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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 November 2003 (27.11.2003)

PCT

(10) International Publication Number
WO 03/098848 A2

(51) International Patent Classification[7]: H04J

(21) International Application Number: PCT/US03/15216

(22) International Filing Date: 14 May 2003 (14.05.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/146,989 15 May 2002 (15.05.2002) US

(71) Applicant: **SONICS, INC.** [US/US]; Suite 620, 2440 W. El Camino Real, Mountain View, CA 94040 (US).

(72) Inventors: **MEYER, Michael, J.**; 1575 Edgewood Drive, Palo Alto, CA 94303 (US). **EVANS, Scott, C.**; 252 Hayes Avenue, Santa Clara, CA 95051 (US). **SYNEK, Kamil**; 1513 Mallard Way, Sunnyvale, CA 94087 (US).

(74) Agents: **MALLIE, Michael, J.** et al.; Blakely, Sokoloff, Taylor & Zafman LLP, 7th Floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025 (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, EC, 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, OM, PH, PL, PT, RO, RU, SD, SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

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.

WO 03/098848 A2

(54) Title: METHOD AND APPARATUS FOR OPTIMIZING DISTRIBUTED MULTIPLEXED BUS INTERCONNECTS

(57) Abstract: A method and apparatus for optimizing distributed multiplexed bus interconnects are described.

METHOD AND APPARATUS FOR OPTIMIZING DISTRIBUTED MULTIPLEXED BUS INTERCONNECTS

FIELD OF THE INVENTION

[0001] The present invention pertains to interconnections. More particularly, the present invention relates to a method and apparatus for optimizing distributed multiplexed bus interconnects.

BACKGROUND OF THE INVENTION

[0002] In computer networks, internetworking, communications, integrated circuits, etc. where there is a need to communicate information, there are often interconnections established to facilitate the transfer of the information. One approach is to use dedicated communication "lines" or links to transfer the information. A bus is usually used when more than two devices need to communicate. A traditional way to implement buses is using tristate bus drivers, where one device drives the bus and other drivers are disabled. Another approach is to have each device use a different set of wires and then to use a multiplexer to select the set of wires of the enabled device.

[0003] However, in multiplexing a bus, there may be communication points that may not need the full capabilities of the bus. Thus, extending a full bus to these entities may be wasteful of resources, such as space, power, etc. This may present a problem.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings, in which like references indicate similar elements and in which:

[0005] Figure 1A and Figure 1B illustrate traditional buses;

[0006] Figure 2A, Figure 2B, and Figure 2C illustrate embodiments of the present invention showing a topology;

[0007] Figure 3A illustrates one embodiment of the present invention showing receivers and transmitters for a specific bus signal;

[0008] Figure 3B illustrates one embodiment of the present invention showing unoptimized connections;

[0009] Figure 3C illustrates one embodiment of the present invention showing a topology after a combiner optimization;

[0010] Figure 3D illustrates one embodiment of the present invention showing a topology after a repeater optimization;

[0011] Figure 3E illustrates one embodiment of the present invention showing a topology after a root optimization;

[0012] Figure 4A and 4B illustrate embodiments of the present invention showing interconnects and logic details;

[0013] Figure 5 illustrates one embodiment of the present invention showing timing paths;

[0014] Figure 6A illustrates one embodiment of the present invention showing a flowchart for optimizing bus signal wiring;

[0015] Figure 6B illustrates one embodiment of the present invention showing a flowchart for optimizing combiners;

[0016] Figure 6C illustrates one embodiment of the present invention showing a flowchart for optimizing bus repeaters;

[0017] Figure 6D illustrates one embodiment of the present invention showing a flowchart for optimizing the root;

[0018] Figure 7A illustrates one embodiment of the present invention showing a combiner block;

[0019] Figure 7B illustrates another embodiment of the present invention showing a combiner block in more detail;

[0020] Figure 7C illustrates one embodiment of the present invention showing a truth Table 1 for checking bus conflicts;

[0021] Figure 8 illustrates various embodiments of the present invention showing optimizing timing constraints;

[0022] Figure 9 illustrates a network environment in which the method and apparatus of the present invention may be implemented; and

[0023] Figure 10 is a block diagram of a computer system.

DETAILED DESCRIPTION

[0024] A method and apparatus for optimizing distributed multiplexed bus interconnects are described.

[0025] The term IP as used in this document denotes Intellectual Property. The term IP, may be used by itself, or may be used with other terms such as core, to denote a design having a functionality. For example, an IP core or IP for short, may consist of

circuitry, buses, communication links, a microprocessor, etc. Additionally, IP may be implemented in a variety of ways, and may be fabricated on an integrated circuit, etc.

[0026] Buses have traditionally been thought of as a string of blocks or connectors connected in a manner shown in Figure 1A or Figure 1B. This approach is simple and easy to implement. In Integrated Circuit (IC) design three-state (tri-states) (Figure 1A) may not be desirable because of the difficulty in adding repeaters to wires that have multiple drivers and buses are commonly implemented via multiplexers. A multiplexer implementation requires a wire from each transmitter to each receiver, which may create a potential for wiring congestion. Another approach is to implement distributed multiplexers, where logic at each block merges signals from other blocks and then possibly fewer wires to the next device. Traditionally bus implementations have either assumed that signals on a bus are a receiver, transmitter, or a transceiver.

[0027] Describing a distributed multiplexed bus topology may be done in a variety of ways. For simple linear topology a list can express the order blocks are connected. However, for more complex topologies where more than one node are merged at any node a tree is a better way to describe the topology. The user may describe the topology of a bus in a parse-tree like syntax, such as:

mux_tree <sub-tree>

<sub-tree> ::= <root> <branch> <branch>

<root> ::= <instance-name>

<branch> ::= { <sub-tree> }

| <instance-name>

| 0 *(a sentinel to indicate no connection)*

[0028] An example tree might be:

mux_tree A { B C { D E } } { F G H }

This line above describes the connectivity between blocks that topologically looks like Figure 2A. When the design is implemented in an IC, the blocks may appear as blocks in Figure 2B. Each node in the tree may communicate with an IP core, optionally a parent node, and optionally a set of child nodes. Each node consists of an IP core and an agent as shown in Figure 2C. The distributed multiplexer (and potentially other functions) may be implemented in the agent. Depending on how the multiplexed bus is connected, the agents may need to be changed, however the core may be unaffected. This may enable reuse of a core without having to change it when the bus changes. Figure 3A shows the receivers and transmitters for each agent in this example. For this

signal of the bus in this example Agents A, C and G need to receive the signal, while Agents B, C, D, and E may transmit it. There may be additional logic in the agent to perform other protocol conversion between these receiver and transmitter interfaces and an actual IP core. Given this transmitter and receiver configuration and the mux_tree specification described above, the agents are wired together as shown in Figure 3B. Note that the output of the mux tree root is connected back to the mux tree root repeater input. The combiner function (Figure 7A) takes as input data and enable from the core or other agent logic and the combiner output of the sub trees. Figure 7B shows the "and-or" implementation of the combiner.

[0029] A tree structure may result in less wiring and/or shorter end-to-end paths than the simple linear wiring. The structure may be specified by the user and/or a program may find the minimal spanning tree. By routing signals using the same topology it may be possible to create predictable wiring delay and/or reduce congestion. Two sets of wires are used between two topologically adjacent nodes: the first may be used to combine the results, and the second may be used to distribute the result back to all nodes.

[0030] Optimizing signal wiring in a distributed multiplexed bus may be done by examining nodes. For example, some nodes may not generate (transmit) certain signals, while other nodes may not use (receive) certain signals. Routing all signals to all nodes may require more wiring and may increase the end-to-end path length for a signal. By removing combining wires from nodes that do not drive the signal and/or the distribution wires to nodes that do not use the result may allow a reduction in the area by requiring fewer drivers and/or less wire, potentially improving chip timing by shortening critical paths, and/or reducing power by using smaller drivers to achieve the same timing.

[0031] Figure 6A illustrates one embodiment of the present invention of a high level algorithm for optimizing bus signal wiring. Note that the sequence need not be performed in a specific order, however the illustrated order is easy to implement. Each signal is analyzed and unnecessary transmitters and receivers are removed. Signals in the bus are optimized individually since each signal may have different transmitters and receivers topologies based on the core function and the bus protocol. The algorithm for removing these unnecessary wires is given below: Given a tree with a root and a list of bus_signals:

```
foreach signal in bus_signals {
        optimize_combiners(root,signal);
        optimize_repeaters(root,signal);
        optimize_root(root,signal);
}
```

[0032] The removal of unnecessary combiners may reduce the amount of wiring used to connect blocks at the top level of the chip and/or may shorten the path of some signals so they may be better optimized for timing, area, power, etc.. The optimization of the combiners, in one embodiment of the present invention, may be done by a bottom up removal of unnecessary combiners for a specific signal. A combiner is unnecessary in an agent if the core attached to the agent and other agent logic does not have a transmitter and none of the children in the sub-tree have a combiner. Figure 6B illustrates in a flow chart, one embodiment of the present invention for optimizing the insertion of combiners. The optimization of combiners may be either additive as shown in Figure 6B, and/or subtractive as shown in the recursive algorithm as is given below:

```
procedure optimize_combiners(sub_tree,signal) {
        has_combiner = node_needs_combiner(sub_tree.node,signal)
        foreach child in sub_tree.children
                if (optimize_combiners(child,signal)) {
                        has_combiner = true
                }
        if (!has_combiner)
                remove_combiner(sub_tree.node,signal)
        return has_combiner
}
```

[0033] Figure 3C illustrates the effect of optimizing combiners. Combiners have been removed from agents F, G, and H. If an combine has one input, then the "or" function can be replaced by a buffer if the input is from a sub-tree or combiner's "and" of data and enable if the input is from the transmitter.

[0034] Unused repeaters may be optimized by removal of unnecessary repeaters for a specific signal. This may have timing, area, power, etc. benefits. A repeater is unnecessary for an agent if the core attached to the agent and other logic in the agent does not have a receiver and none of the children in the sub-tree have a repeater. Figure 6C illustrates a flowchart, for one embodiment of the present invention, for

adding repeaters to the mux tree. The repeater optimization process may be additive as shown in Figure 6C, and/or it may be subtractive as show in the algorithm below:

```
procedure optimize_repeaters(sub_tree,signal) {
    has_repeater = node_needs_repeater(sub_tree.node,signal)
    foreach child in sub_tree.children
        if (optimize_repeaters(child,signal)) {
            has_repeater = true
        }
    if (!has_repeater)
        remove_repeater(sub_tree.node,signal)
    return has_repeater
}
```

[0035] Figure 3D illustrates the effect of optimizing repeaters. Repeaters have been removed from agents D, E and H.

[0036] If all the transmitters are in one sub-tree, then wires from the root of the entire tree to the root of that sub-tree used for returning the result may be removed as this node can drive the result directly to the sub-tree and to the root. Figure 6D illustrates the flowchart for optimizing the root for a signal, while example pseudo-code is given below:

```
procedure optimize_root(sub_tree,signal) {
    if node_needs_transmitter(sub_tree,signal)
        connect_combiner_to_repeater(sub_tree.root);
        return;
    new_root = NULL
    foreach child in sub_tree.children {
        if (node_needs_transmitter(child,signal)) {
            if (new_root != NULL) {
                return;
            }
            new_root = child;
        }
    }
    if (new_root != NULL)
        optimize_root(child,signal)
```

}

[0037] Figure 3E illustrates the effect of optimizing the root. In this example, the repeater wire connecting A and B has been removed and agent B acts as the root for this signal.

[0038] Optimizing the timing of the distributed multiplexed bus may result in the reduction of power, area, etc. Logic synthesis is a program that translates equations into optimized logic gates. In addition to the logic equations, synthesis may also accepts constraints. Timing constraints can describe when inputs are available and when outputs are required, logic synthesis tries to optimize the logic gates to best meet these constraints. In prior approaches timing constraints may not have considered the position in the bus topology when generating constraints. This may lead to over-constraints and consequently a sub-optimal design in terms of area and/or power. One embodiment of the present invention considers the location of each agent in the bus topology when generating constraints. Constraints are generated after the signal wiring has been optimized. Based on prior characterization an estimate is made for each component of timing which makes up the overall bus delay. The components include those that are scalable and those that are fixed. The scalable components may include the register to bus output (Figure 5-A), bus input to bus output (Figure 5-C,F), and bus input to register delays (Figure 5-E,H). The fixed components may be the delays due to wiring between an output and an input ports (Figure 5-B,D,G) based on the location of the agents. Using these components a delay calculation can be done on all of the paths which compose the bus. Given a multiplexed bus topology shown in Figure 4A, a multiplexed bus in Figure 4-B will be used. Applying the above optimizations will result in the path from D to A through B1 being optimized for better timing than the logically equivalent path from C to A through B2. Additionally the size of the driver in node C will be reduced because that path is less critical than the path starting from E.

[0039] Figure 8 illustrates one embodiment of the present invention for optimizing the timing constraints. The delay calculation is done by searching all possible paths from an output port and adding up the timing components which make up the path. A list of each unique path is kept track of along with its overall delay. Each of the components of the path is stored in the list for the scaling process.

[0040] The next procedure is to scale each path to meet the timing required by the bus. Paths which exceed the timing are scaled down to meet it by calculating a scale factor which reduces each scalable timing component. Paths which have timing less

than that required are scaled up to meet it by calculating a scale factor which increases each scalable timing component. Scaling proceeds by starting with the longest delay paths, applying the path specific scaling factor to each component and marking each component as scaled. This process continues for each path generated above. The ordering and marking is important so as not to increase the delay on a timing component required by a longer path. By allowing more time (scaling up the timing components) logic synthesis my be able to select slower cells, which are smaller and use less power. The end result may be less area and/or less power requirements for the overall design.

[0041] Figure 5 illustrates the effect of this process. In this configuration the constraints for path A,B,C,D,E are looser because the timing on the path through A,B,C,D,F,G,H is a longer path. Consequently, logic synthesis will optimize the timing from G, H more than that from D,E.

[0042] Multiple simultaneous drivers of the same signal may be legal for certain signals (like an error or interrupt signal), but illegal for other signals (like address). Simulator are able to detect multiple simultaneous tri-state drivers when they are driving conflicting values (one driver driving a 1 and another driving 0) and generate an X to aid in detecting design errors. Detecting multiple drivers in a distributed multiplexed bus is difficult because the combining function ("and-or" for example) may not enable the simulator to catch this design error, and the distributed nature of its implementation may make it hard to add a single checker. Another approach is to distribute the checking in each combiner function of the distributed multiplexer.

[0043] An "or" implementation of an N-bit combiner verilog logic equation is given below:

Output = (core_input & {N{core_enable}) | left_input | right_input

[0044] A checker to detect conflicts can look for cases where the core is enabled (core_input is not zero) and either the left or right input is not zero. Additionally, if both the left input and the right input is not zero then, there are multiple drivers. Table 1 (Figure 7C) gives the truth table for detecting bus conflict. The verilog logic equations for this are:

error = (Core_enable && | (left_input | right_input)) || (|left_input || |right_input)

[0045] This may not catch the case where multiple cores are driving zero, however, the probability of this for multi-bit signals is relatively low, so this check is nearly as good as the more complicated check of all of the core enable signals. This can then be used

to stop the simulation and report a design error as shown below:

```
if (error) begin
        $display("multiple drivers");
        $finish;
end
```

[0046] Thus, what has been disclosed is a method and apparatus for optimizing distributed multiplexed bus interconnects.

[0047] Figure 9 illustrates a network environment 900 in which the techniques described may be applied. The network environment 900 has a network 902 that connects S servers 904-1 through 904-S, and C clients 908-1 through 108-C. As shown, several systems in the form of S servers 904-1 through 904-S and C clients 908-1 through 908-C are connected to each other via a network 1902, which may be, for example, an on-chip communication network. Note that alternatively the network 902 might be or include one or more of: inter-chip communications, an optical network, the Internet, a Local Area Network (LAN), Wide Area Network (WAN), satellite link, fiber network, cable network, or a combination of these and/or others. The servers may represent, for example: a master device on a chip; a memory; an intellectual property core, such as a microprocessor, communications interface, etc.; a disk storage system; and/or computing resources. Likewise, the clients may have computing, storage, and viewing capabilities. The method and apparatus described herein may be applied to essentially any type of communicating means or device whether local or remote, such as a LAN, a WAN, a system bus, on-chip bus, etc. It is to be further appreciated that the use of the term client and server is for clarity in specifying who initiates a communication (the client) and who responds (the server). No hierarchy is implied unless explicitly stated. Both functions may be in a single communicating device, in which case the client-server and server-client relationship may be viewed as peer-to-peer. Thus, if two devices such as 908-1 and 904-S can both initiate and respond to communications, their communication may be viewed as peer-to-peer. Likewise, communications between 904-1 and 904-S, and 908-1 and 908-C may be viewed as peer to peer if each such communicating device is capable of initiation and response to communication.

[0048] Figure 10 illustrates a computer system 1000 in block diagram form, which may be representative of any of the clients and/or servers shown in Figure 9. The block diagram is a high level conceptual representation and may be implemented in a variety

of ways and by various architectures. Bus system 1002 interconnects a Central Processing Unit (CPU) 1004, Read Only Memory (ROM) 1006, Random Access Memory (RAM) 1008, storage 1010, display 1020, audio, 1022, keyboard 1024, pointer 1026, miscellaneous input/output (I/O) devices 1028, and communications 1030. The bus system 1002 may be for example, one or more of such buses as an on-chip bus, a system bus, Peripheral Component Interconnect (PCI), Advanced Graphics Port (AGP), Small Computer System Interface (SCSI), Institute of Electrical and Electronics Engineers (IEEE) standard number 1394 (FireWire), Universal Serial Bus (USB), etc. The CPU 1004 may be a single, multiple, or even a distributed computing resource. Storage 1010, may be Compact Disc (CD), Digital Versatile Disk (DVD), hard disks (HD), optical disks, tape, flash, memory sticks, video recorders, etc. Display 1020 might be, for example, a Cathode Ray Tube (CRT), Liquid Crystal Display (LCD), a projection system, Television (TV), etc. Note that depending upon the actual implementation of the system, the system may include some, all, more, or a rearrangement of components in the block diagram. For example, an on-chip communications system on an integrated circuit may lack a display 1020, keyboard 1024, and a pointer 1026. Another example may be a thin client might consist of a wireless hand held device that lacks, for example, a traditional keyboard. Thus, many variations on the system of Figure 10 are possible.

[0049] For purposes of discussing and understanding the invention, it is to be understood that various terms are used by those knowledgeable in the art to describe techniques and approaches. Furthermore, in the description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be evident, however, to one of ordinary skill in the art that the present invention may be practiced without these specific details. In some instances, well-known structures and devices are shown in block diagram form, rather than in detail, in order to avoid obscuring the present invention. These embodiments are described in sufficient detail to enable those of ordinary skill in the art to practice the invention, and it is to be understood that other embodiments may be utilized and that logical, mechanical, electrical, and other changes may be made without departing from the scope of the present invention.

[0050] Some portions of the description may be presented in terms of algorithms and symbolic representations of operations on, for example, data bits within a computer memory. These algorithmic descriptions and representations are the means used by

those of ordinary skill in the data processing arts to most effectively convey the substance of their work to others of ordinary skill in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of acts leading to a desired result. The acts are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical, magnetic, or optical signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

[0051] It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "communicating" or "displaying" or the like, can refer to the action and processes of a computer system, or an electronic device, that manipulates and transforms data represented as physical (electronic) quantities within the electronic device or computer system's registers and memories into other data similarly represented as physical quantities within the electronic device and/or computer system memories or registers or other such information storage, transmission, or display devices.

[0052] The present invention can be implemented by an apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general-purpose computer, selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but not limited to, any type of disk including floppy disks, hard disks, optical disks, compact disk- read only memories (CD-ROMs), digital versatile disk (DVD), and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), electrically programmable read-only memories (EPROM)s, electrically erasable programmable read-only memories (EEPROMs), FLASH memories, magnetic or optical cards, etc., or any type of media suitable for storing electronic instructions either local to the computer or remote to the computer.

[0053] The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general purpose systems may be

used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the required method. For example, any of the methods according to the present invention can be implemented in hard-wired circuitry, by programming a general-purpose processor, or by any combination of hardware and software. One of ordinary skill in the art will immediately appreciate that the invention can be practiced with computer system configurations other than those described, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, digital signal processing (DSP) devices, set top boxes, network PCs, minicomputers, mainframe computers, and the like. The invention can also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. This communications network is not limited by size, and may range from, for example, on-chip communications to WANs such as the Internet.

[0054] The methods of the invention may be implemented using computer software. If written in a programming language conforming to a recognized standard, sequences of instructions designed to implement the methods can be compiled for execution on a variety of hardware platforms and for interface to a variety of operating systems. It will be appreciated that a variety of programming languages may be used to implement the teachings of the invention as described herein. Furthermore, it is common in the art to speak of software, in one form or another (e.g., program, procedure, application, driver,...), as taking an action or causing a result. Such expressions are merely a shorthand way of saying that execution of the software by a computer causes the processor of the computer to perform an action or produce a result.

[0055] It is to be understood that various terms and techniques are used by those knowledgeable in the art to describe communications, protocols, applications, implementations, mechanisms, etc. One such technique is the description of an implementation of a technique in terms of an algorithm or mathematical expression. That is, while the technique may be, for example, implemented as executing code on a computer, the expression of that technique may be more aptly and succinctly conveyed and communicated as a formula, algorithm, or mathematical expression. Thus, one of ordinary skill in the art would recognize a block denoting A+B=C as an additive function whose implementation in hardware and/or software would take two inputs (A and B) and produce a summation output (C). Thus, the use of formula, algorithm, or mathematical expression as descriptions is to be understood as having a physical embodiment in at

least hardware and/or software (such as a computer system in which the techniques of the present invention may be practiced as well as implemented as an embodiment).

[0056] A machine-readable medium is understood to include any mechanism for storing or transmitting information in a form readable by a machine (e.g., a computer). For example, a machine-readable medium includes read only memory (ROM); random access memory (RAM); magnetic disk storage media; optical storage media; flash memory devices; electrical, optical, acoustical, or other form of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.); etc.

[0057] Thus, a method and apparatus for optimizing distributed multiplexed bus interconnects have been described.

CLAIMS

What is claimed is:

1. A method comprising:

optimizing a distributed multiplexed bus interconnect wherein said optimizing comprises optimizing a parameter selected from the group consisting of combiner optimization, repeater optimization, and root optimization.

2. The method of claim 1 wherein optimizing further comprises examining resources associated with a plurality of agents, the resources selected from the group consisting of receiver, transmitter, distribution wires, bus wiring, drivers, buffers, and logic gates.

3. The method of claim 2 wherein the optimization results in a tree structure.

4. The method of claim 2 wherein optimizing further comprises operations selected from the group consisting of removal of combiner, removal of repeater, removal of routing, and scaling of buffers.

5. The method of claim 4 wherein the removal is performed bottom up.

6. The method of claim 1 wherein the optimizing is done at a point in time before fabrication of a device.

7. A processing system comprising a processor, which when executing a set of instructions performs the method of claim 1.

8. A machine-readable medium having stored thereon instructions, which when executed performs the method of claim 1.

9. A method comprising:

marking all bus signals as unoptimized;

(a) determining if all bus signals have been optimized; and

if so, then stopping;

else,

(b) picking one of the unoptimized bus signals;

(c) optimizing signal combiners;

(d) optimizing signal repeaters;

(e) optimizing signal root;

(f) marking the signal as optimized; and

(g) looping to (a).

10. A method comprising:

(a) determining if all transmitters are connected; and

if so, then returning;

else,

(b) picking an unconnected transmitter;

(c) adding a combiner to an agent connected to the transmitter's core;

(d) determining if the agent is a root; and

if so, then looping to (a);

else,

(e) determining if a parent has a combiner; and

if not, then adding a combing to the parent agent;

(f) attaching the agent's combiner output to a input of the parent agents combiner;

(g) setting the agent to equal the parent agent; and

(h) looping to (d).

11. A method comprising:

(a) determining if all receivers are connected; and

if so, then returning;

else,

(b) picking an unconnected receiver;

(c) adding a repeater to an agent connected to the receiver's core;

(d) determining if the agent is a root; and

if so, then looping to (a);

else,

(e) determining if a parent has a repeater; and

if not, then adding a repeater to the parent agent;

(f) attaching the parent agent's repeater output to a input of the parent agent's repeater;
(g) setting the agent to equal the parent agent; and
(h) looping to (d).

12. A method comprising:
setting an agent to equal a root;
(a) connecting the agent combiner output to the agent repeater input'
(b) determining if the agent has a transmitter input; and
if so, then returning;
else,
(c) determining if the agent has a single combiner input; and
if not, then returning;
(d) removing a repeater wire from the agent to a child;
(e) setting the agent to equal the child with the combiner; and
(f) looping to (a).

13. An apparatus comprising:
means for optimizing a distributed multiplexed bus interconnect wherein said means for optimizing comprises means for optimizing selected from the group consisting of means for combiner optimization, means for repeater optimization, and means for root optimization.

14. The apparatus of claim 13 wherein means for optimizing is a means for optimizing before said apparatus is fabricated.

15. The apparatus of claim 13 wherein means for optimizing further comprises means for optimizing at a point in time selected from the group consisting of at time of fabrication, at a power up, at a reset, at an initialization prior to normal operation, and dynamically during normal operation.

16. A machine-readable medium having stored thereon information representing the apparatus of claim 13.

17. A system comprising:
a plurality of agents;
a plurality of interfaces; and
a multiplexed bus connecting the plurality of agents.

18. The system of claim 17 wherein the multiplexed bus connecting the plurality of agents is optimized by a parameter selected from the group consisting of combiner optimization, repeater optimization, and root optimization.

19. The system of claim 18 wherein the optimization is done at time of system design.

20. The system of claim 17 further comprising transferring a payment and/or a credit.

21. An apparatus comprising:
a distributed multiplexed bus; and
a plurality of agents interconnected in an optimized manner via the distributed multiplexed bus based upon locations of said plurality of agents.

wherein said means for optimizing comprises means for optimizing selected from the group consisting of means for combiner optimization, means for repeater optimization, and means for root optimization.

22. The apparatus of claim 21 wherein the optimized manner further comprises optimized logic selected from the group consisting of a combiner, a repeater, and a buffer.

23. The apparatus of claim 21 wherein optimized manner occurs at a point in time selected from the group consisting of before fabrication of the apparatus, at time of fabrication of the apparatus, at a power up of the apparatus, at a reset of the apparatus, at an initialization of the apparatus prior to normal operation, and dynamically during normal operation of the apparatus.

24. A machine-readable medium having stored thereon information representing the apparatus of claim 21.

25. A method comprising:

searching all possible paths from an output port;

adding up timing components along each path;

maintaining a list of each unique path and an associated delay; and

scaling each path to meet a required timing constraint for said path.

26. The method of claim 25 wherein the scaling proceeds by starting with the longest delay path.

27. The method of claim 25 wherein the scaling further comprises scaling up or down.

28. The method of claim 27 wherein scaling down further comprises replacing a cell with a slower cell.

29. A method comprising:

initializing a path list;

collecting bus port outputs;

tracing for each output port all paths and adding them to the path list;

sorting paths in the path list from longest to shortest; and

determining for each path a scale factor and scaling path components with said scale factor.

30. The method of claim 29 wherein tracing for each output port further comprises:

creating a partial path consisting of register to output component;

(a) finding all input ports connected to said output port;

(b) creating a first path consisting of a first partial path up to said first partial path current point, wire delay, and input port to register component;

(c) adding said created first path (in (b)) to the path list;

repeating (b)-(c) for each input port;

finding all input port to output port connections;

(d) creating a second path consisting of a second partial path up to said second partial path current point, and input port to output port component;

(e) adding said created second path (in (d)) to the path list;

repeating (d)-(e) for each input port; and

looping to (a) for each output.

31. The method of claim 29 wherein determining for each path a scale factor further comprises:

initializing a path delay to 0 and initializing a target delay to a clock period for each path in the path list;

(a) determining if a component is not already scaled and is not fixed; and

if so, setting path delay to equal path delay plus component delay

(b) determining if the component is fixed; and

if so, setting target delay to equal the target delay minus the component delay;

(c) determining if at the end of the path; and

if so, setting scale factor to equal target delay divided by the path delay;

else looping to (a).

32. The method of claim 29 wherein scaling path components further comprises:

determining each path in the path list;

(a) determining for each said path each component in each said path:

(b) determining if a component is not already scaled and is not fixed; and

if so, scaling component by a given scale factor and marking the component as being scaled;

(b) determining if at the end of said path; and

if not, then looping to (b); and

if so, then determining if at the end of the path list; and

if not, then looping to (a);

else done with scaling.

33. A processing system comprising a processor, which when executing a set of instructions performs the method of claim 29.

34. A machine-readable medium having stored thereon instructions, which when executed performs the method of claim 29.

35. A method comprising:

accepting a timing constraint for a multiplexed bus connection;

determining locations for a plurality of agents along the multiplexed bus connection; and

generating scaling factors for the plurality of agents based upon said determined locations to meet the timing constraint.

36. The method of claim 35 wherein the scaling factors may be applied to scalable components selected from the group consisting of register to bus output, bus input to bus output, bus input to register delays, and buffer delays.

37. A processing system comprising a processor, which when executing a set of instructions performs the method of claim 35.

38. A machine-readable medium having stored thereon instructions, which when executed performs the method of claim 35.

39. An apparatus comprising:

means for initializing a path list;

means for collecting bus port outputs;

means for tracing for each output port all paths and means for adding them to the path list;

means for sorting paths in the path list from longest to shortest; and

means for determining for each path a scale factor and means for scaling path components with said scale factor.

40. A machine-readable medium having stored thereon information representing the apparatus of claim 39.

41. A method comprising;

selecting a multiplexed bus; and

detecting multiple simultaneously active drivers on the multiplexed bus.

42. The method of claim 41 wherein the detecting is distributed to one or more

combiner in a distributed multiplexer.

43. The method of claim 42 wherein the combiner is a N-bit "or" implemented combiner.

44. The method of claim 43 wherein the detecting further comprises examining an enable, a left input, and a right input.

45. A processing system comprising a processor, which when executing a set of instructions performs the method of claim 41.

46. A machine-readable medium having stored thereon instructions, which when executed performs the method of claim 41.

47. An apparatus comprising;

means for selecting a multiplexed bus; and

means for detecting multiple simultaneously active drivers on the multiplexed bus.

48. The apparatus of claim 47 wherein means for detecting is distributed to one or more combiner in a distributed multiplexer.

49. The apparatus of claim 48 wherein the combiner is a N-bit "or" implemented combiner.

50. The apparatus of claim 49 wherein the means for detecting further comprises means for examining an enable, a left input, and a right input.

51. A machine-readable medium having stored thereon information representing the apparatus of claim 47.

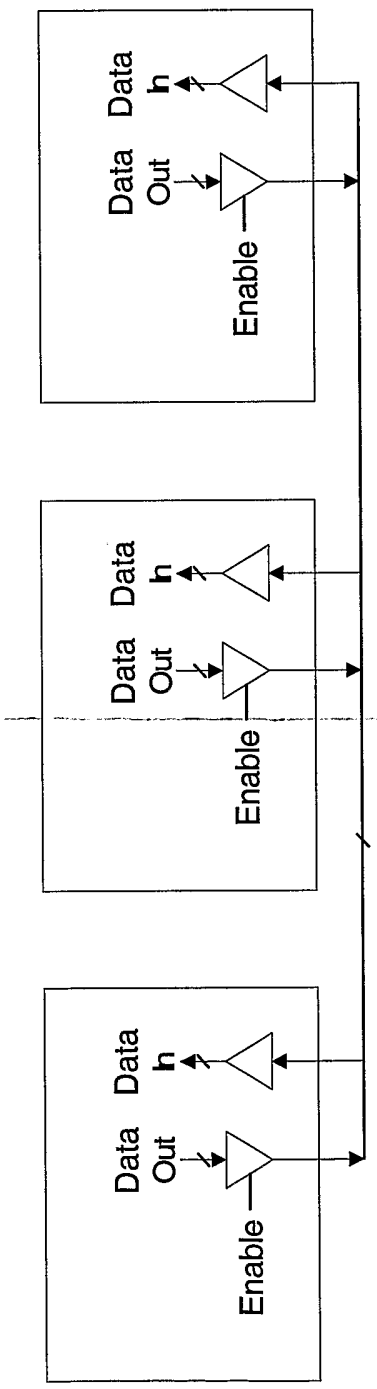


Figure 1A: Tri-State Bus (Prior Art)

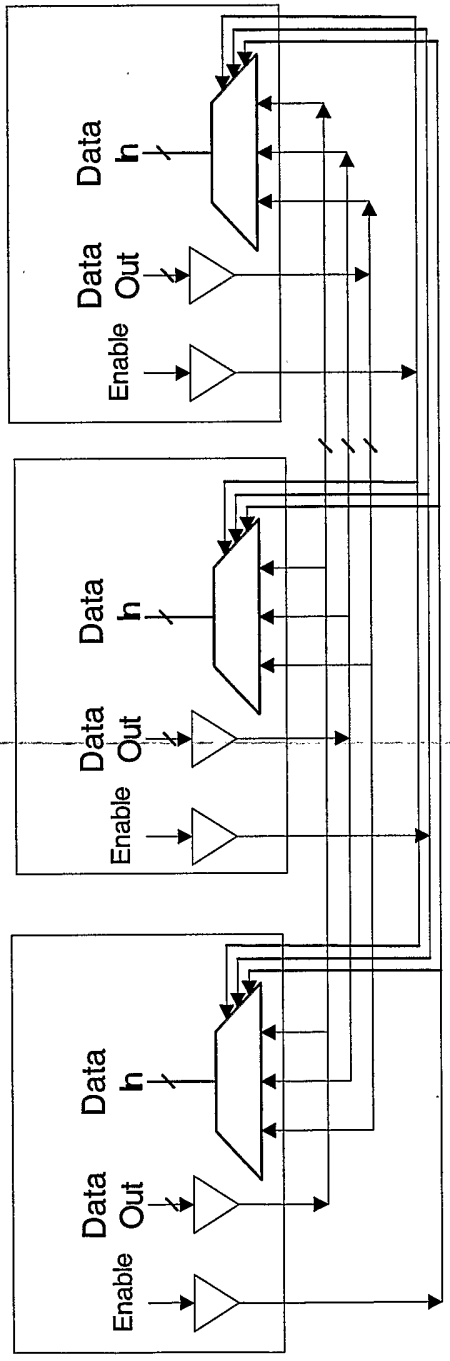


Figure 1B: Multiplexed Bus (Prior Art)

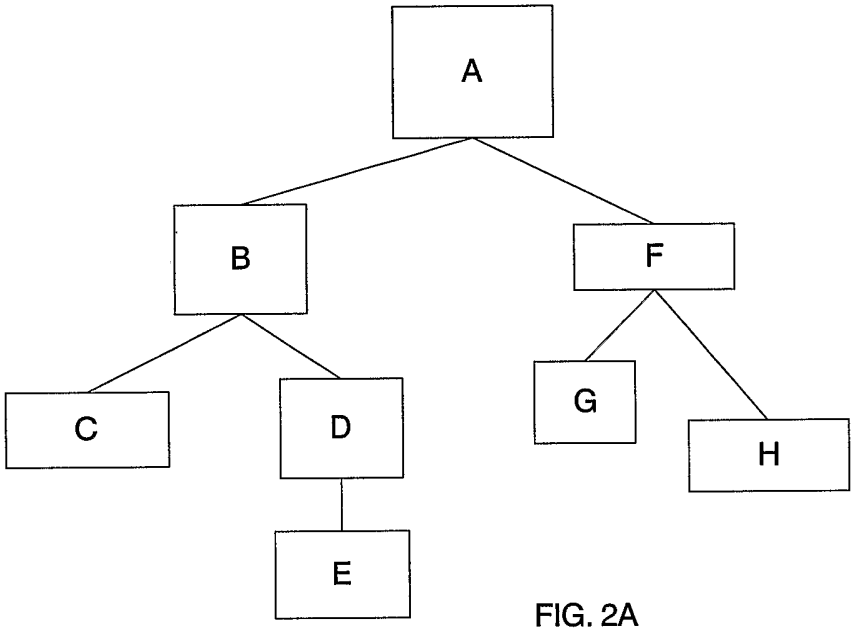


FIG. 2A

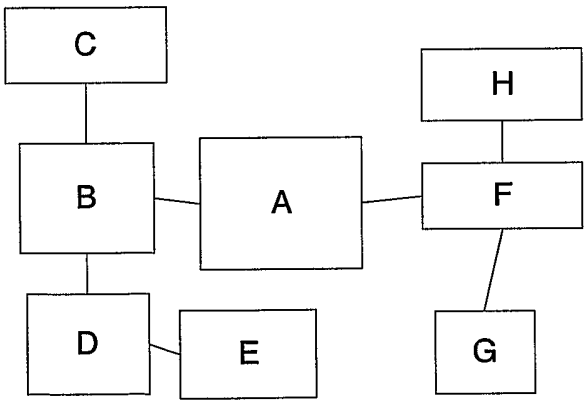


FIG. 2B

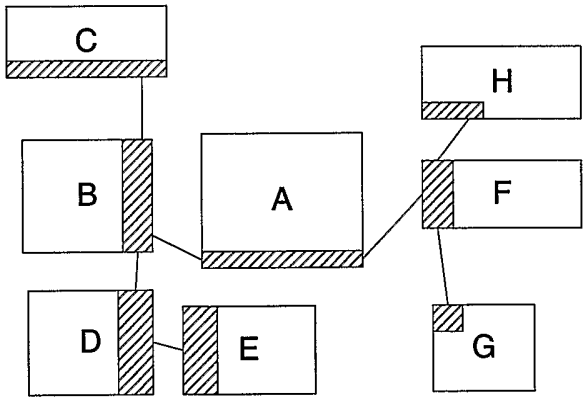
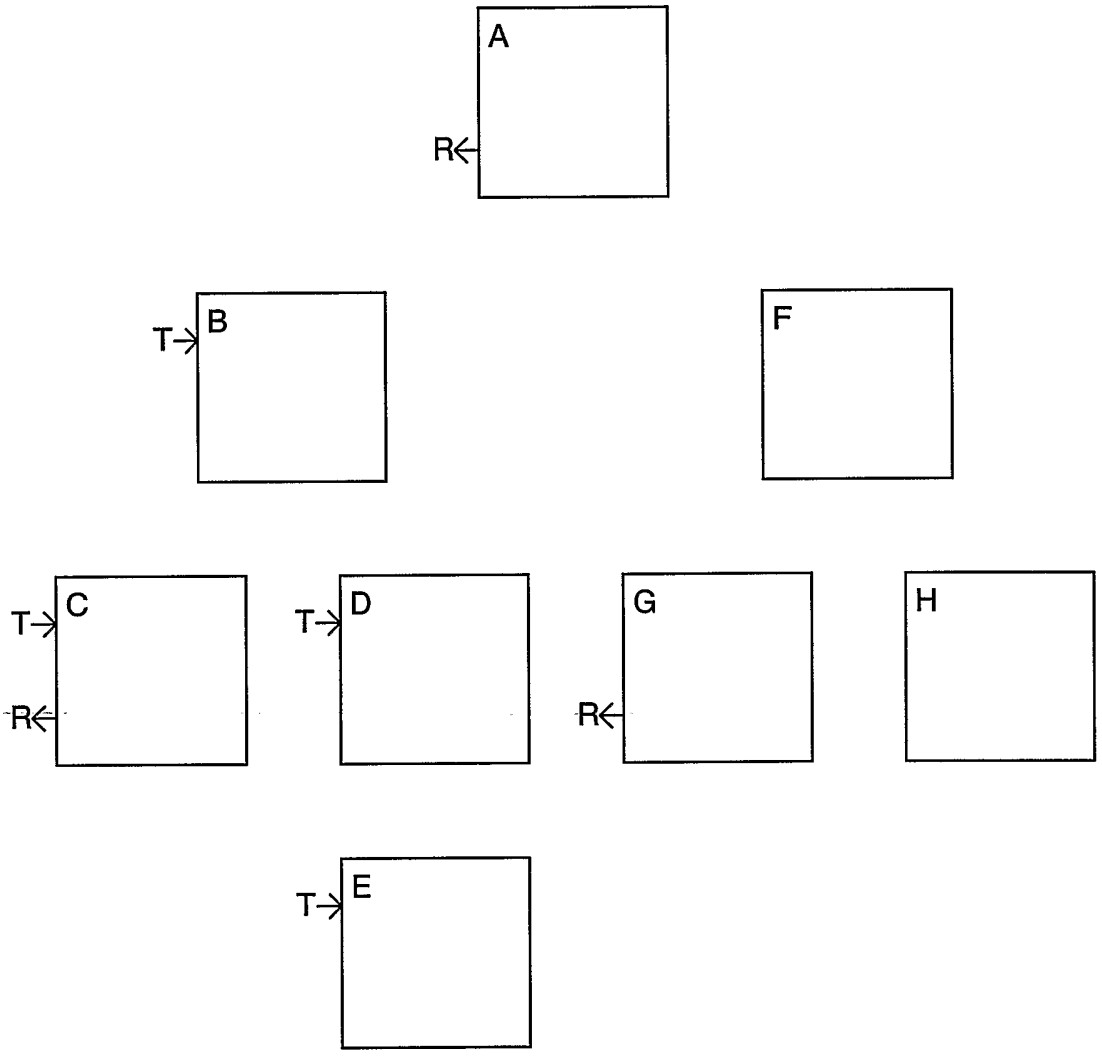




FIG. 2C



Given:

Agents A, C and G have receivers
Agents B, C, D and E have transmitters

Figure 3A: Receivers and Transmitters for specific bus signal

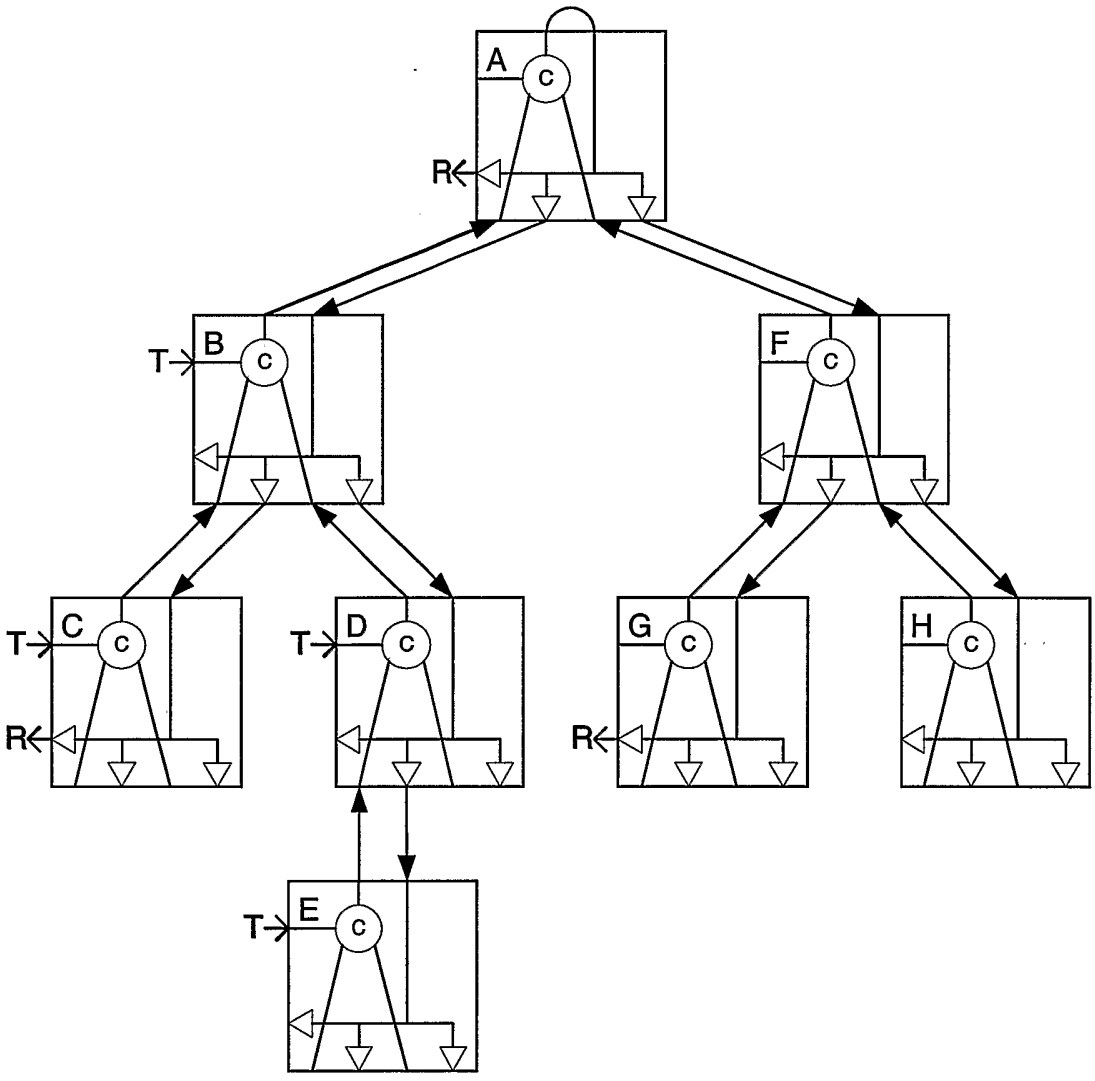


Figure 3B: Unoptimized connections for mux_tree A { B C { D E 0 } } { F G H }

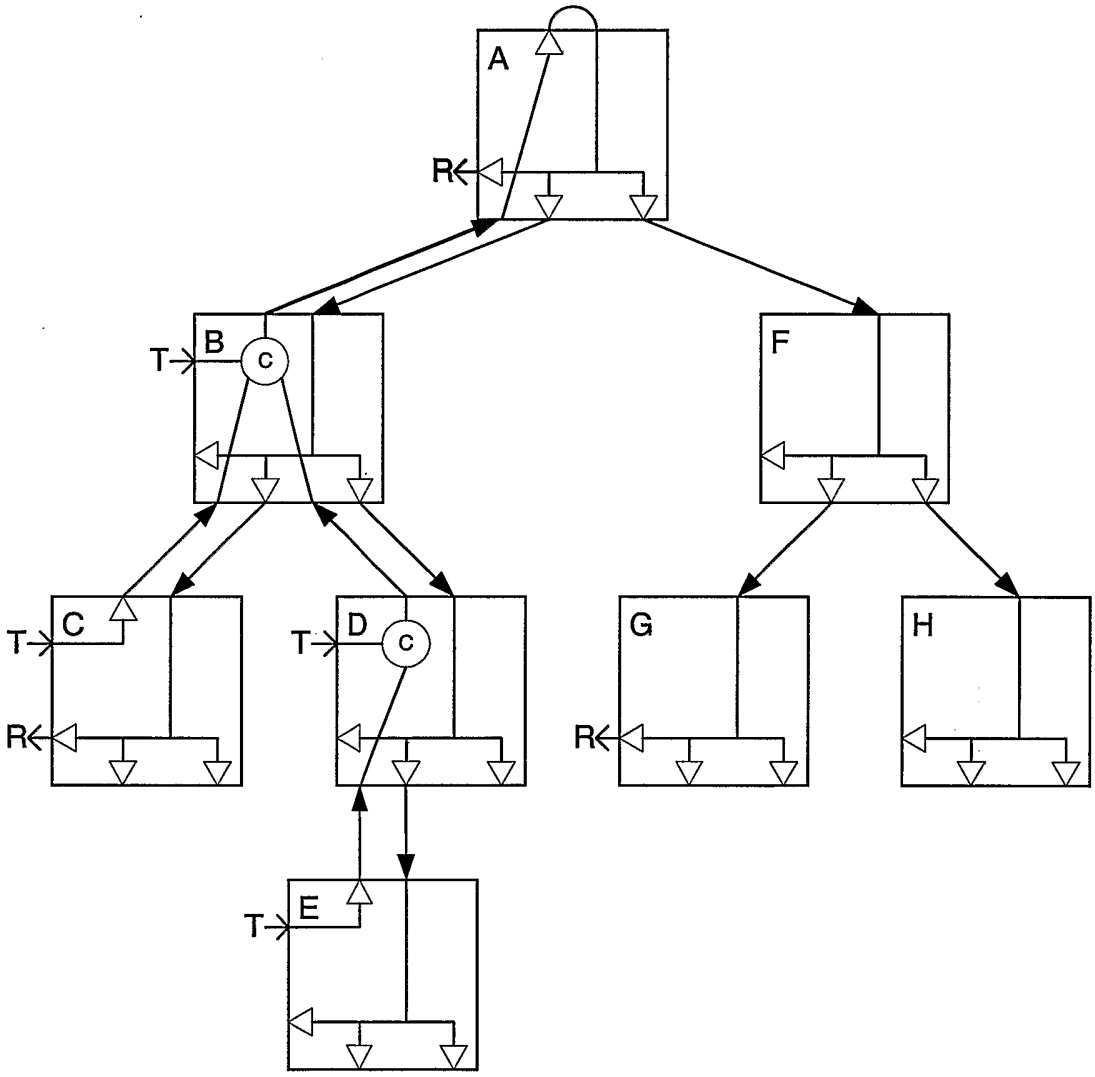


Figure 3C: After combiner optimization

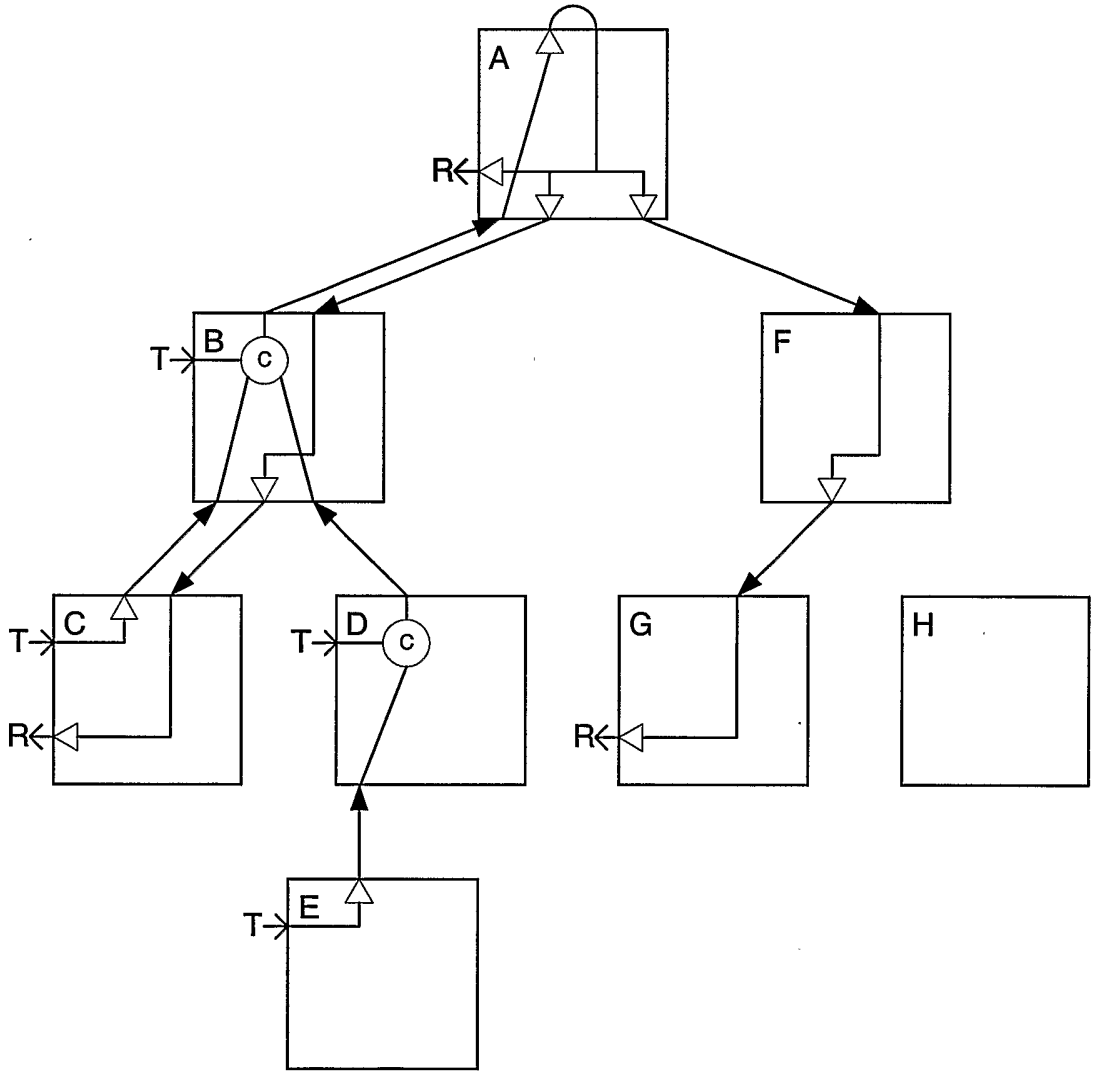


Figure 3D: After repeater optimization

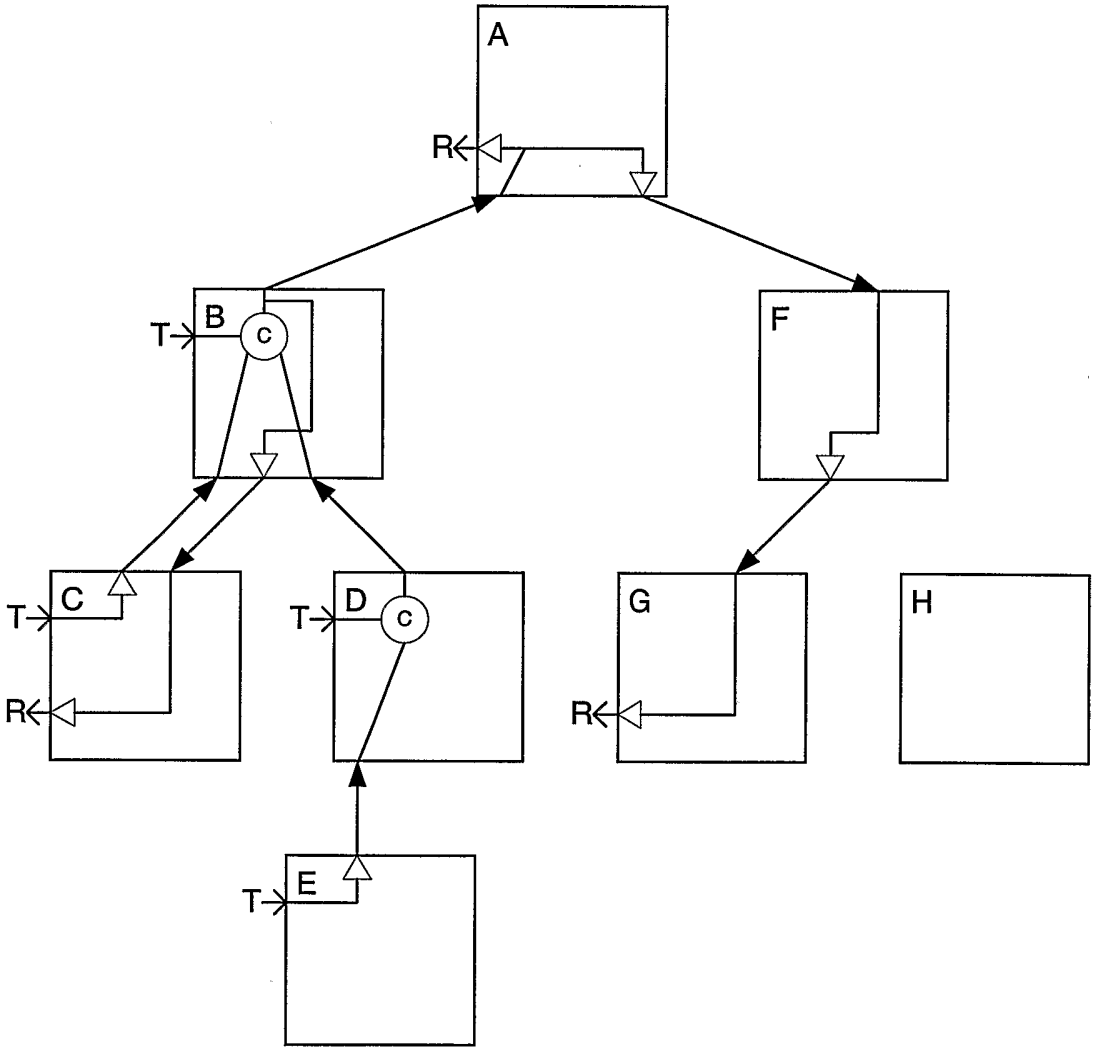


Figure 3E: After root optimization

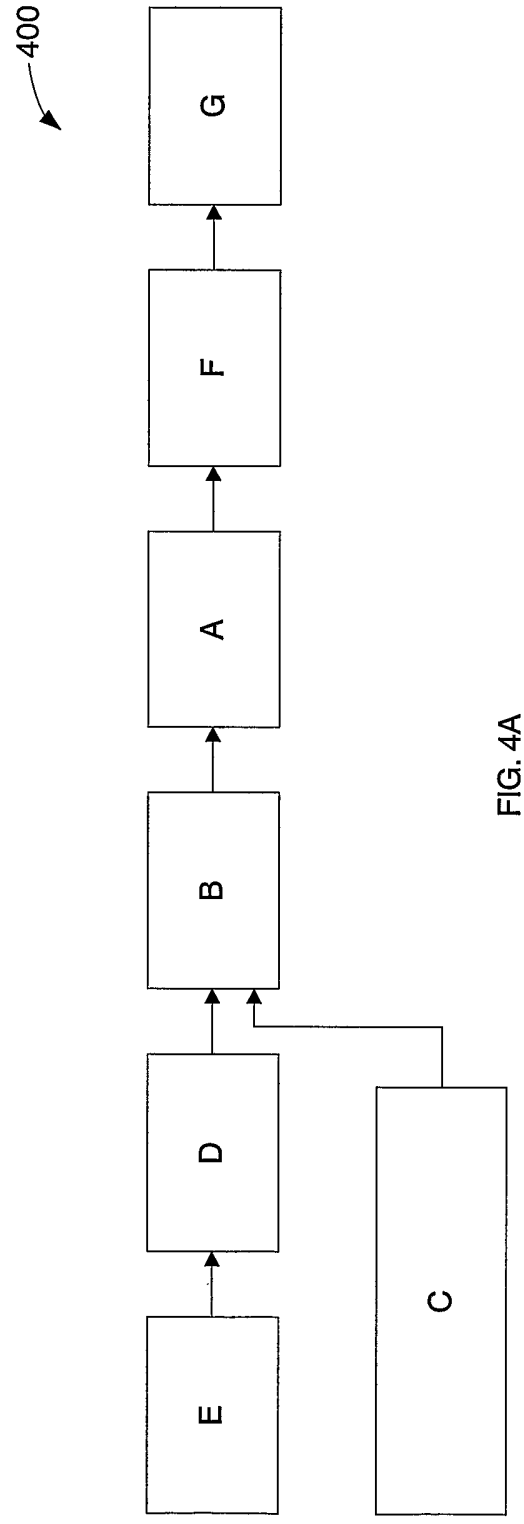

400
G
F
A
B
D
C
E

FIG. 4A

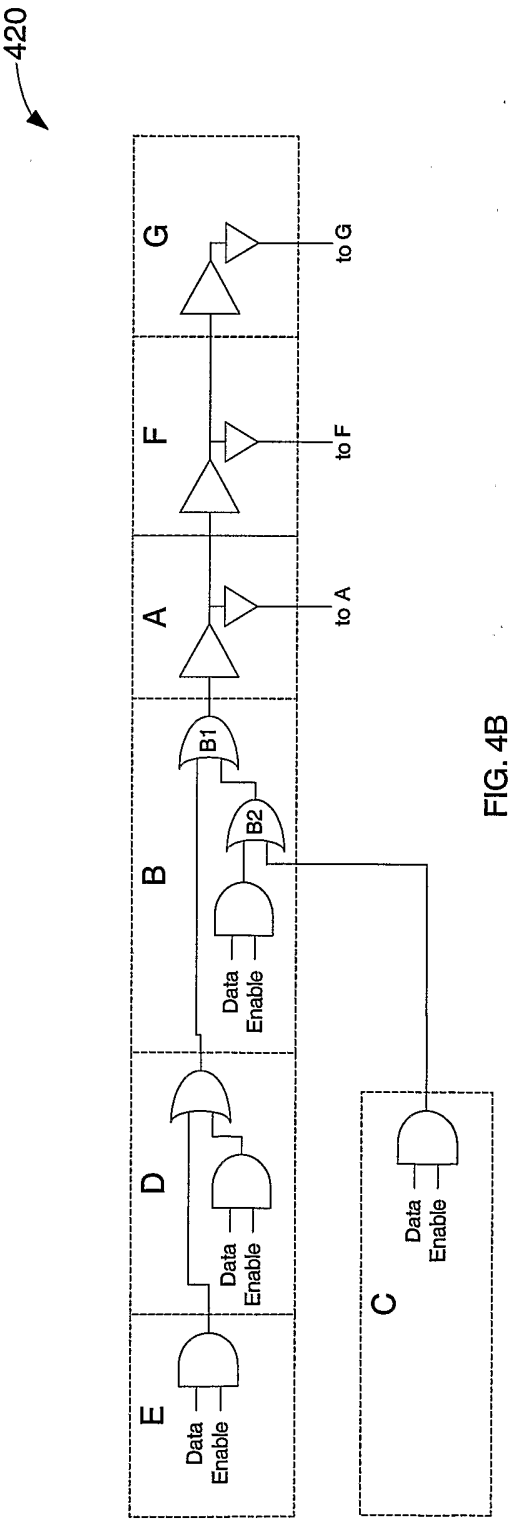
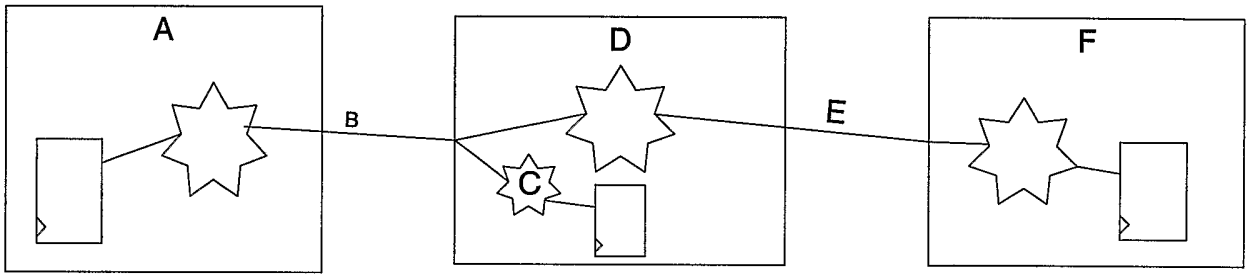

420
G
to G
F
to F
A
to A
B
B1
B2
Data
Enable
D
Data
Enable
E
Data
Enable
C
Data
Enable

FIG. 4B



Path A B D E F is a critical path and required to meet clock period.
Path A B C is much shorter but segment A must still meet timing requirement. Can only scale segment C.

FIG. 5

Optimize Bus Signal Wiring

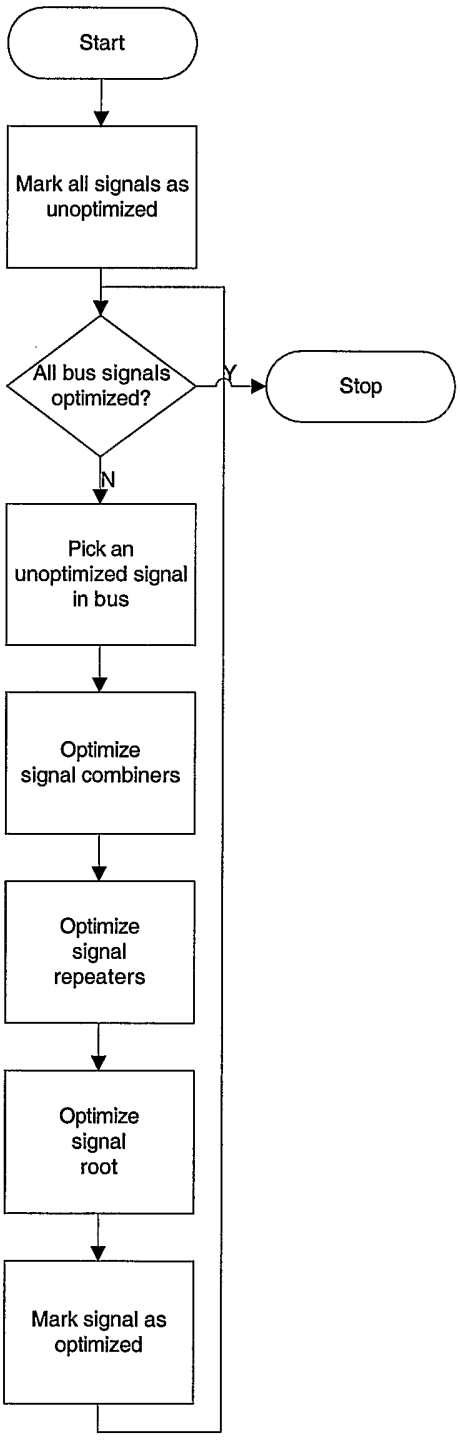


Figure 6A: Flowchart for Optimizing Bus Signal Wiring

Optimize signal combiners

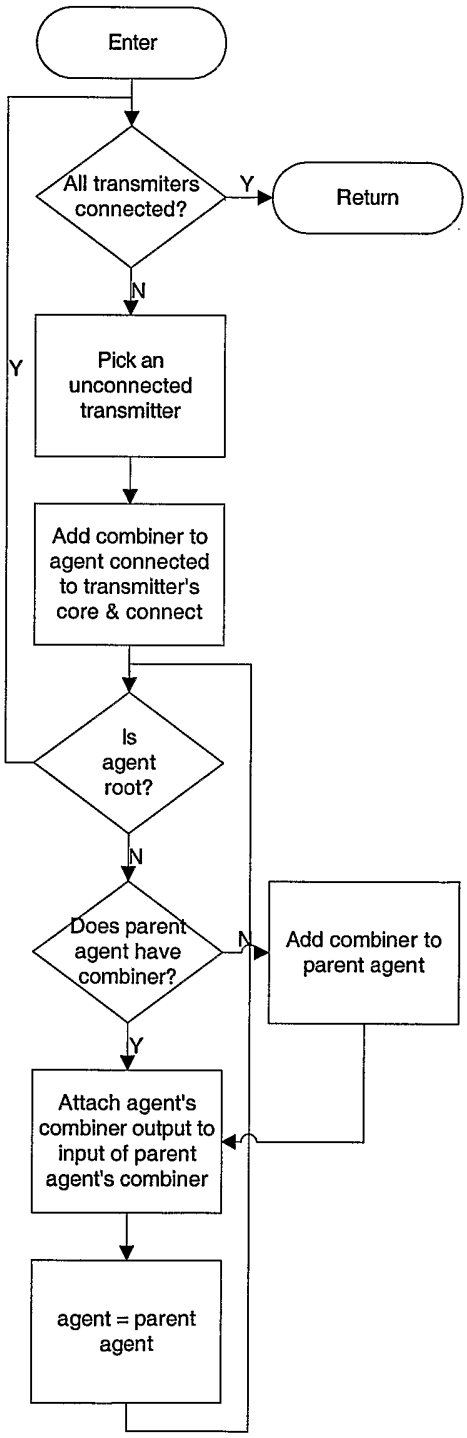


Figure 6B: Flowchart for Optimizing Combiners

Optimize signal repeaters

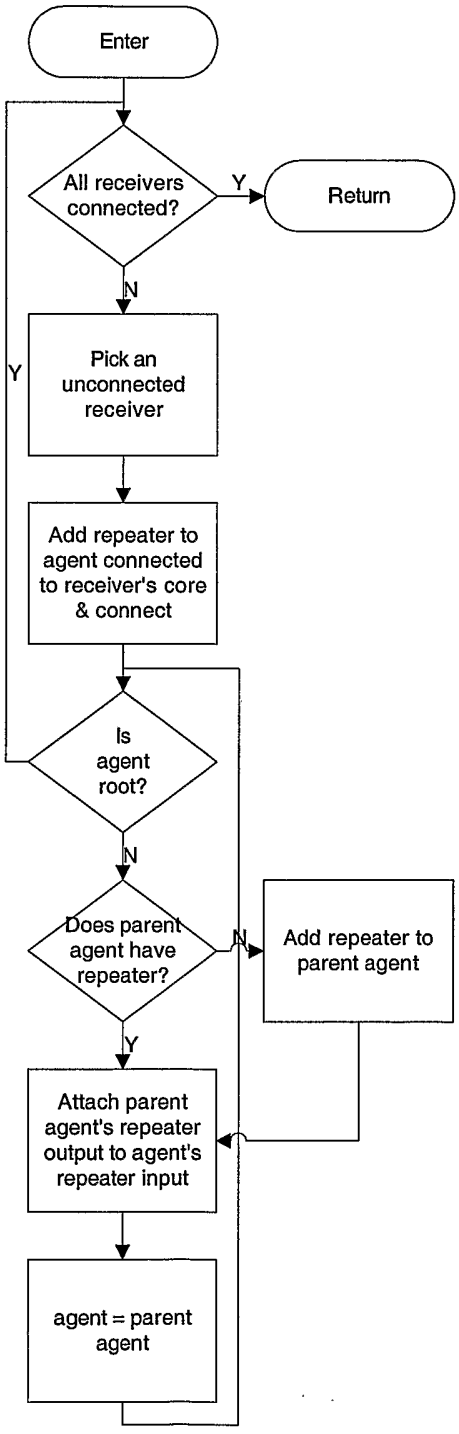


Figure 6C: Flowchart for Optimizing Repeaters

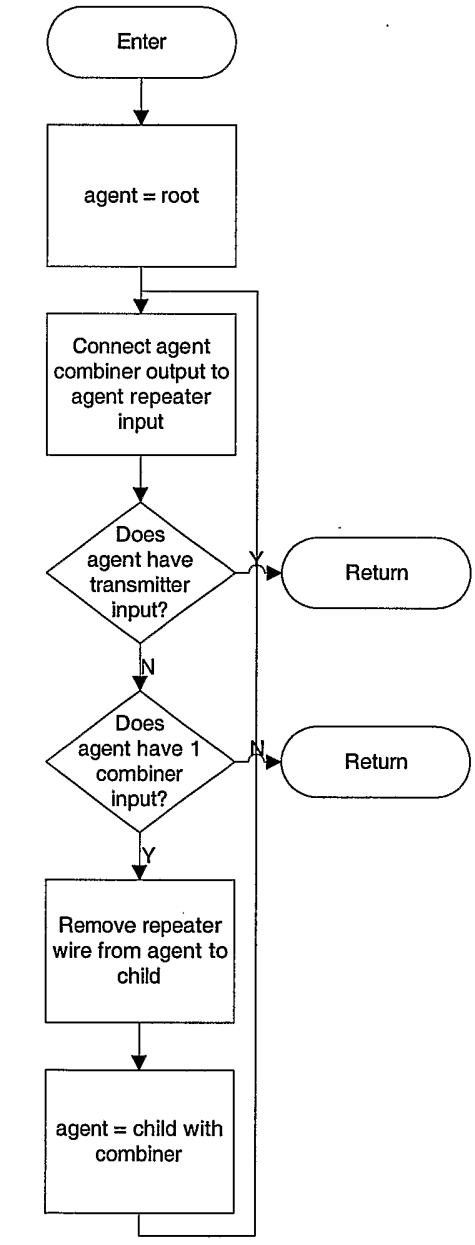


Figure 6D: Flowchart for Optimizing Root

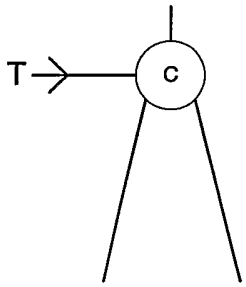


Figure 7A: The combiner block

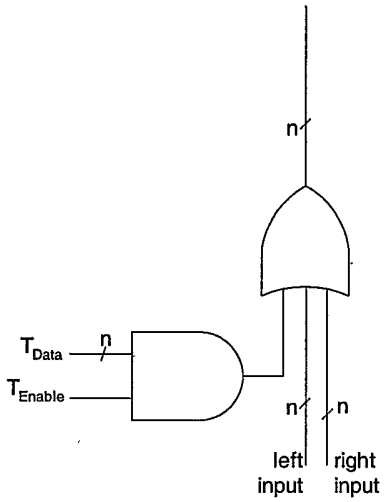


Figure 7B: Details of the combiner block

Tenable	left input	right input	bus state
0	0	0	No conflict
1	!=0	X	Conflict
1	X	!=0	Conflict
X	!=0	!=0	Conflict

Table 1: Conflict Detection

Figure 7C

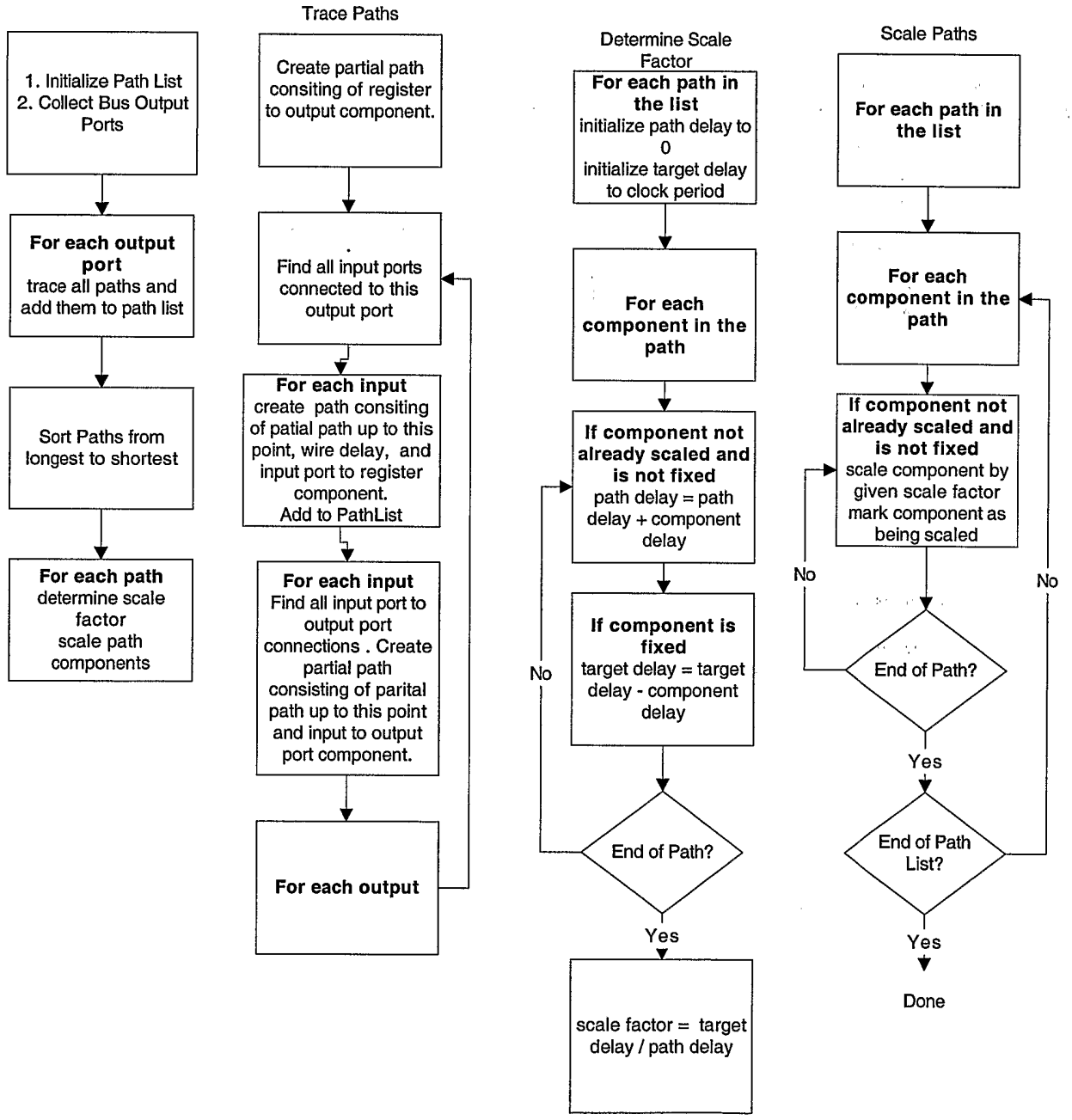


Figure 8: Flowchart for optimizing timing constraints

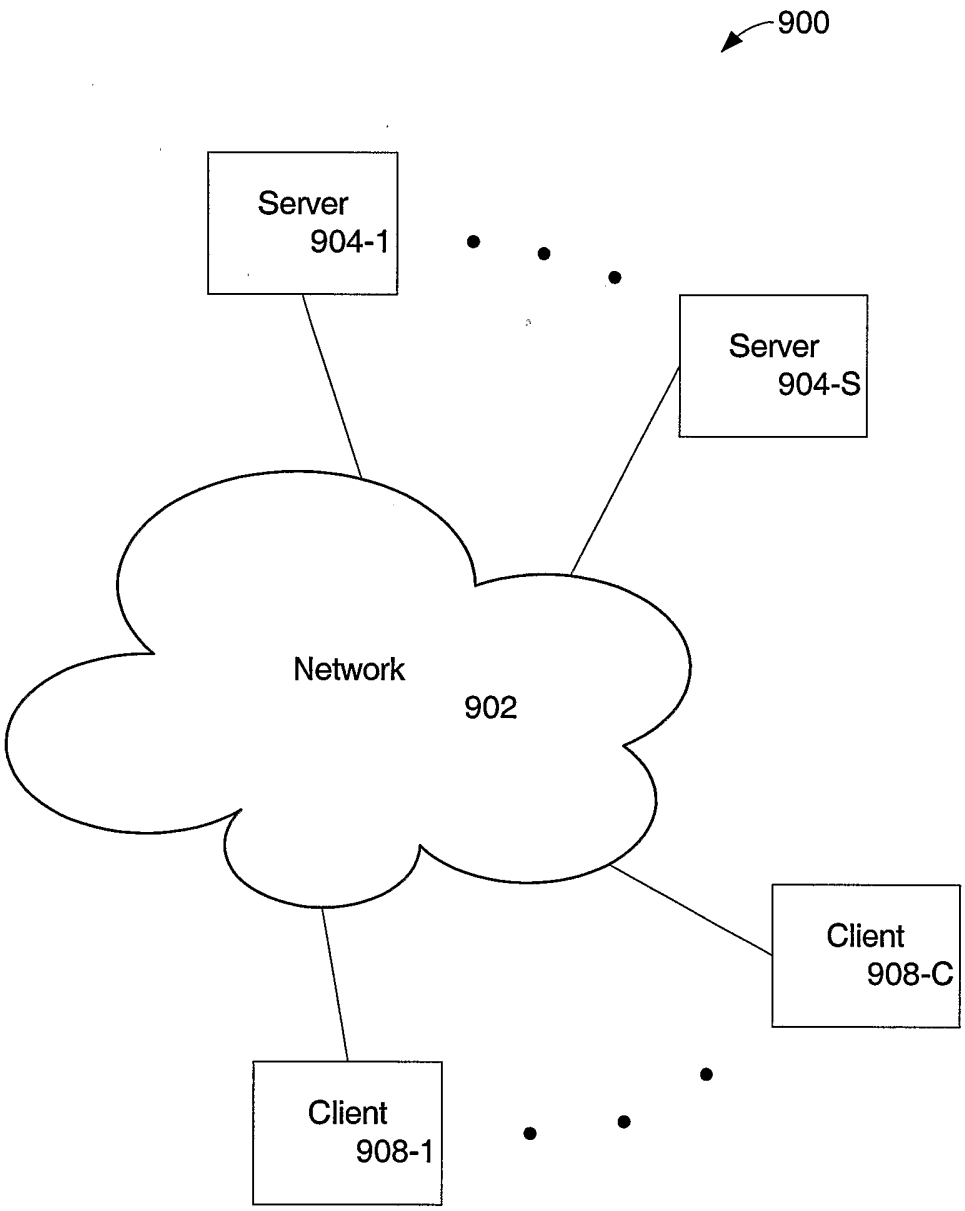


FIG. 9 (Prior Art)

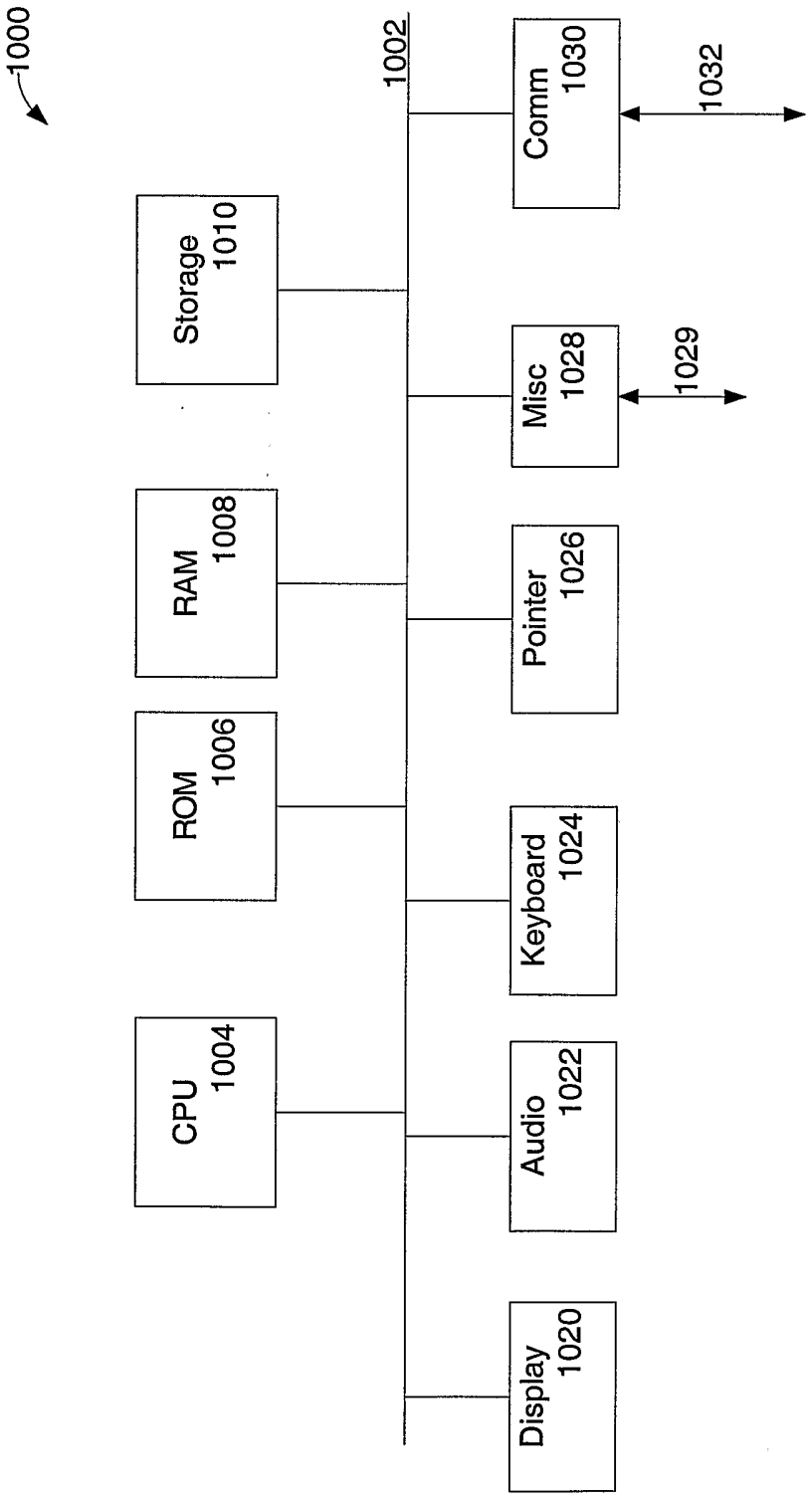


FIG. 10 (Prior Art)