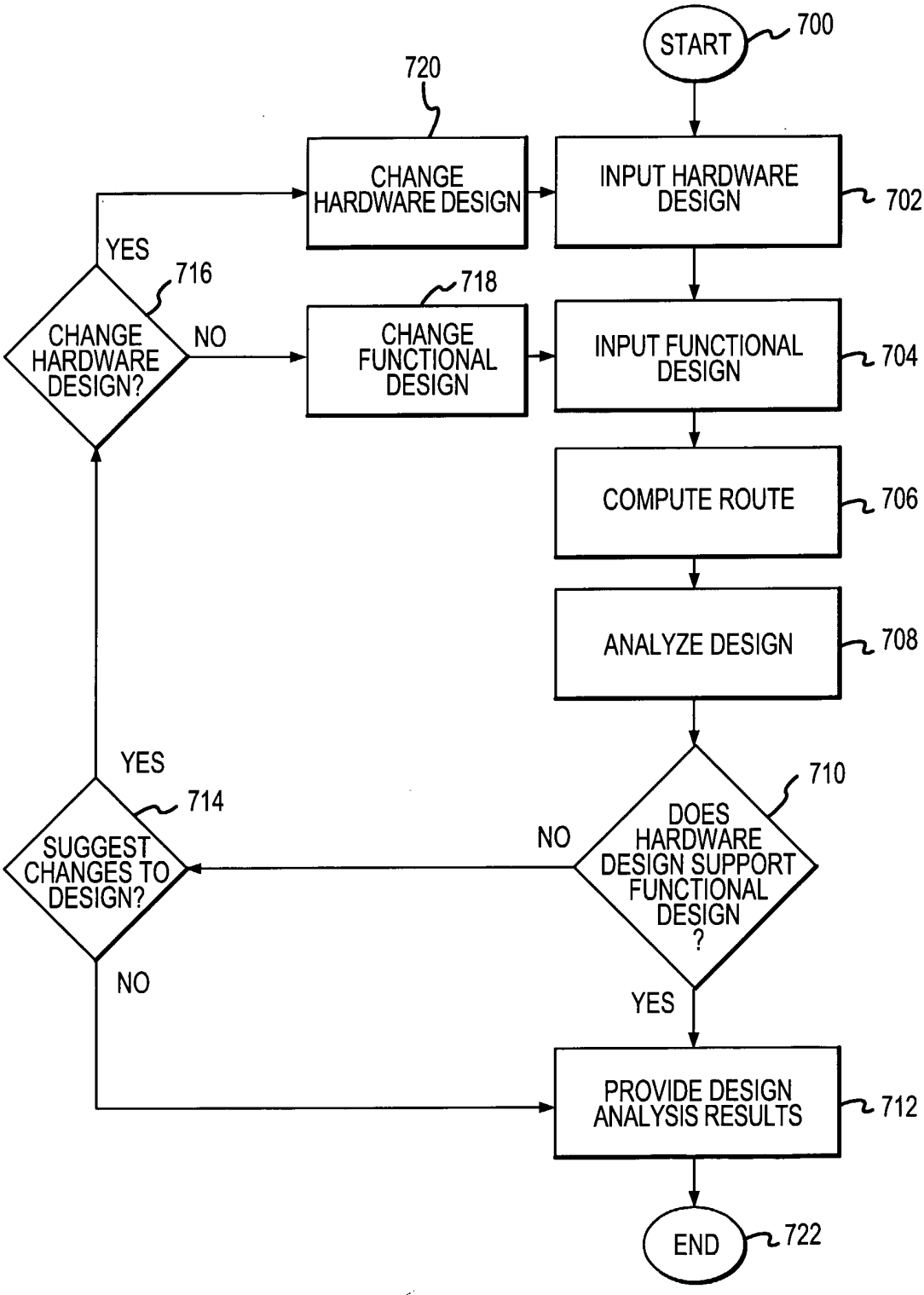

START
700
720
CHANGE HARDWARE DESIGN
INPUT HARDWARE DESIGN
702
YES
716
CHANGE HARDWARE DESIGN?
NO
718
CHANGE FUNCTIONAL DESIGN
INPUT FUNCTIONAL DESIGN
704
COMPUTE ROUTE
706
ANALYZE DESIGN
708
YES
714
SUGGEST CHANGES TO DESIGN?
NO
710
DOES HARDWARE DESIGN SUPPORT FUNCTIONAL DESIGN ?
NO
YES
PROVIDE DESIGN ANALYSIS RESULTS
712
END
722

FIG. 7

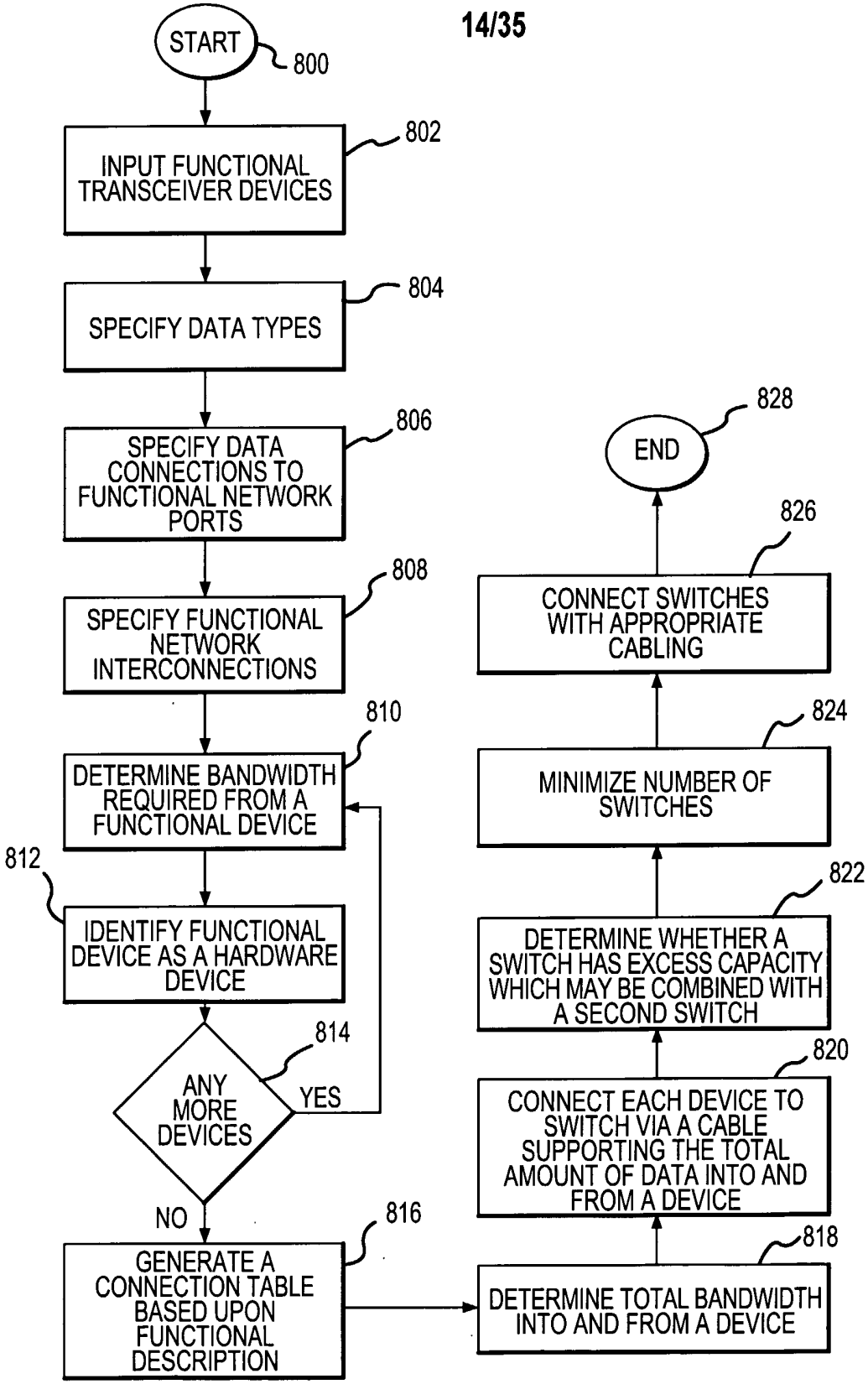

START
800
INPUT FUNCTIONAL TRANSCEIVER DEVICES
802
SPECIFY DATA TYPES
804
SPECIFY DATA CONNECTIONS TO FUNCTIONAL NETWORK PORTS
806
SPECIFY FUNCTIONAL NETWORK INTERCONNECTIONS
808
DETERMINE BANDWIDTH REQUIRED FROM A FUNCTIONAL DEVICE
810
IDENTIFY FUNCTIONAL DEVICE AS A HARDWARE DEVICE
812
ANY MORE DEVICES
814
YES
NO
GENERATE A CONNECTION TABLE BASED UPON FUNCTIONAL DESCRIPTION
816
DETERMINE TOTAL BANDWIDTH INTO AND FROM A DEVICE
818
CONNECT EACH DEVICE TO SWITCH VIA A CABLE SUPPORTING THE TOTAL AMOUNT OF DATA INTO AND FROM A DEVICE
820
DETERMINE WHETHER A SWITCH HAS EXCESS CAPACITY WHICH MAY BE COMBINED WITH A SECOND SWITCH
822
MINIMIZE NUMBER OF SWITCHES
824
CONNECT SWITCHES WITH APPROPRIATE CABLING
826
END
828

FIG. 8

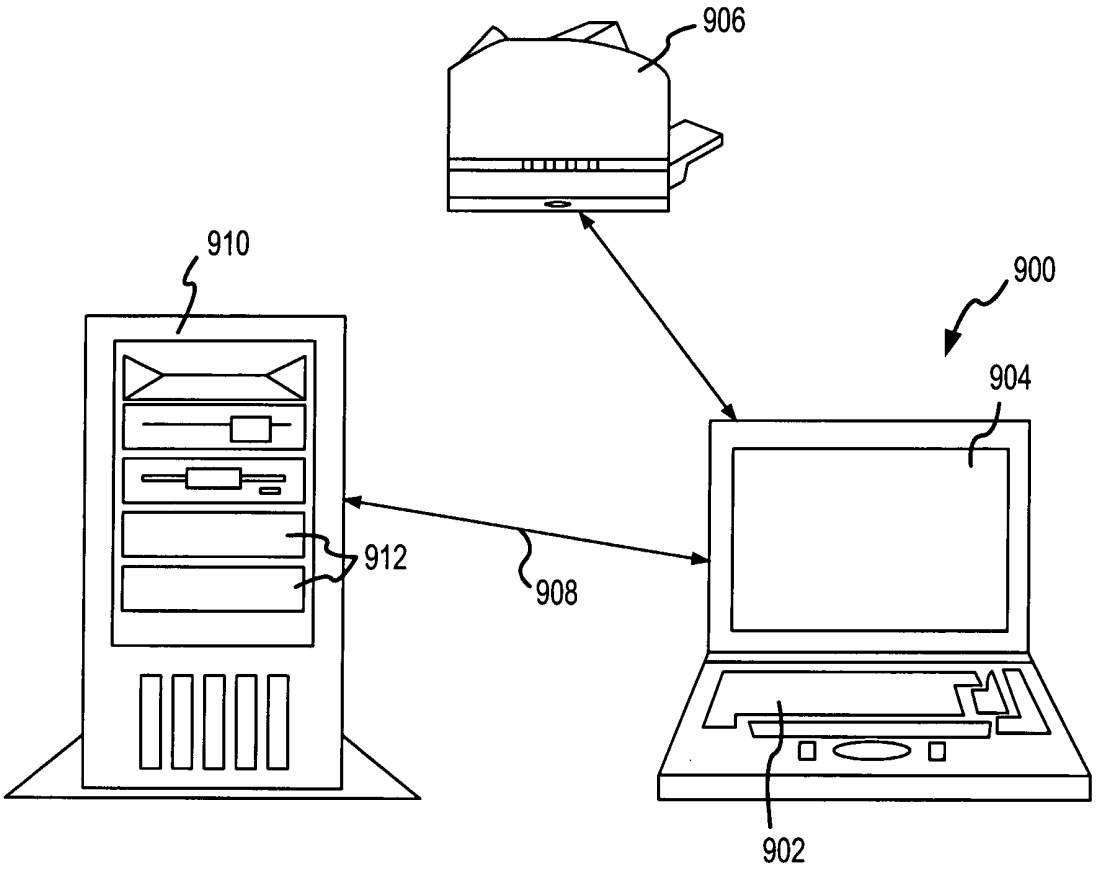


FIG.9

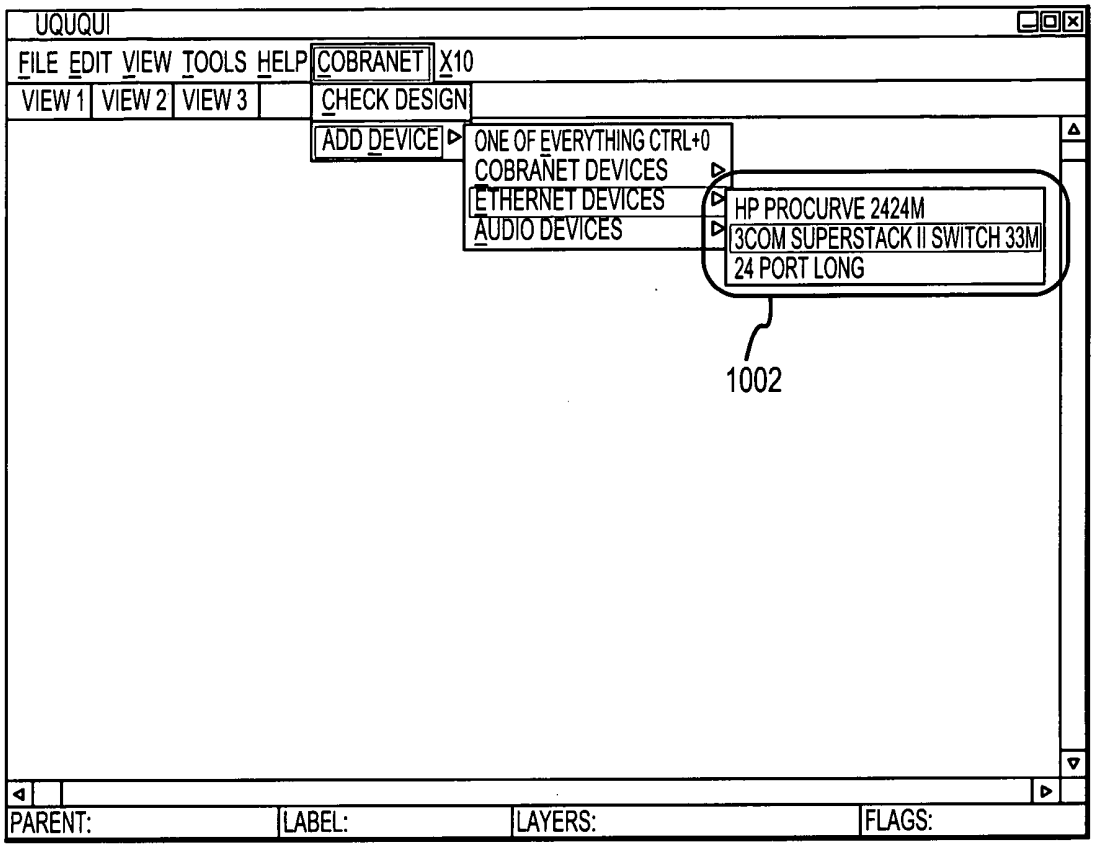

UQUQUI
FILE EDIT VIEW TOOLS HELP
COBRANET
X10
VIEW 1
VIEW 2
VIEW 3
CHECK DESIGN
ADD DEVICE
ONE OF EVERYTHING CTRL+0
COBRANET DEVICES
ETHERNET DEVICES
AUDIO DEVICES
HP PROCURVE 2424M
3COM SUPERSTACK II SWITCH 33M
24 PORT LONG
1002
PARENT:
LABEL:
LAYERS:
FLAGS:

FIG.10A

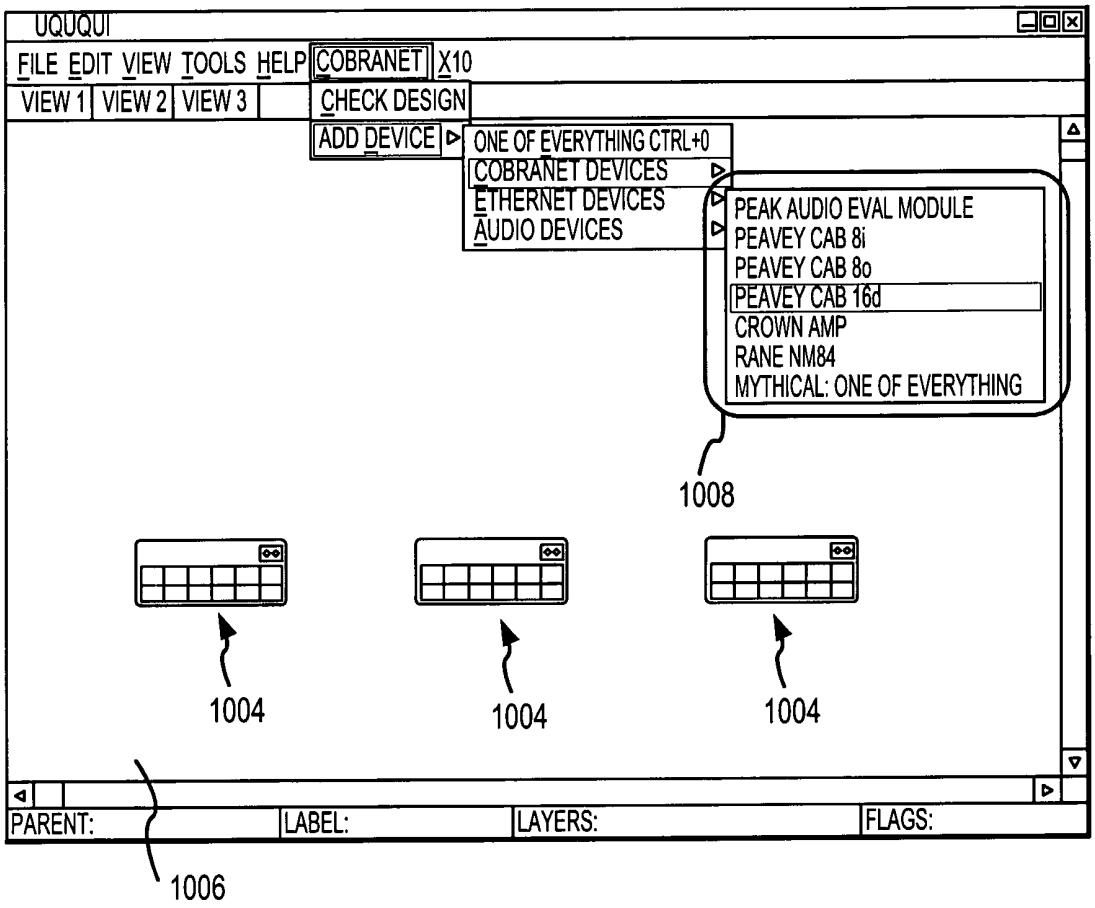

UQUQUI
FILE EDIT VIEW TOOLS HELP
COBRANET
X10
VIEW 1
VIEW 2
VIEW 3
CHECK DESIGN
ADD DEVICE
ONE OF EVERYTHING CTRL+0
COBRANET DEVICES
ETHERNET DEVICES
AUDIO DEVICES
PEAK AUDIO EVAL MODULE
PEAVEY CAB 8i
PEAVEY CAB 8o
PEAVEY CAB 16d
CROWN AMP
RANE NM84
MYTHICAL: ONE OF EVERYTHING
1008
1004
1004
1004
PARENT:
LABEL:
LAYERS:
FLAGS:
1006

FIG.10B

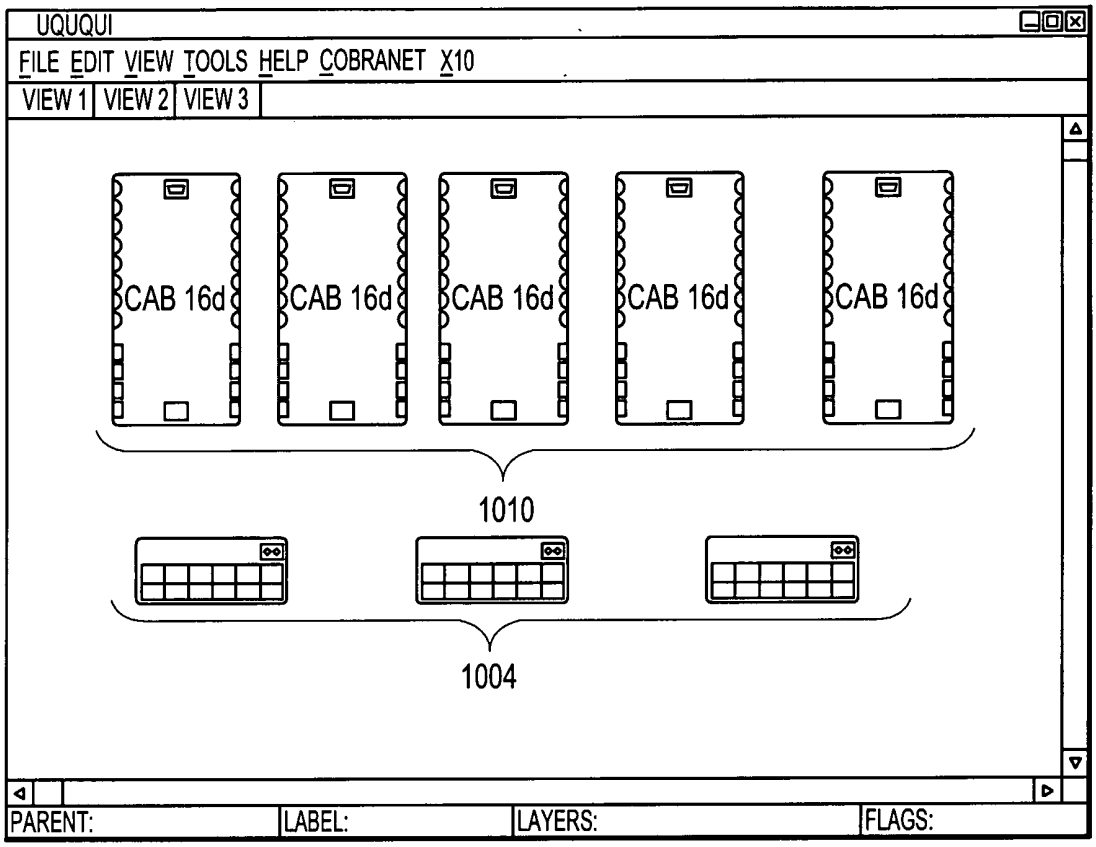

UQUQUI
FILE EDIT VIEW TOOLS HELP COBRANET X10
VIEW 1
VIEW 2
VIEW 3
CAB 16d
CAB 16d
CAB 16d
CAB 16d
CAB 16d
1010
1004
PARENT:
LABEL:
LAYERS:
FLAGS:

FIG.10C

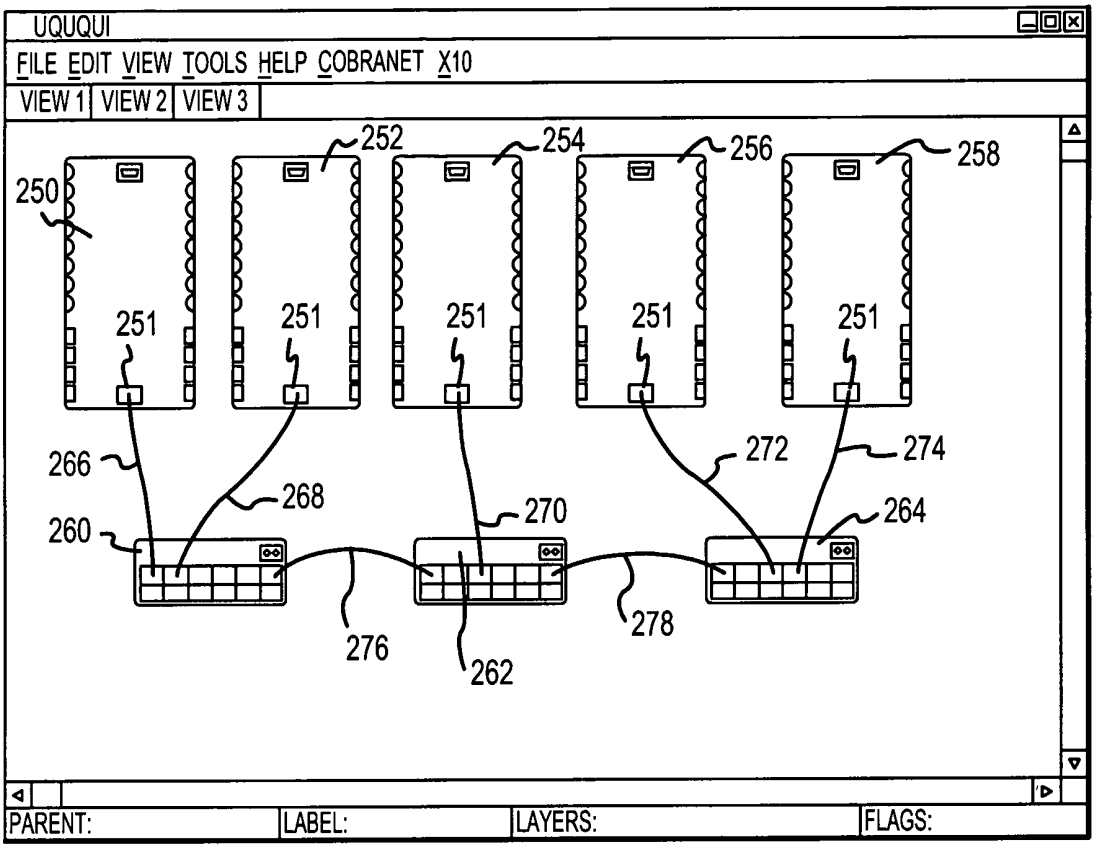

UQUQUI
FILE EDIT VIEW TOOLS HELP COBRANET X10
VIEW 1 VIEW 2 VIEW 3
250
251
252
254
256
258
266
268
260
270
272
274
264
276
262
278
PARENT:
LABEL:
LAYERS:
FLAGS:

FIG.10D

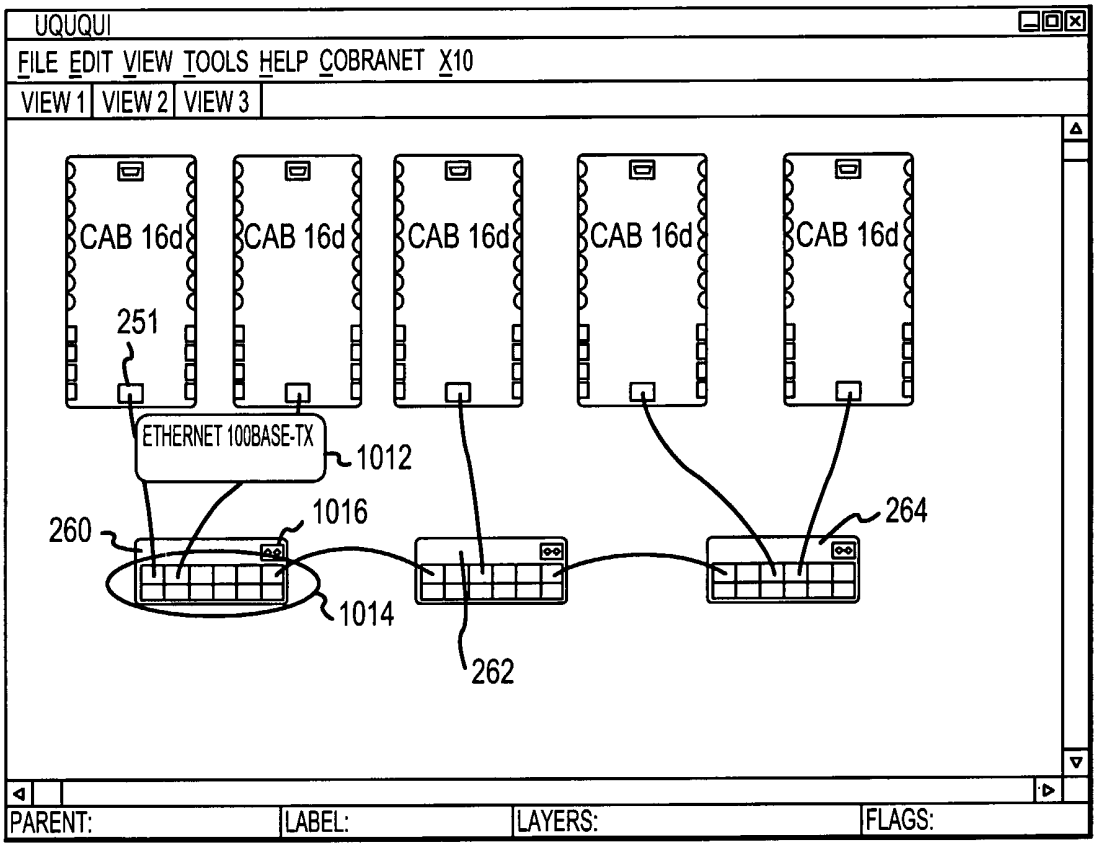

UQUQUI
FILE EDIT VIEW TOOLS HELP COBRANET X10
VIEW 1
VIEW 2
VIEW 3
CAB 16d
CAB 16d
CAB 16d
CAB 16d
CAB 16d
251
ETHERNET 100BASE-TX
1012
1016
260
264
1014
262
PARENT:
LABEL:
LAYERS:
FLAGS:

FIG.10E

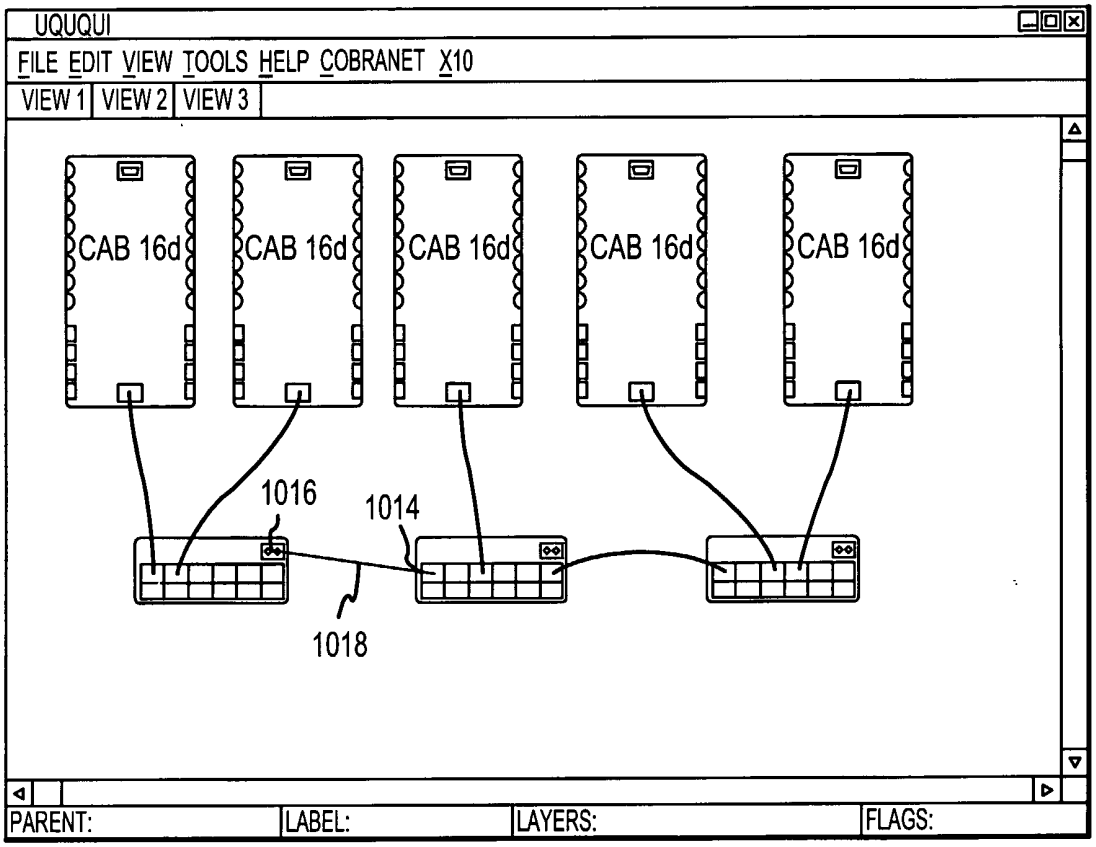

UQUQUI
FILE EDIT VIEW TOOLS HELP COBRANET X10
VIEW 1
VIEW 2
VIEW 3
CAB 16d
CAB 16d
CAB 16d
CAB 16d
CAB 16d
1016
1014
1018
PARENT:
LABEL:
LAYERS:
FLAGS:

FIG.10F

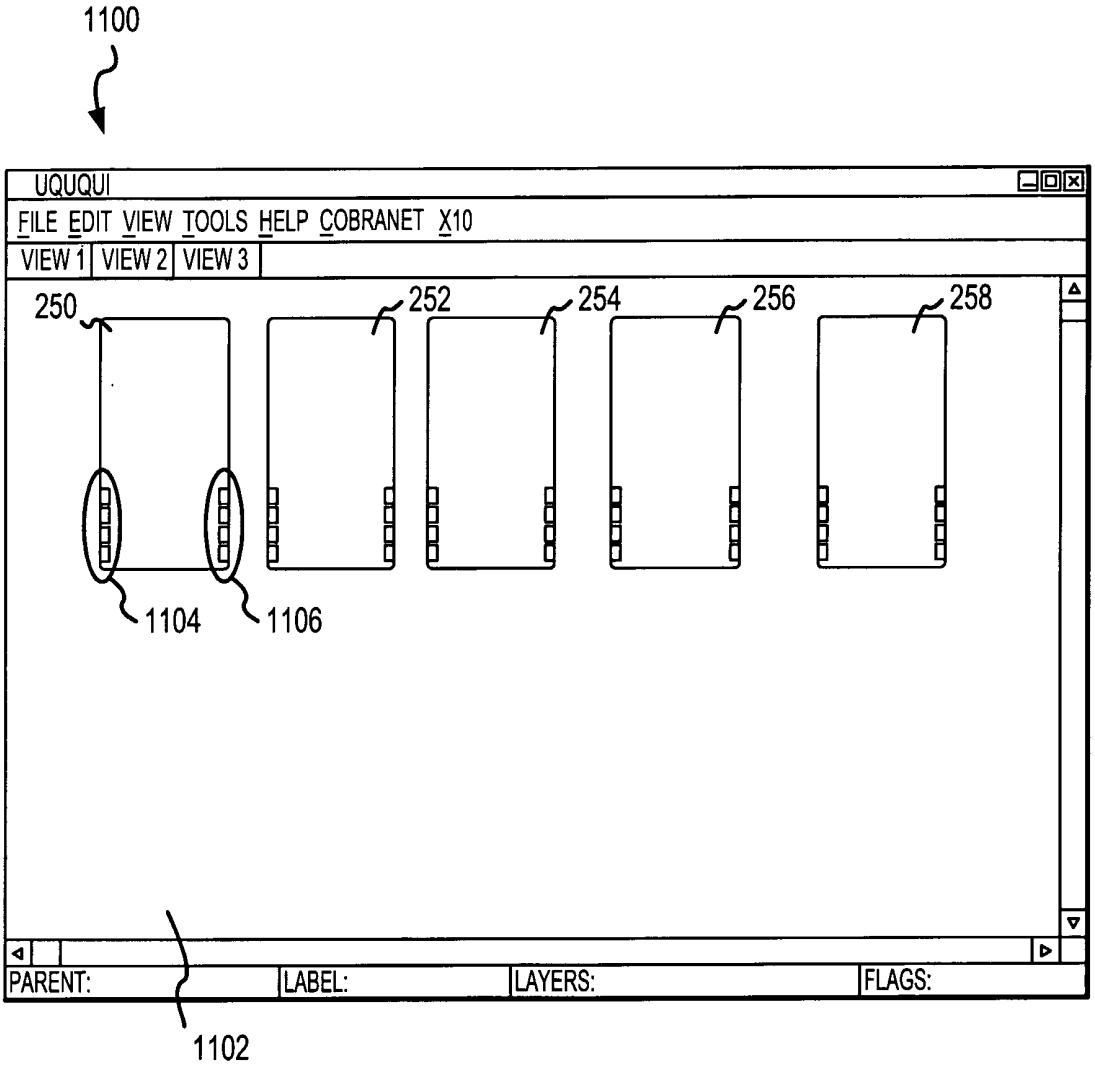

1100
UQUQUI
FILE EDIT VIEW TOOLS HELP COBRANET X10
VIEW 1 VIEW 2 VIEW 3
250
252
254
256
258
1104
1106
PARENT:
LABEL:
LAYERS:
FLAGS:
1102

FIG.11A

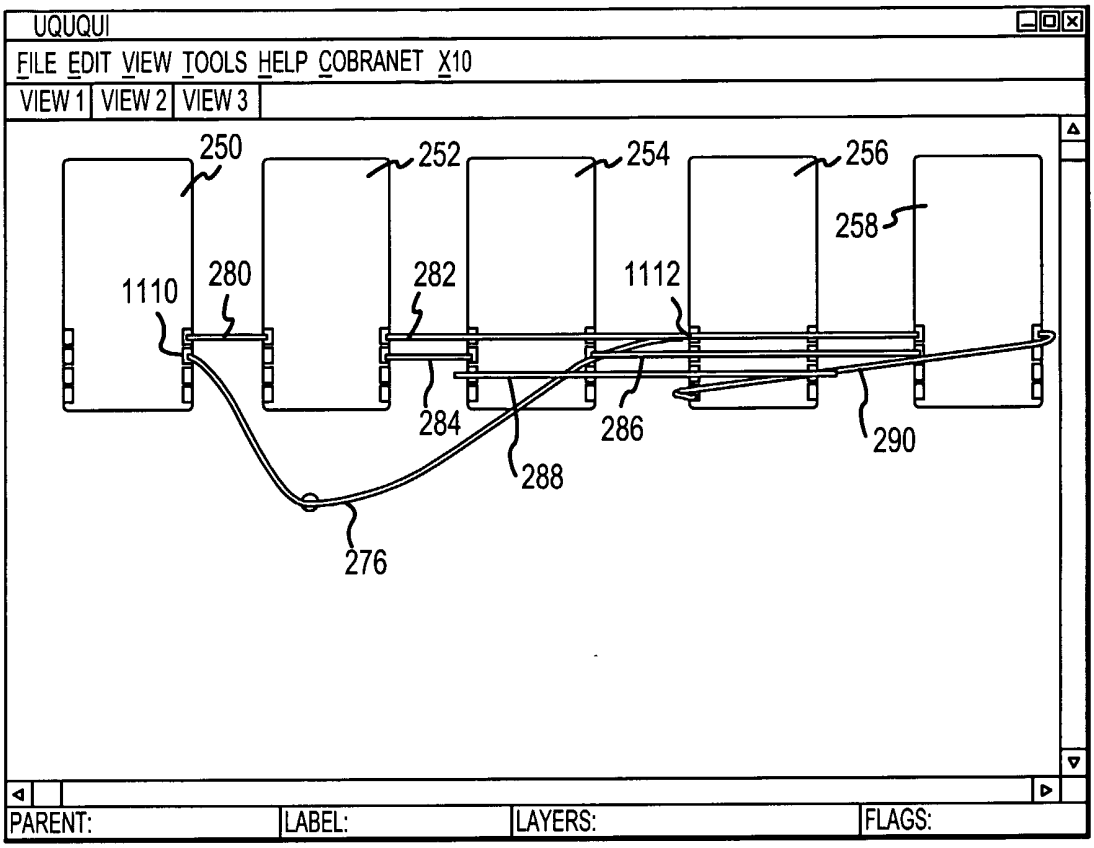

UQUQUI
FILE EDIT VIEW TOOLS HELP COBRANET X10
VIEW 1
VIEW 2
VIEW 3
250
252
254
256
258
1110
280
282
1112
284
286
288
290
276
PARENT:
LABEL:
LAYERS:
FLAGS:

FIG.11B

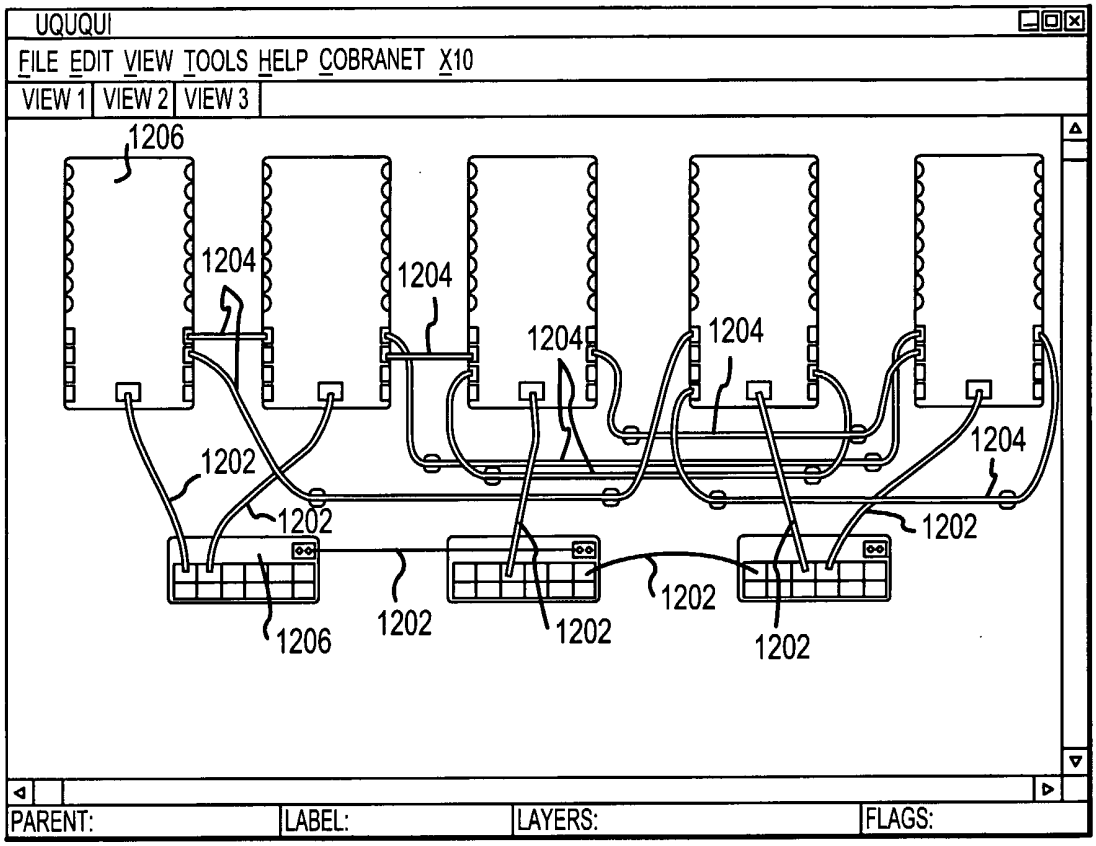

UQUQUI
FILE EDIT VIEW TOOLS HELP COBRANET X10
VIEW 1
VIEW 2
VIEW 3
1206
1204
1204
1204
1204
1204
1202
1202
1202
1202
1202
1202
1206
1202
1202
PARENT:
LABEL:
LAYERS:
FLAGS:

FIG.12

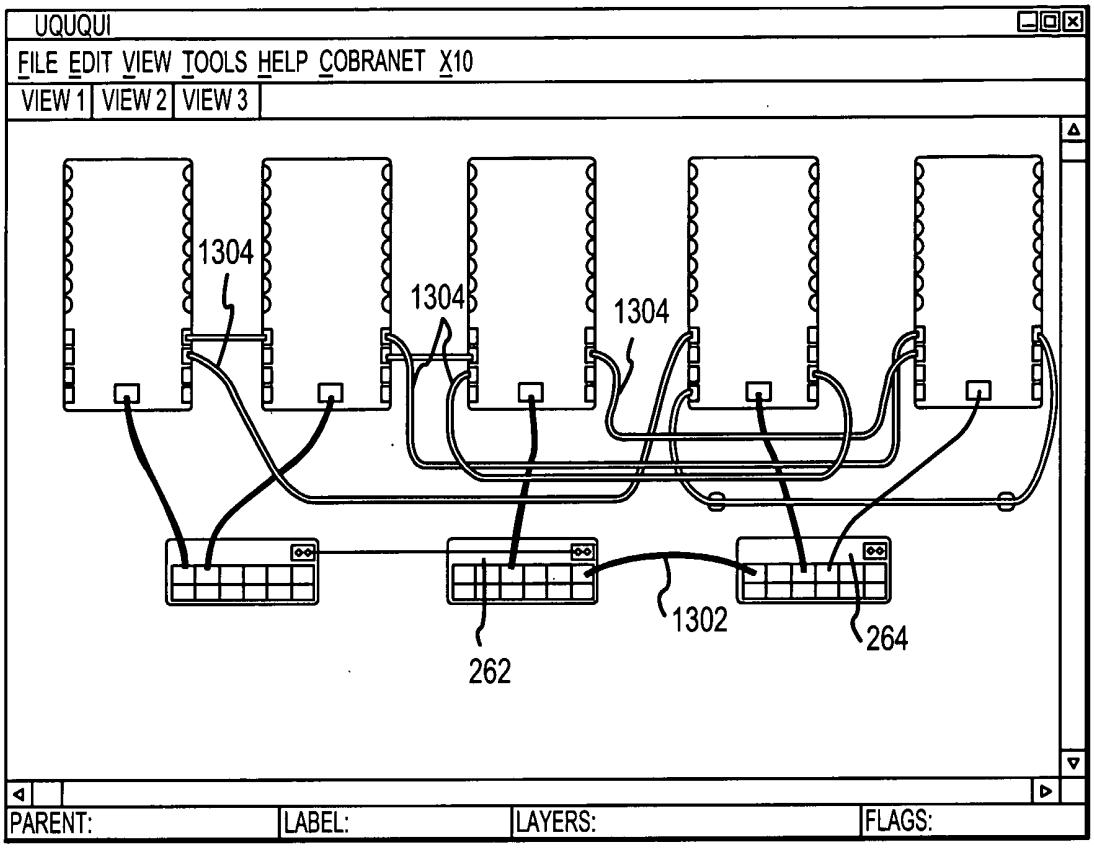

UQUQUI
FILE EDIT VIEW TOOLS HELP COBRANET X10
VIEW 1
VIEW 2
VIEW 3
1304
1304
1304
1302
262
264
PARENT:
LABEL:
LAYERS:
FLAGS:

FIG.13

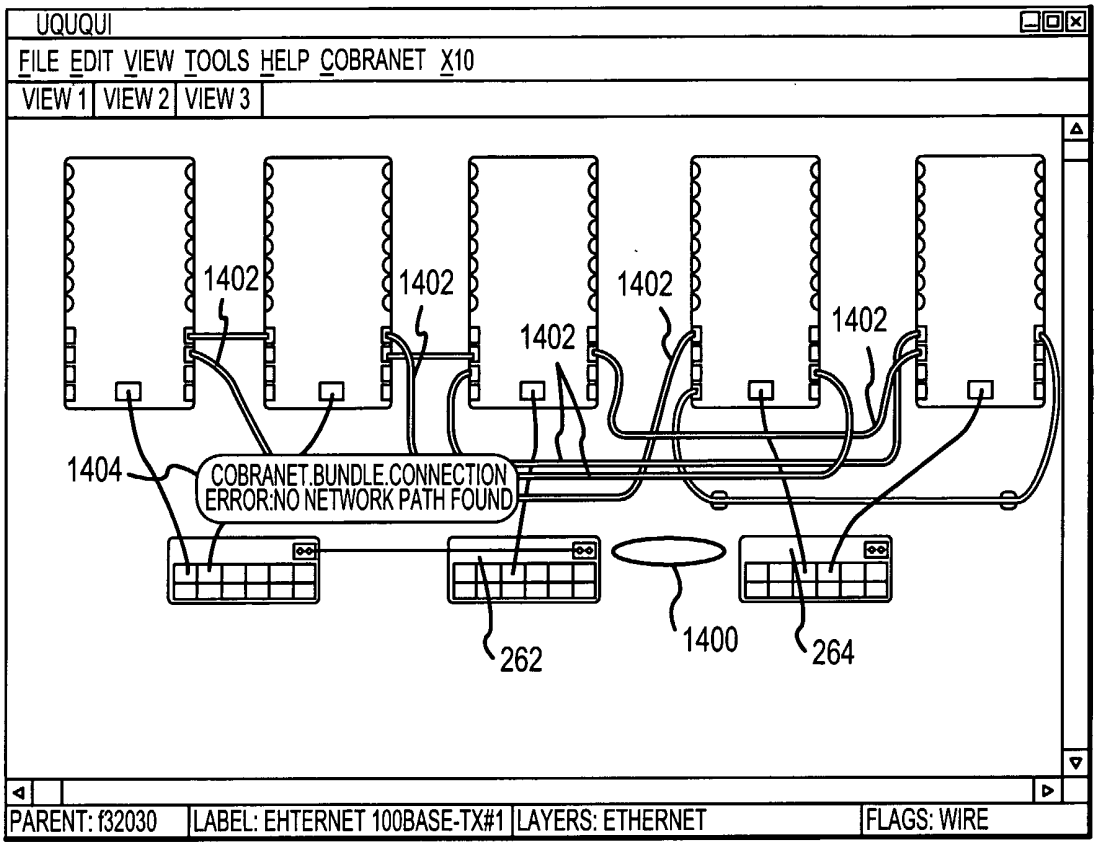

UQUQUI
FILE EDIT VIEW TOOLS HELP COBRANET X10
VIEW 1
VIEW 2
VIEW 3
1402
1402
1402
1402
1402
1404
COBRANET.BUNDLE.CONNECTION
ERROR:NO NETWORK PATH FOUND
262
1400
264
PARENT: f32030
LABEL: EHTERNET 100BASE-TX#1
LAYERS: ETHERNET
FLAGS: WIRE

FIG.14A

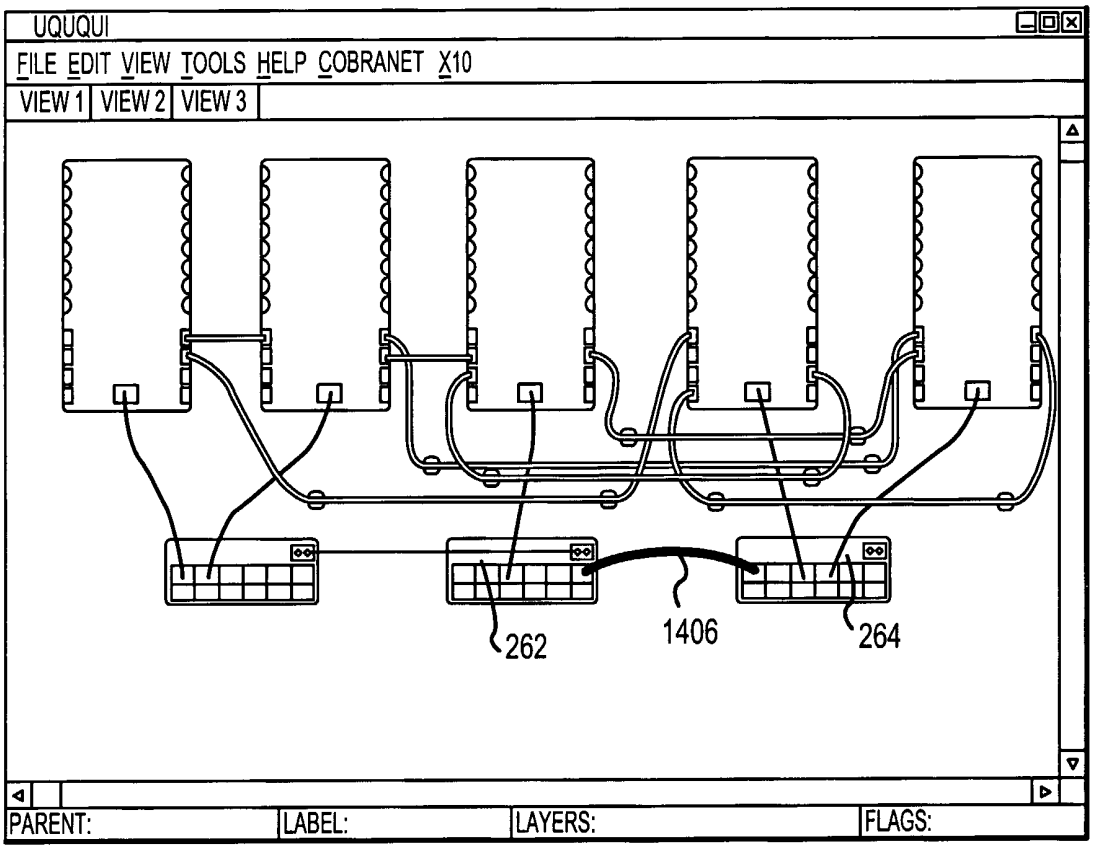

UQUQUI
FILE EDIT VIEW TOOLS HELP COBRANET X10
VIEW 1 VIEW 2 VIEW 3
262
1406
264
PARENT:
LABEL:
LAYERS:
FLAGS:

FIG.14B

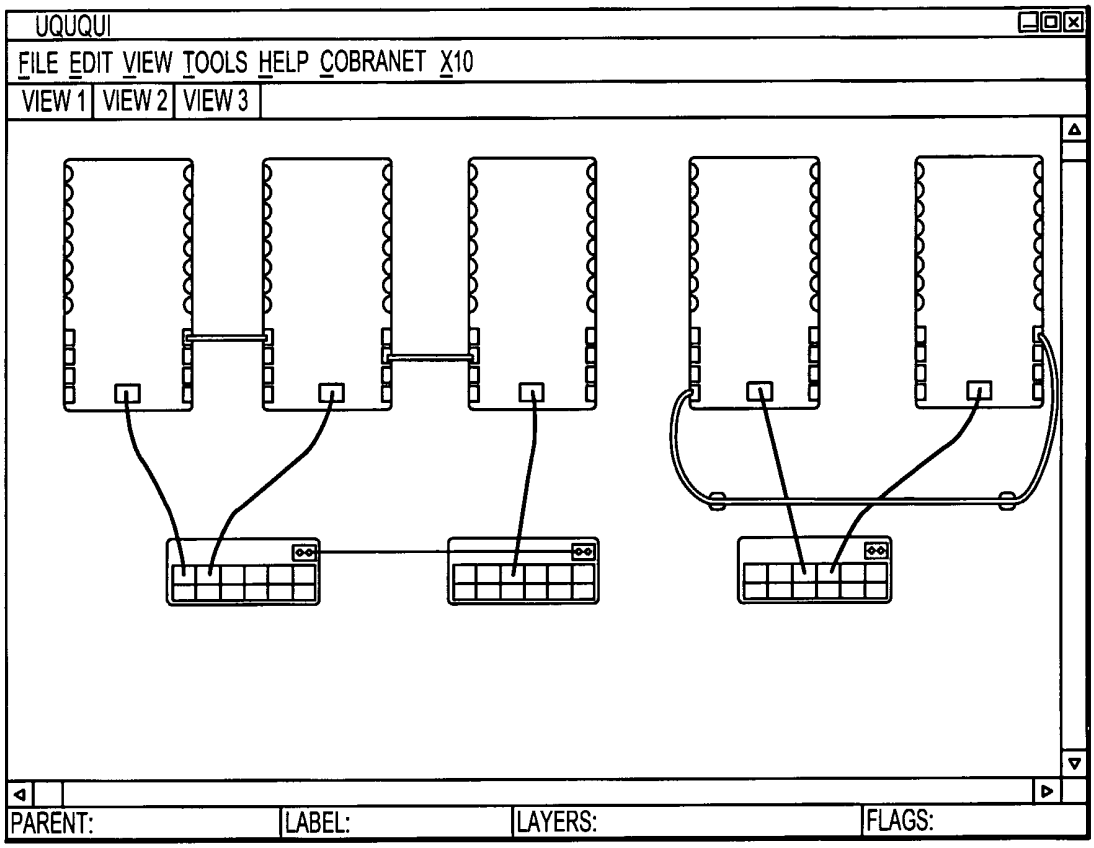

UQUQUI
FILE EDIT VIEW TOOLS HELP COBRANET X10
VIEW 1
VIEW 2
VIEW 3
PARENT:
LABEL:
LAYERS:
FLAGS:

FIG.14C

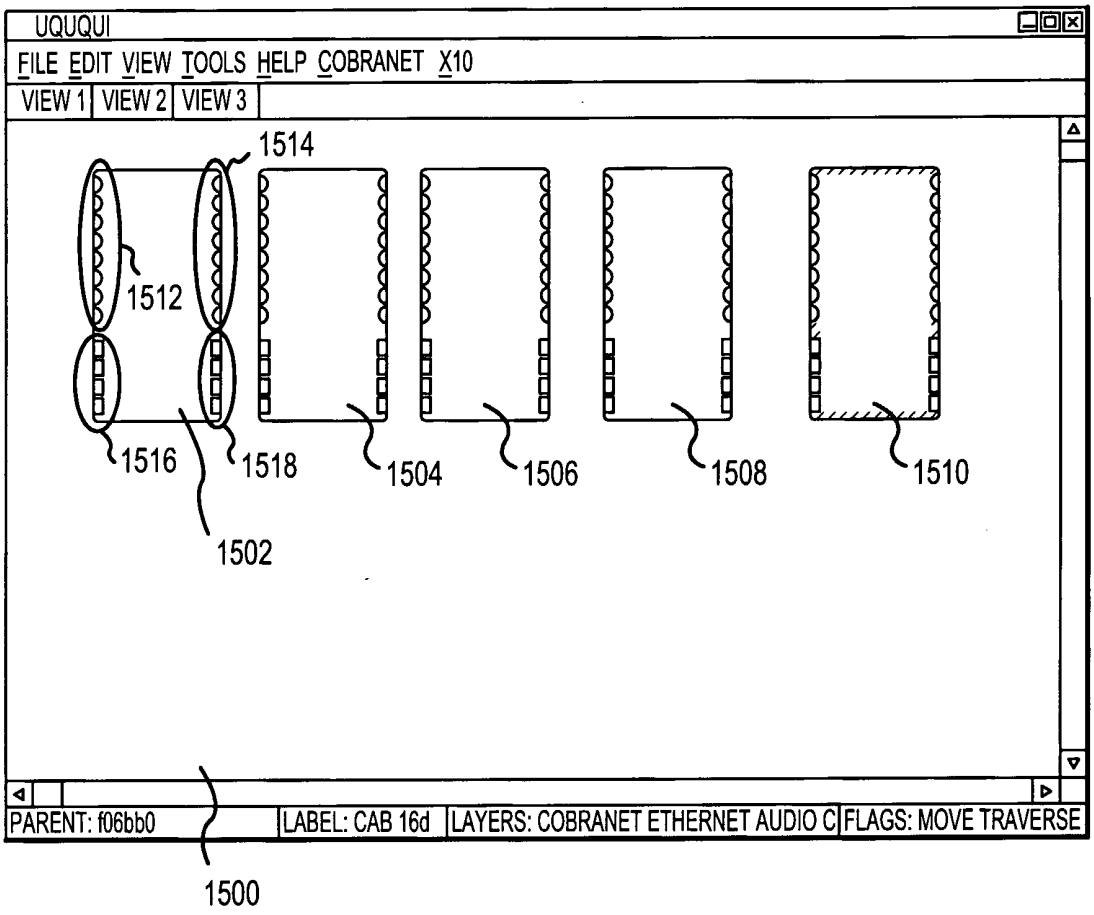

UQUQUI
FILE EDIT VIEW TOOLS HELP COBRANET X10
VIEW 1
VIEW 2
VIEW 3
1514
1512
1516
1518
1504
1506
1508
1510
1502
PARENT: f06bb0
LABEL: CAB 16d
LAYERS: COBRANET ETHERNET AUDIO C
FLAGS: MOVE TRAVERSE
1500

FIG.15A

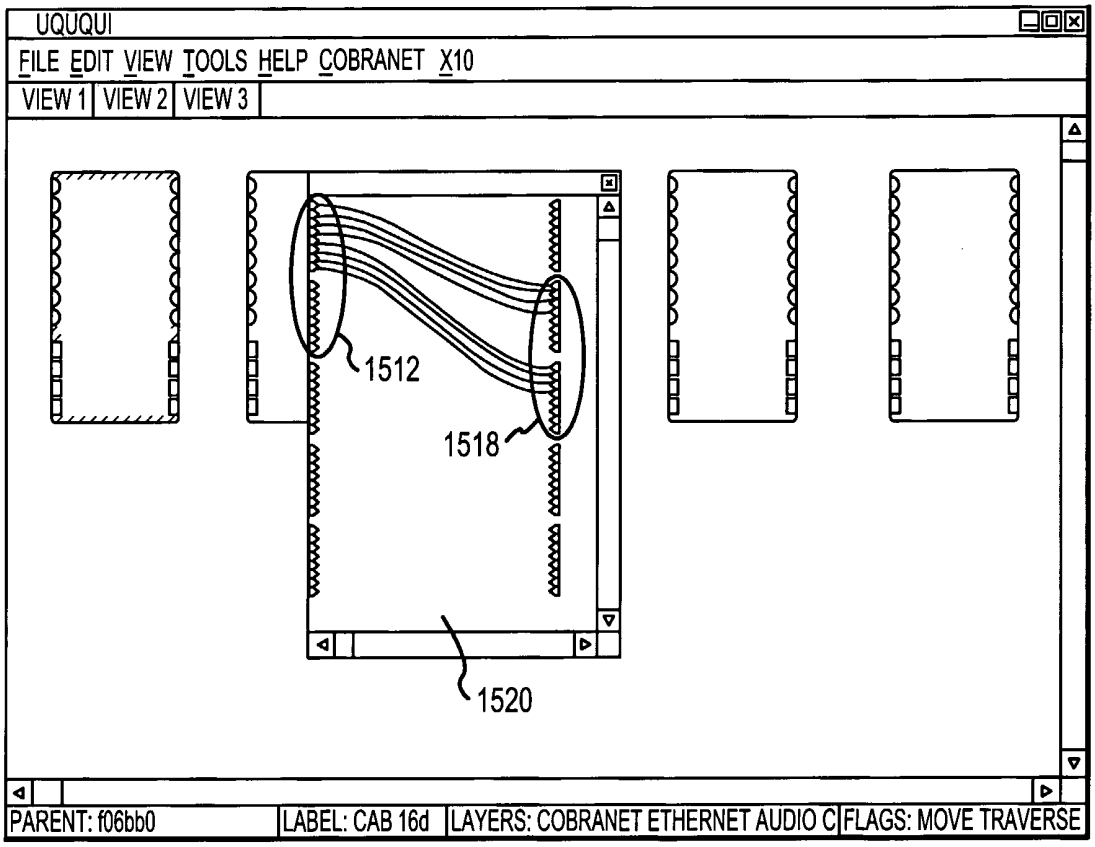

UQUQUI
FILE EDIT VIEW TOOLS HELP COBRANET X10
VIEW 1
VIEW 2
VIEW 3
1512
1518
1520
PARENT: f06bb0
LABEL: CAB 16d
LAYERS: COBRANET ETHERNET AUDIO C
FLAGS: MOVE TRAVERSE

FIG.15B

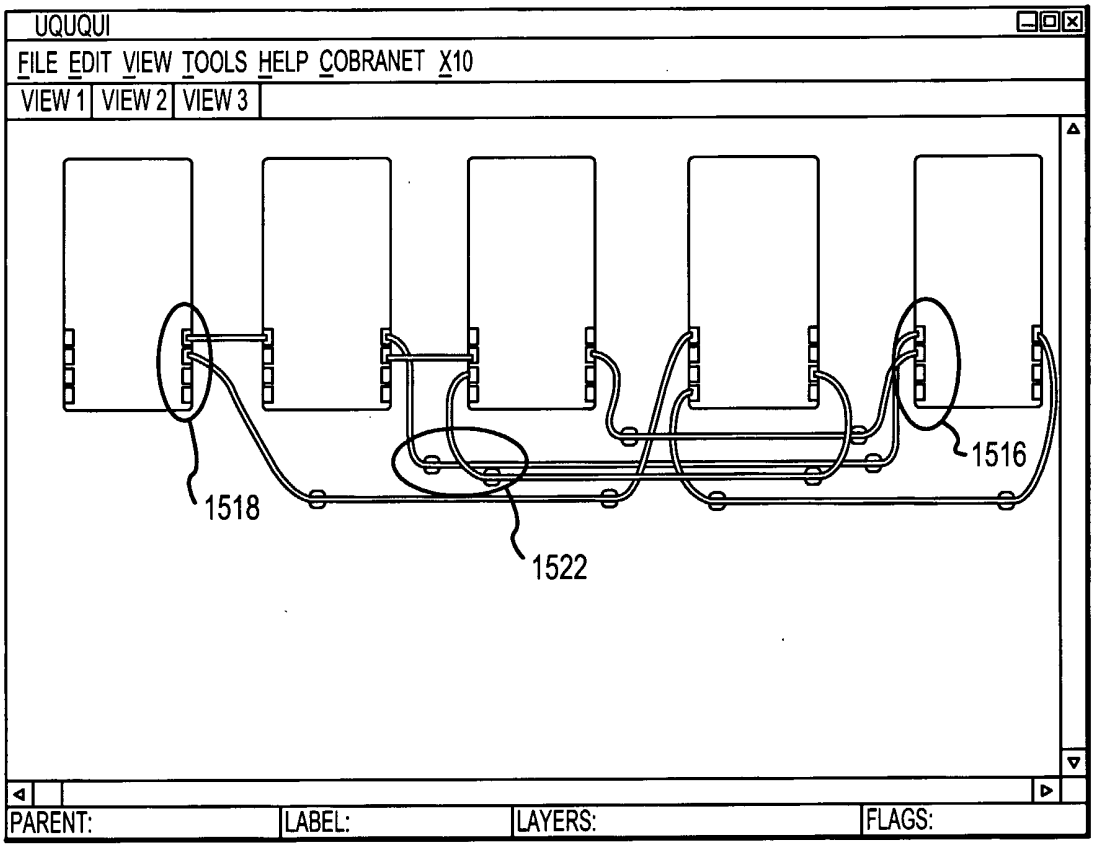

UQUQUI
FILE EDIT VIEW TOOLS HELP COBRANET X10
VIEW 1
VIEW 2
VIEW 3
1518
1522
1516
PARENT:
LABEL:
LAYERS:
FLAGS:

FIG.15C

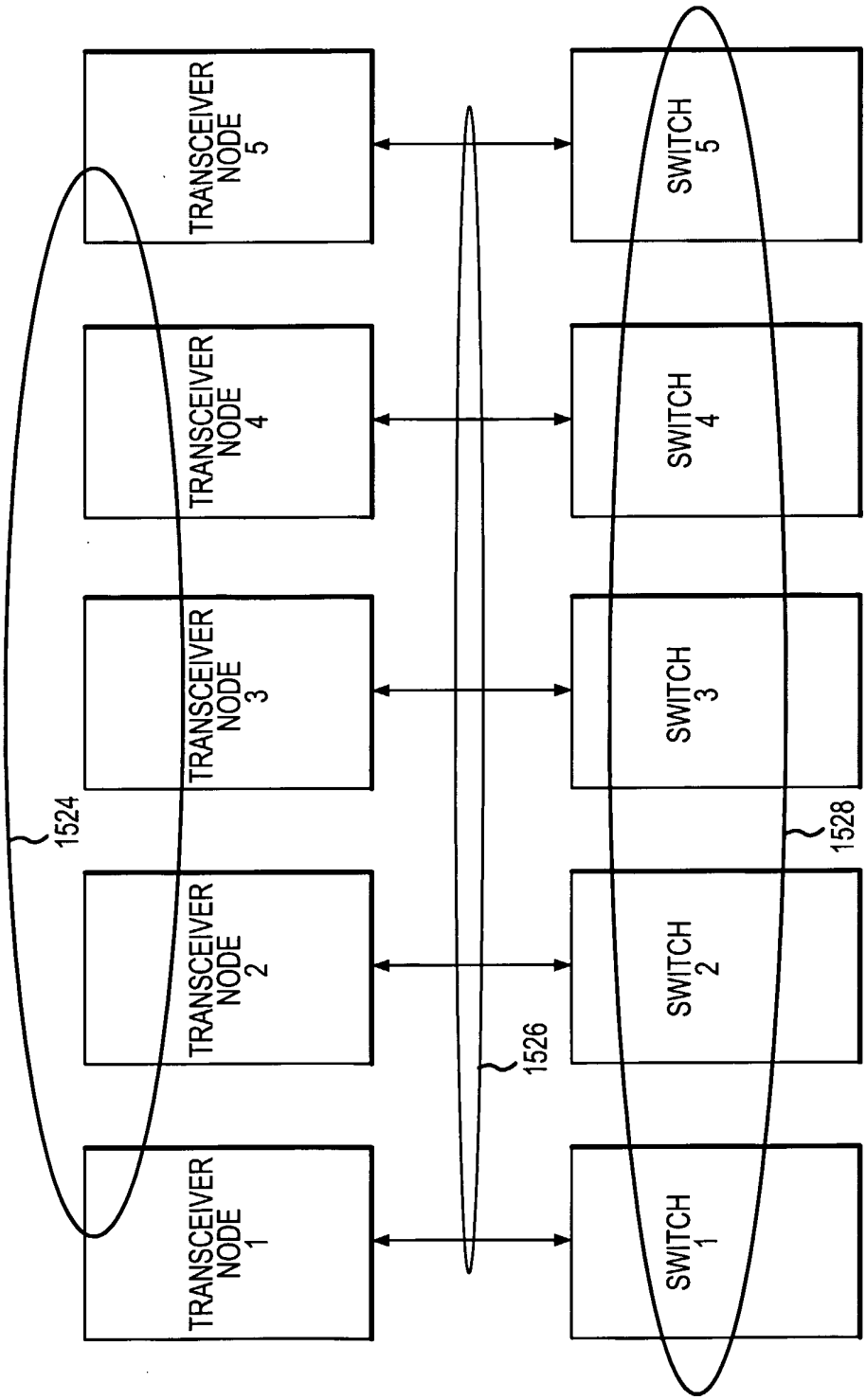

TRANSCEIVER NODE 5
TRANSCEIVER NODE 4
TRANSCEIVER NODE 3
TRANSCEIVER NODE 2
TRANSCEIVER NODE 1
SWITCH 5
SWITCH 4
SWITCH 3
SWITCH 2
SWITCH 1
1524
1526
1528

FIG. 15D

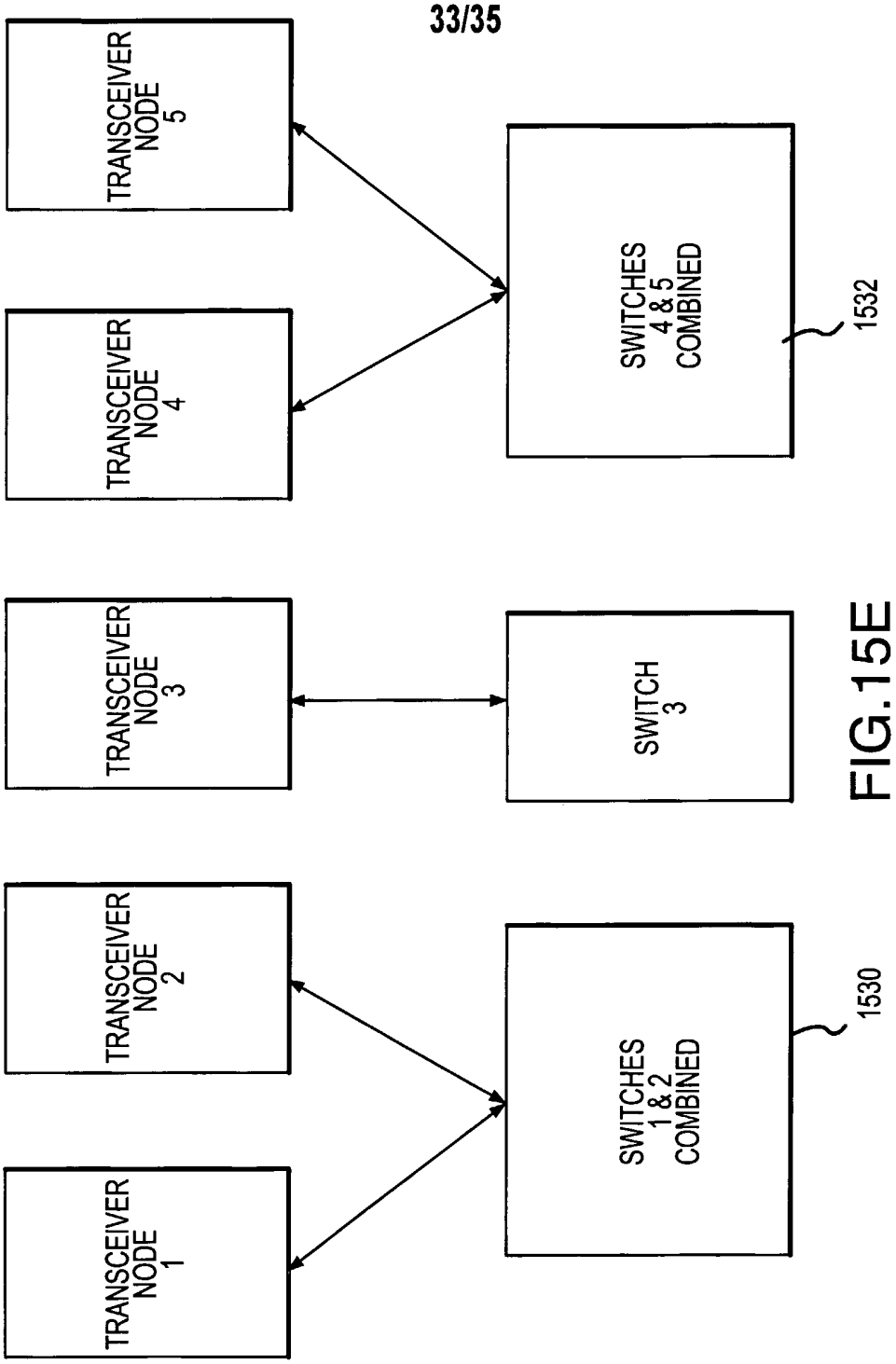

TRANSCEIVER NODE 1
TRANSCEIVER NODE 2
SWITCHES 1 & 2 COMBINED
1530
TRANSCEIVER NODE 3
SWITCH 3
TRANSCEIVER NODE 4
TRANSCEIVER NODE 5
SWITCHES 4 & 5 COMBINED
1532

FIG.15E

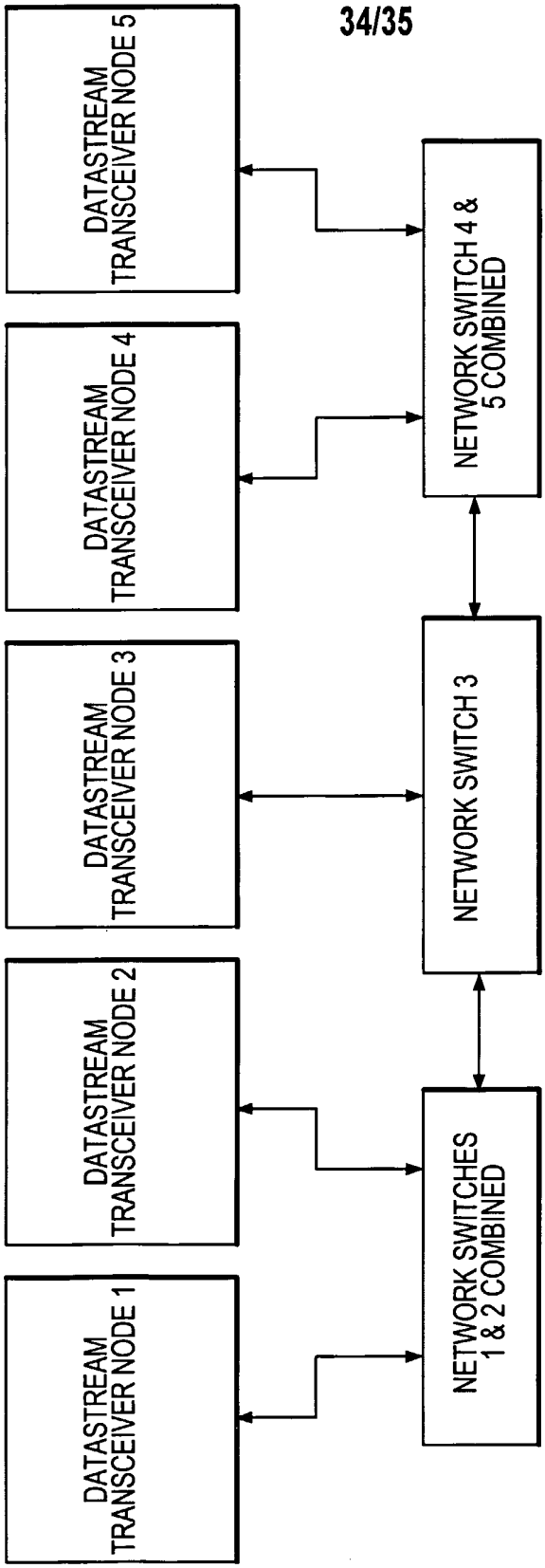

DATASTREAM TRANSCEIVER NODE 5
DATASTREAM TRANSCEIVER NODE 4
DATASTREAM TRANSCEIVER NODE 3
DATASTREAM TRANSCEIVER NODE 2
DATASTREAM TRANSCEIVER NODE 1
NETWORK SWITCH 4 & 5 COMBINED
NETWORK SWITCH 3
NETWORK SWITCHES 1 & 2 COMBINED

FIG. 15F

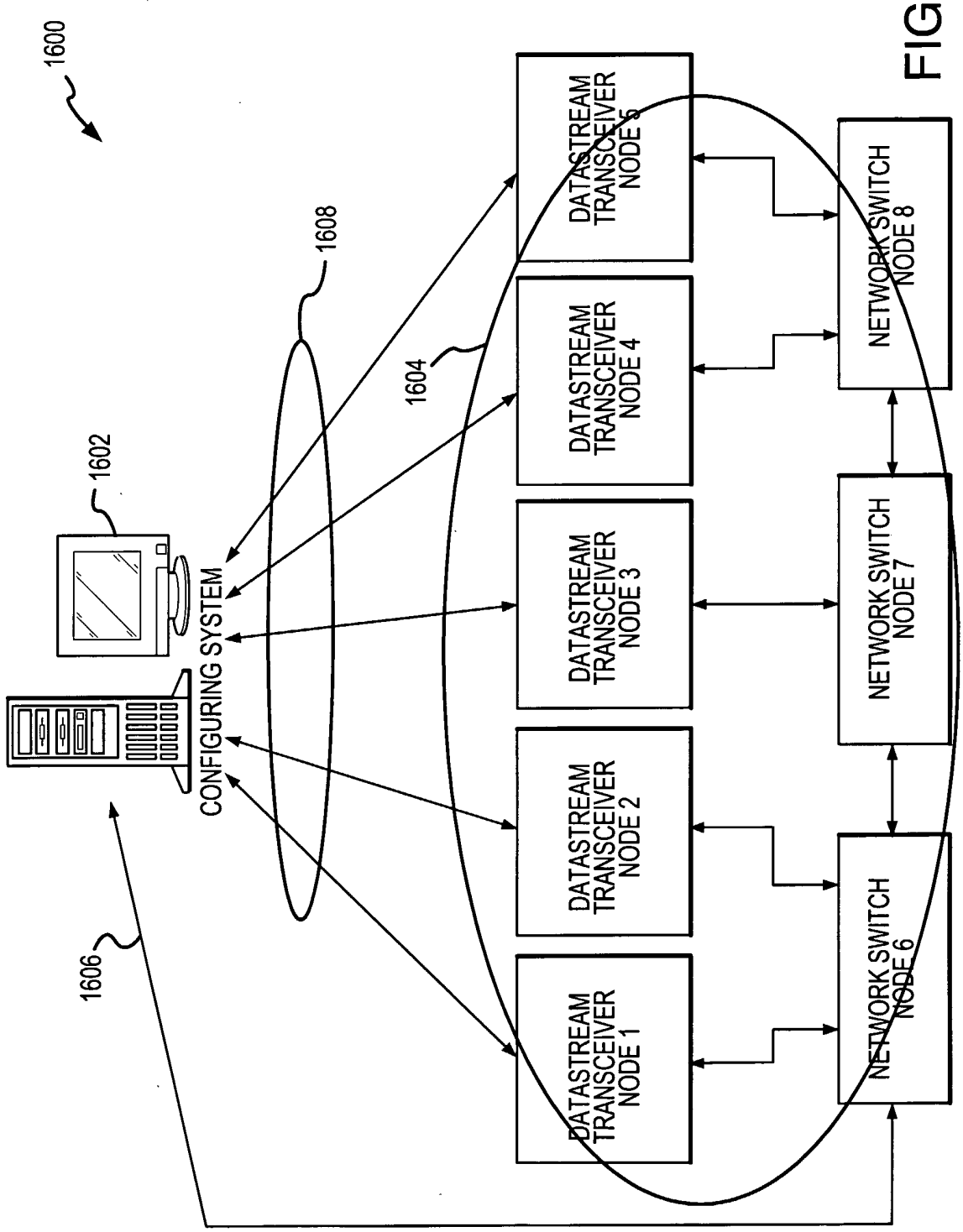

1600
1602
1604
1606
1608
CONFIGURING SYSTEM
DATASTREAM TRANSCEIVER NODE 1
DATASTREAM TRANSCEIVER NODE 2
DATASTREAM TRANSCEIVER NODE 3
DATASTREAM TRANSCEIVER NODE 4
DATASTREAM TRANSCEIVER NODE 5
NETWORK SWITCH NODE 6
NETWORK SWITCH NODE 7
NETWORK SWITCH NODE 8

FIG. 16