



(19) **United States**

(12) **Patent Application Publication**
Schulz et al.

(10) **Pub. No.: US 2004/0130927 A1**

(43) **Pub. Date: Jul. 8, 2004**

(54) **PIPELINE ACCELERATOR HAVING
MULTIPLE PIPELINE UNITS AND RELATED
COMPUTING MACHINE AND METHOD**

Related U.S. Application Data

(60) Provisional application No. 60/422,503, filed on Oct. 31, 2002.

(75) Inventors: **Kenneth R. Schulz**, Manassas, VA (US); **John W. Rapp**, Manassas, VA (US); **Larry Jackson**, Manassas, VA (US); **Mark Jones**, Centreville, VA (US); **Troy Cherasaro**, Culpeper, VA (US)

Publication Classification

(51) **Int. Cl.⁷** **G11C 11/22**
(52) **U.S. Cl.** **365/145**

Correspondence Address:

Bryan A. Santarelli
GRAYBEAL JACKSON HALEY LLP
Suite 350
155 - 108th Avenue NE
Bellevue, WA 98004-5901 (US)

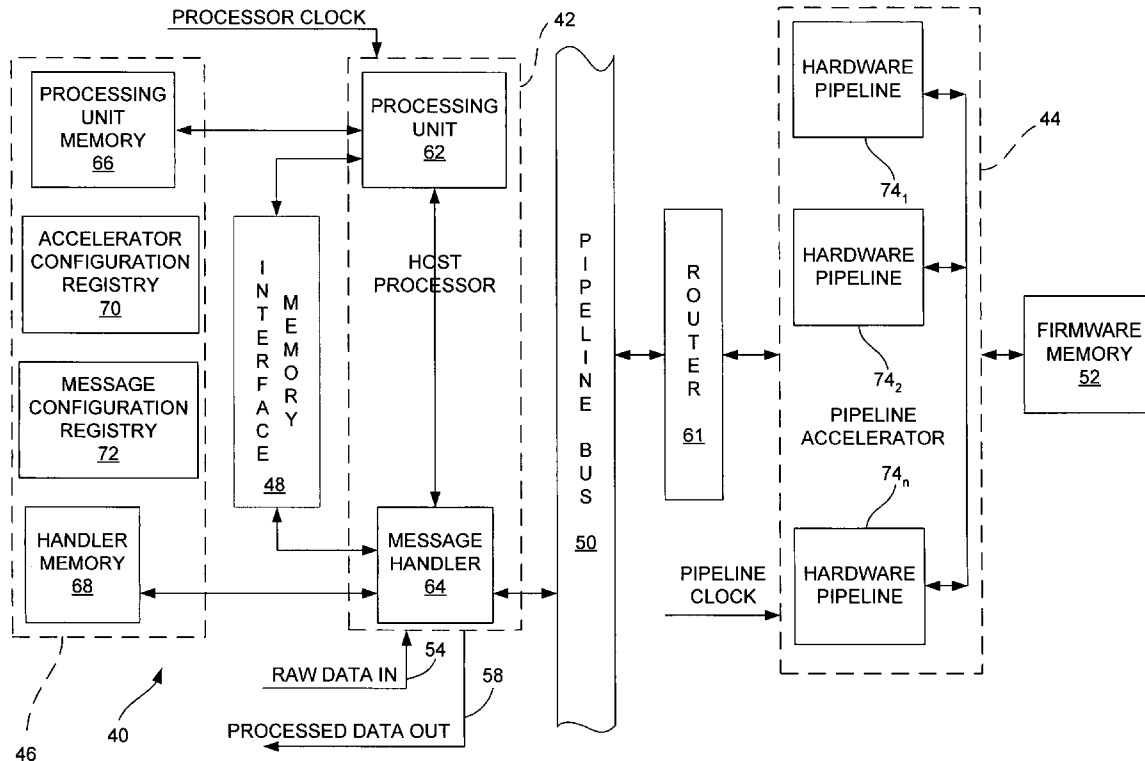
(57) **ABSTRACT**

A pipeline accelerator includes a bus and a plurality of pipeline units, each unit coupled to the bus and including at least one respective hardwired-pipeline circuit. By including a plurality of pipeline units in the pipeline accelerator, one can increase the accelerator's data-processing performance as compared to a single-pipeline-unit accelerator. Furthermore, by designing the pipeline units so that they communicate via a common bus, one can alter the number of pipeline units, and thus alter the configuration and functionality of the accelerator, by merely coupling or uncoupling pipeline units to or from the bus. This eliminates the need to design or redesign the pipeline-unit interfaces each time one alters one of the pipeline units or alters the number of pipeline units within the accelerator.

(73) Assignee: **Lockheed Martin Corporation**

(21) Appl. No.: **10/683,932**

(22) Filed: **Oct. 9, 2003**



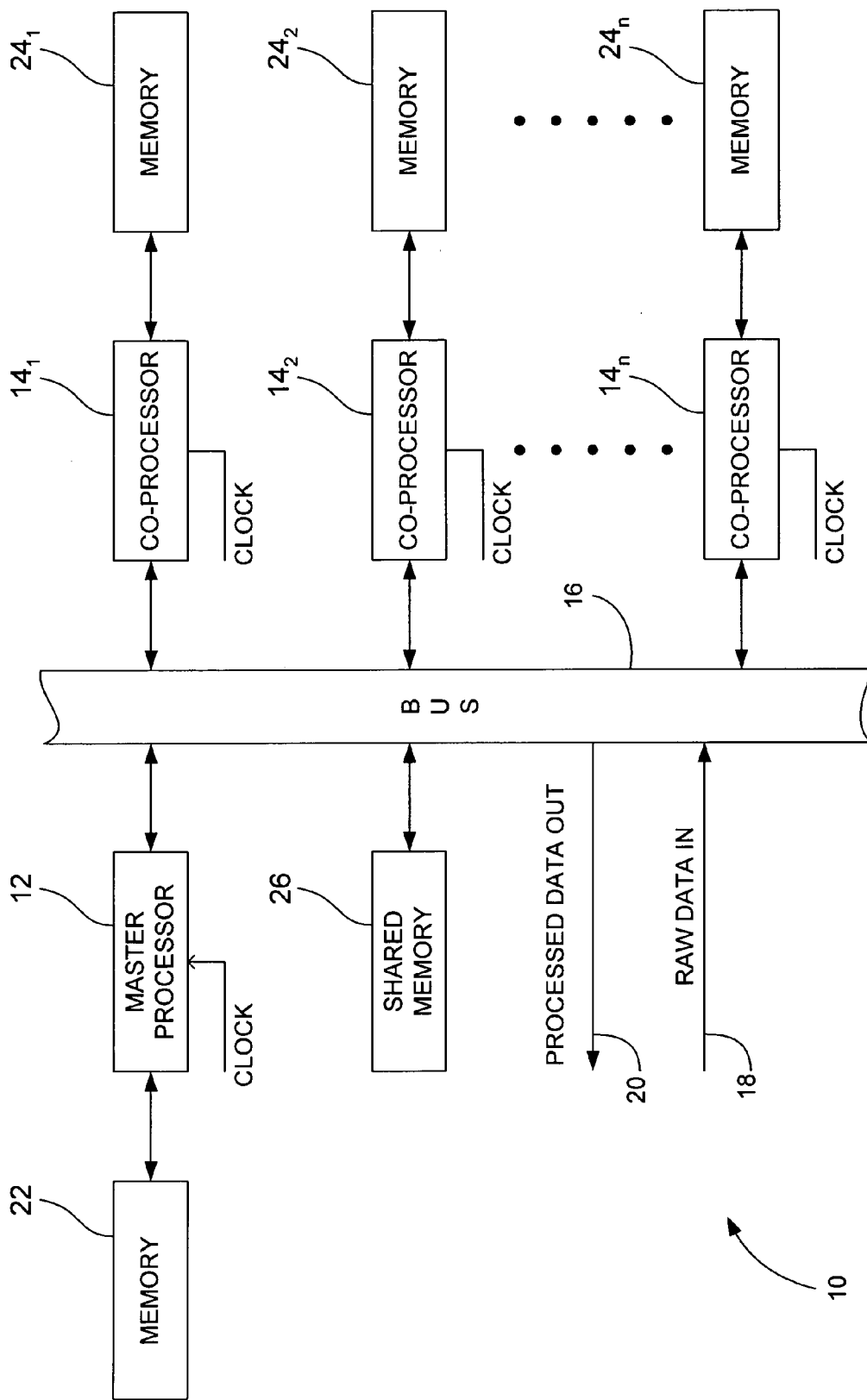


FIG. 1 (PRIOR ART)

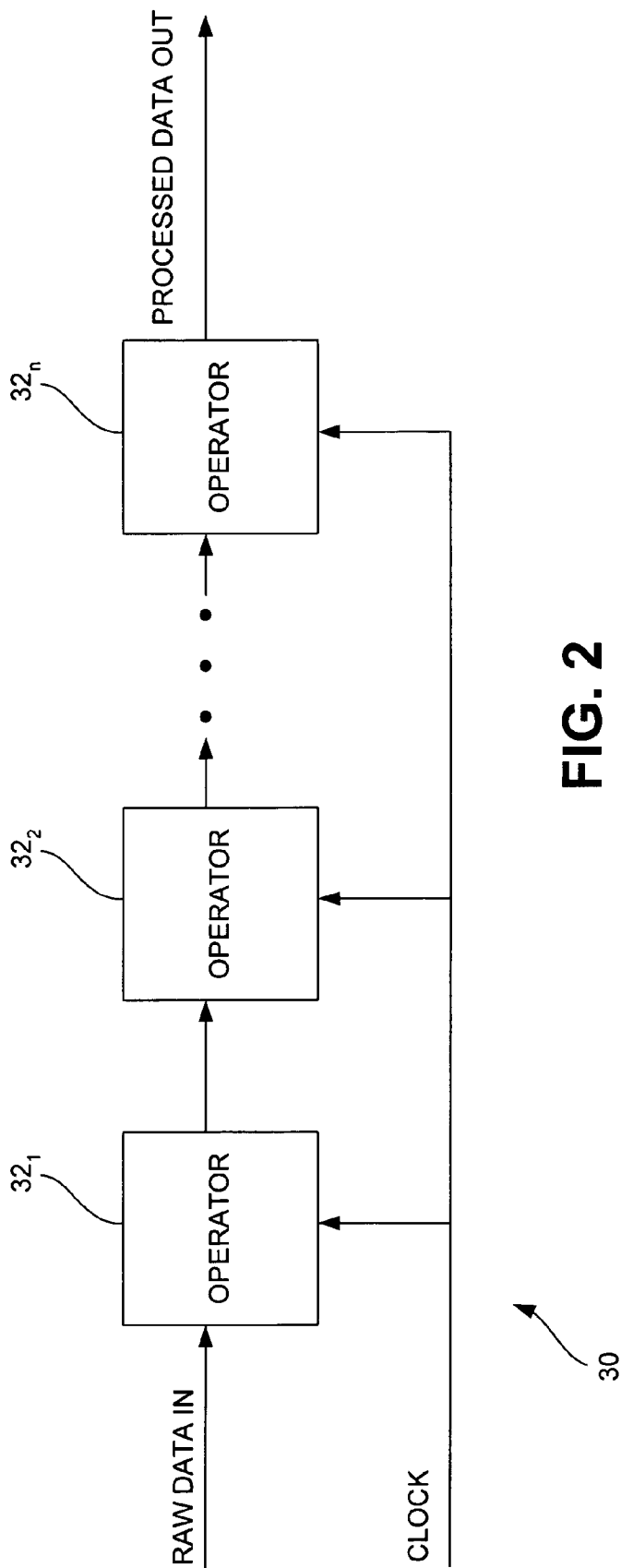


FIG. 2

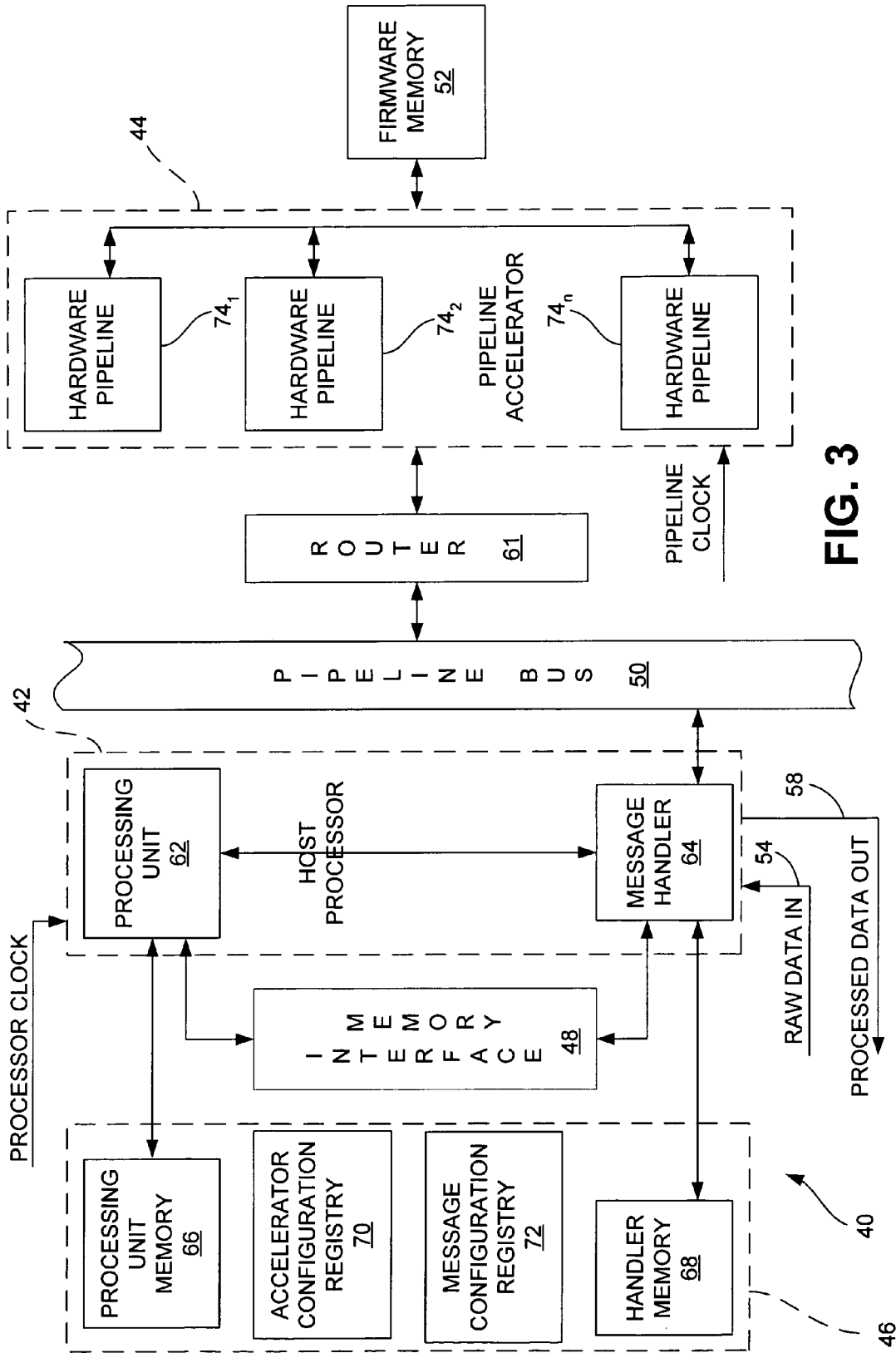


FIG. 3

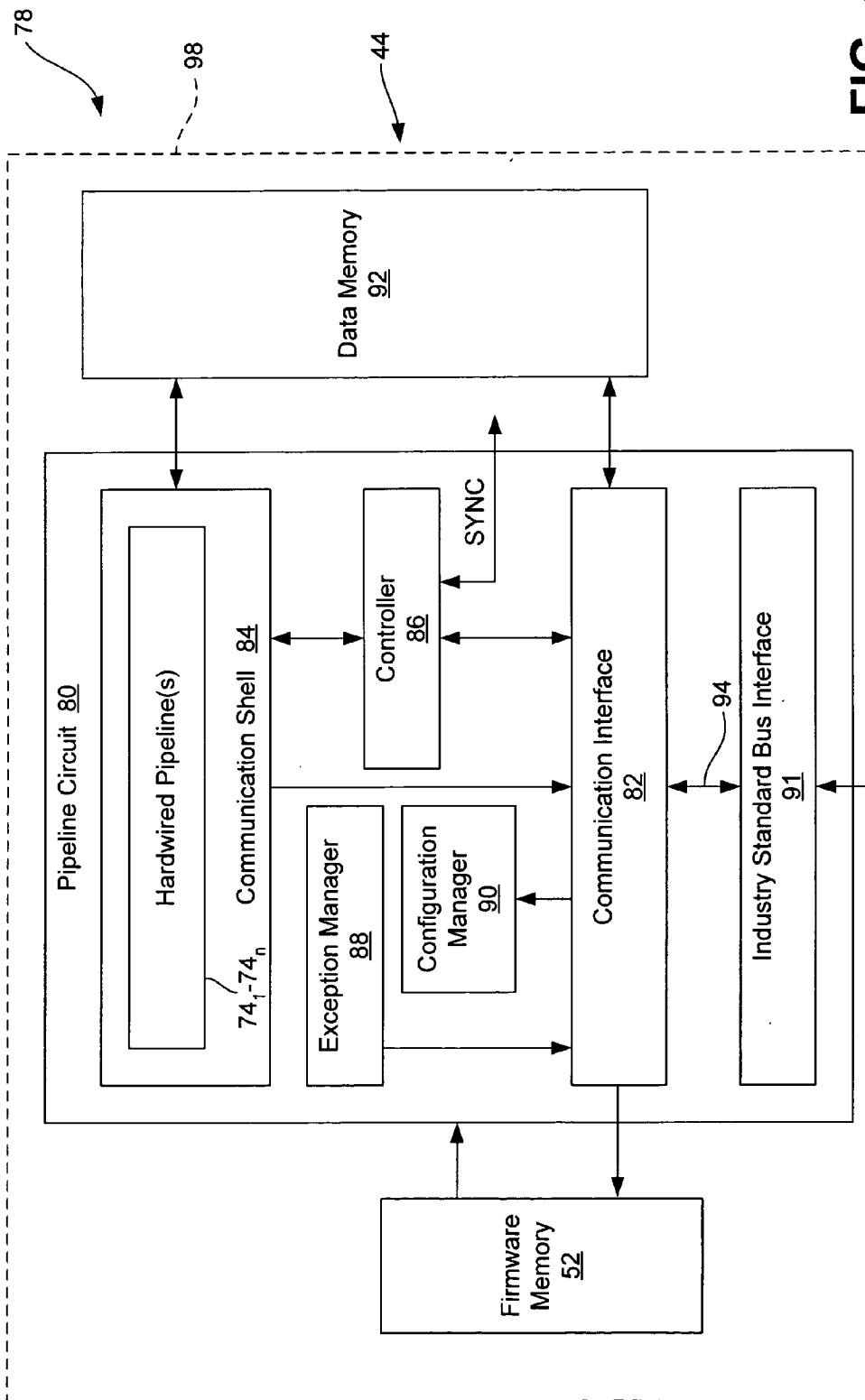


FIG. 4

To/From Pipeline Bus 50 or Router 61 (FIG. 3)

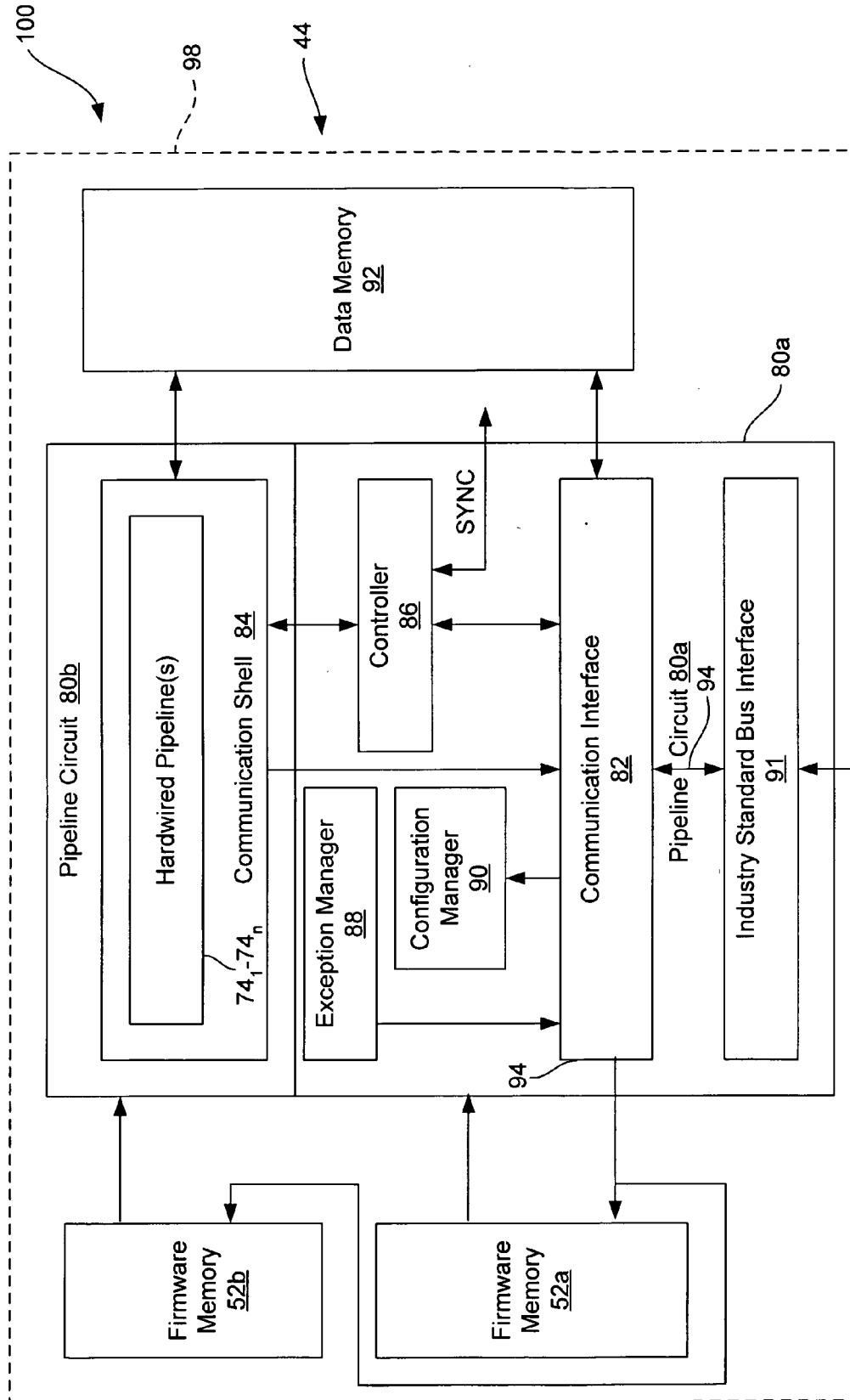


FIG. 5

To/From Pipeline Bus 50 or Router 61 (FIG. 3)

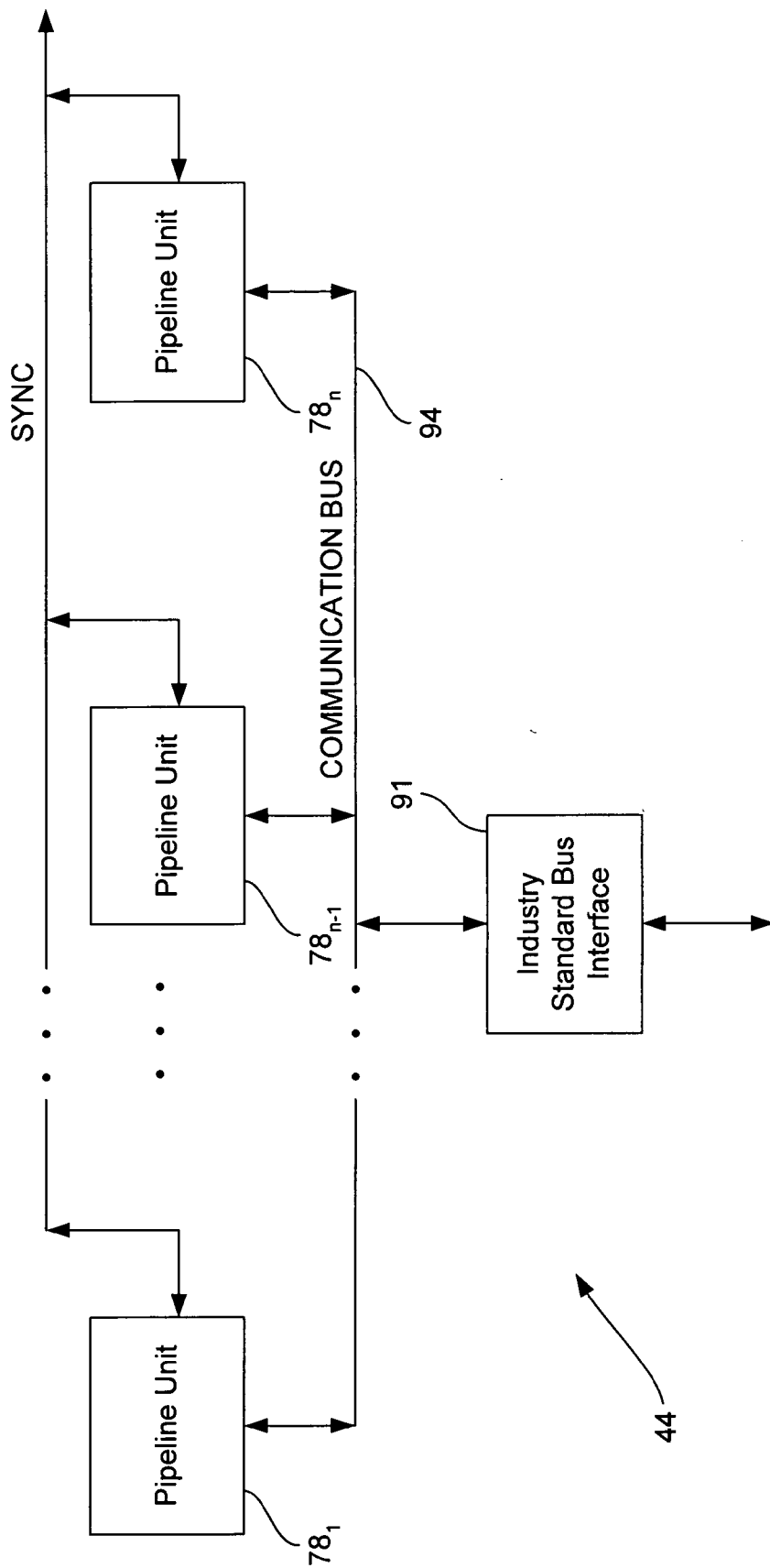


FIG. 6

To/From Pipeline Bus 50 or Router 61 (FIG.3)

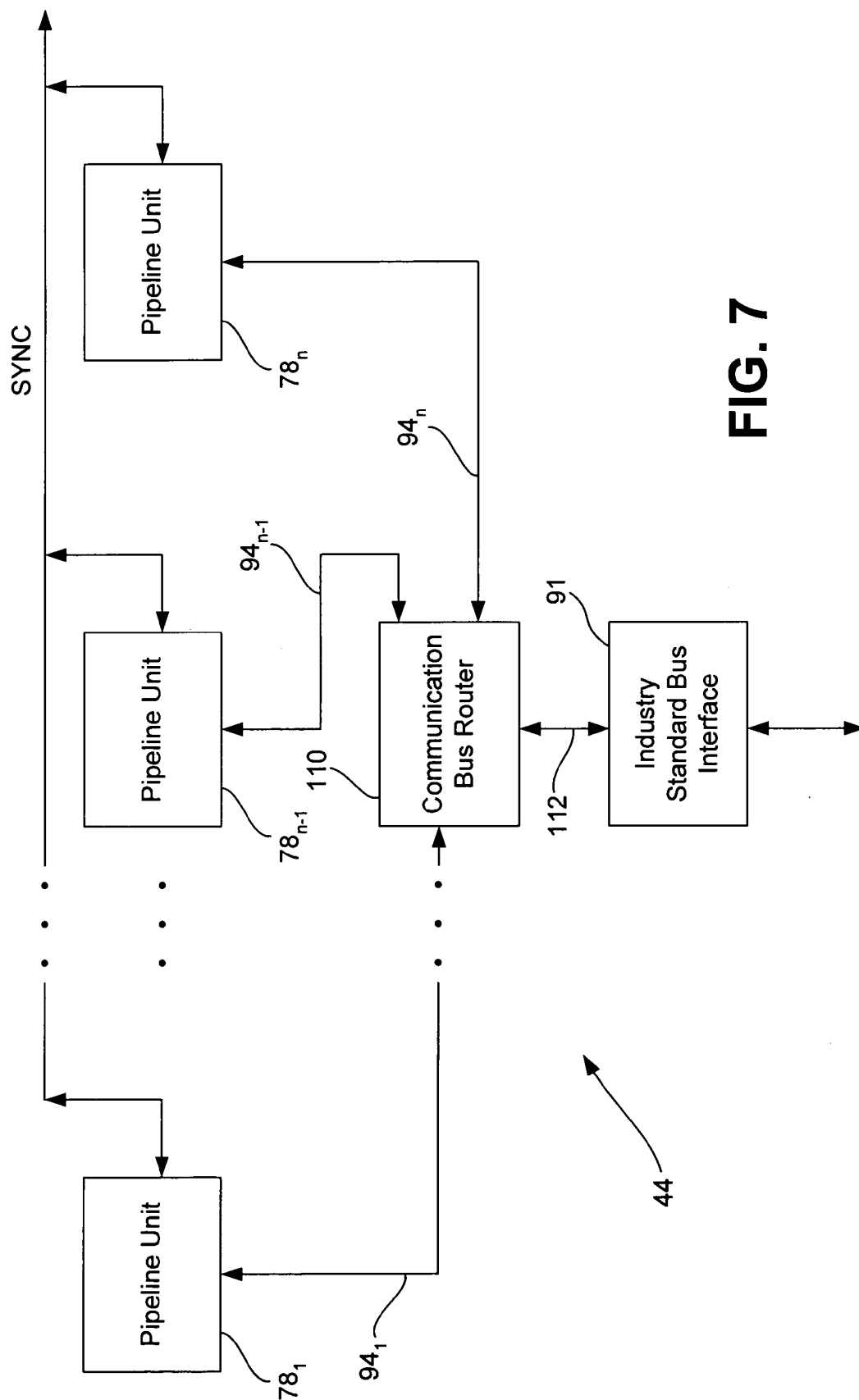


FIG. 7

To/From Pipeline Bus 50 or Router 61 (FIG.3)

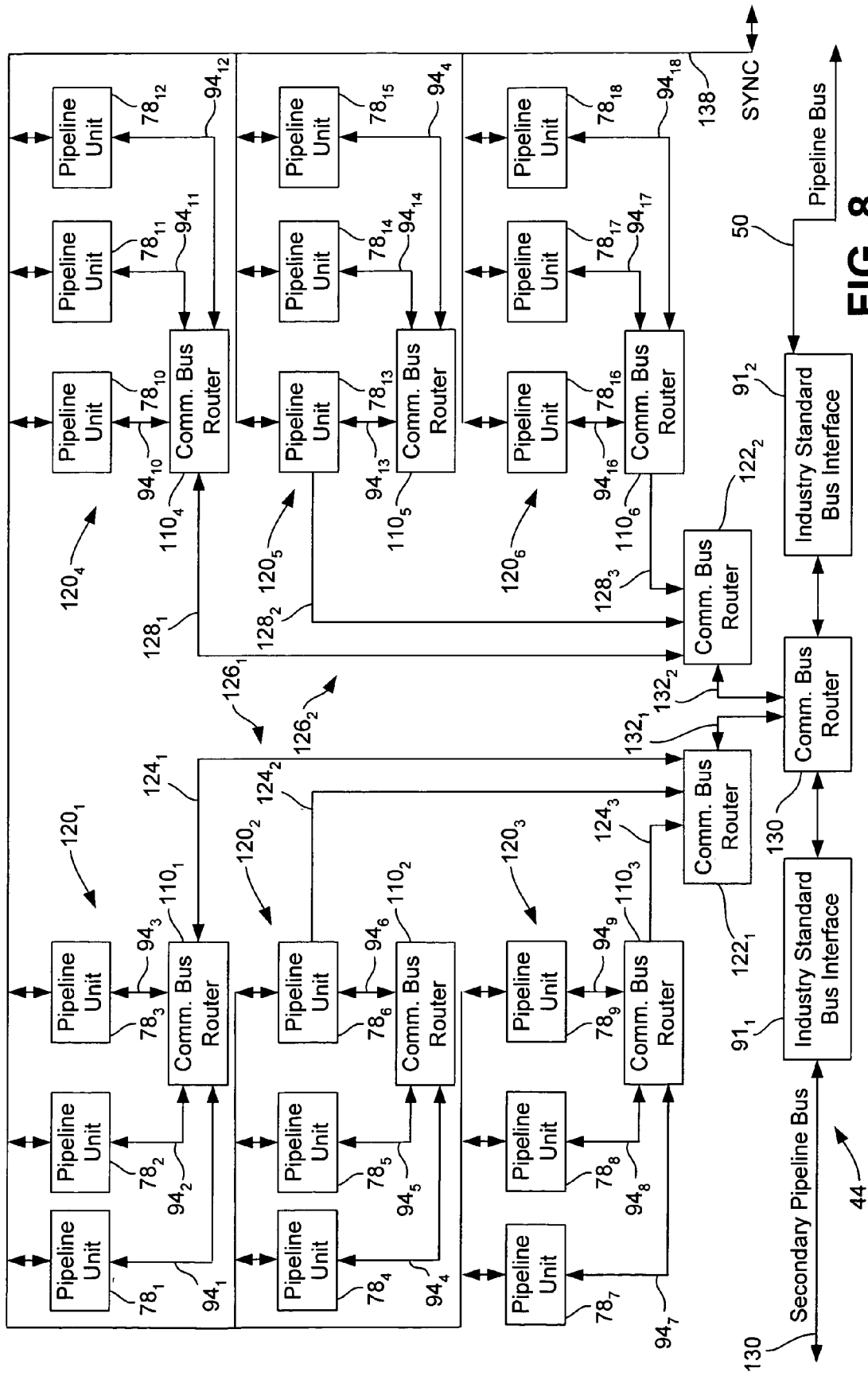


FIG. 8

PIPELINE ACCELERATOR HAVING MULTIPLE PIPELINE UNITS AND RELATED COMPUTING MACHINE AND METHOD

CLAIM OF PRIORITY

[0001] This application claims priority to U.S. Provisional Application Ser. No. 60/422,503, filed on Oct. 31, 2002, which is incorporated by reference.

CROSS REFERENCE TO RELATED APPLICATIONS

[0002] This application is related to U.S. Patent App. Ser. Nos. _____ entitled IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-11-3), _____ entitled COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-12-3), _____ entitled PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-13-3), and _____ entitled PROGRAMMABLE CIRCUIT AND RELATED COMPUTING MACHINE AND METHOD (Attorney Docket No. 1934-14-3), which have a common filing date and owner and which are incorporated by reference.

BACKGROUND

[0003] A common computing architecture for processing relatively large amounts of data in a relatively short period of time includes multiple interconnected processors that share the processing burden. By sharing the processing burden, these multiple processors can often process the data more quickly than a single processor can for a given clock frequency. For example, each of the processors can process a respective portion of the data or execute a respective portion of a processing algorithm.

[0004] FIG. 1 is a schematic block diagram of a conventional computing machine 10 having a multi-processor architecture. The machine 10 includes a master processor 12 and coprocessors 14₁-14_n, which communicate with each other and the master processor via a bus 16, an input port 18 for receiving raw data from a remote device (not shown in FIG. 1), and an output port 20 for providing processed data to the remote source. The machine 10 also includes a memory 22 for the master processor 12, respective memories 24₁-24_n for the coprocessors 14₁-14_n, and a memory 26 that the master processor and coprocessors share via the bus 16. The memory 22 serves as both a program and a working memory for the master processor 12, and each memory 24_i-24_n serves as both a program and a working memory for a respective coprocessor 14_i-14_n. The shared memory 26 allows the master processor 12 and the coprocessors 14 to transfer data among themselves, and from/to the remote device via the ports 18 and 20, respectively. The master processor 12 and the coprocessors 14 also receive a common clock signal that controls the speed at which the machine 10 processes the raw data.

[0005] In general, the computing machine 10 effectively divides the processing of raw data among the master processor 12 and the coprocessors 14. The remote source (not shown in FIG. 1) such as a sonar array loads the raw data

via the port 18 into a section of the shared memory 26, which acts as a first-in-first-out (FIFO) buffer (not shown) for the raw data. The master processor 12 retrieves the raw data from the memory 26 via the bus 16, and then the master processor and the coprocessors 14 process the raw data, transferring data among themselves as necessary via the bus 16. The master processor 12 loads the processed data into another FIFO buffer (not shown) defined in the shared memory 26, and the remote source retrieves the processed data from this FIFO via the port 20.

[0006] In an example of operation, the computing machine 10 processes the raw data by sequentially performing n+1 respective operations on the raw data, where these operations together compose a processing algorithm such as a Fast Fourier Transform (FFT). More specifically, the machine 10 forms a data-processing pipeline from the master processor 12 and the coprocessors 14. For a given frequency of the clock signal, such a pipeline often allows the machine 10 to process the raw data faster than a machine having only a single processor.

[0007] After retrieving the raw data from the raw-data FIFO (not shown) in the memory 26, the master processor 12 performs a first operation, such as a trigonometric function, on the raw data. This operation yields a first result, which the processor 12 stores in a first-result FIFO (not shown) defined within the memory 26. Typically, the processor 12 executes a program stored in the memory 22, and performs the above-described actions under the control of the program. The processor 12 may also use the memory 22 as working memory to temporarily store data that the processor generates at intermediate intervals of the first operation.

[0008] Next, after retrieving the first result from the first-result FIFO (not shown) in the memory 26, the coprocessor 14₁ performs a second operation, such as a logarithmic function, on the first result. This second operation yields a second result, which the coprocessor 14₁ stores in a second-result FIFO (not shown) defined within the memory 26. Typically, the coprocessor 14₁ executes a program stored in the memory 24₁, and performs the above-described actions under the control of the program. The coprocessor 14₁ may also use the memory 24₁ as working memory to temporarily store data that the coprocessor generates at intermediate intervals of the second operation.

[0009] Then, the coprocessors 24₂-24_n sequentially perform third—nth operations on the second—(n-1)th results in a manner similar to that discussed above for the coprocessor 24₁.

[0010] The nth operation, which is performed by the coprocessor 24_n, yields the final result, i.e., the processed data. The coprocessor 24_n loads the processed data into a processed-data FIFO (not shown) defined within the memory 26, and the remote device (not shown in FIG. 1) retrieves the processed data from this FIFO.

[0011] Because the master processor 12 and coprocessors 14 are simultaneously performing different operations of the processing algorithm, the computing machine 10 is often able to process the raw data faster than a computing machine having a single processor that sequentially performs the different operations. Specifically, the single processor cannot retrieve a new set of the raw data until it performs all n+1

operations on the previous set of raw data. But using the pipeline technique discussed above, the master processor **12** can retrieve a new set of raw data after performing only the first operation. Consequently, for a given clock frequency, this pipeline technique can increase the speed at which the machine **10** processes the raw data by a factor of approximately $n+1$ as compared to a single-processor machine (not shown in **FIG. 1**).

[0012] Alternatively, the computing machine **10** may process the raw data in parallel by simultaneously performing $n+1$ instances of a processing algorithm, such as an FFT, on the raw data. That is, if the algorithm includes $n+1$ sequential operations as described above in the previous example, then each of the master processor **12** and the coprocessors **14** sequentially perform all $n+1$ operations on respective sets of the raw data. Consequently, for a given clock frequency, this parallel-processing technique, like the above-described pipeline technique, can increase the speed at which the machine **10** processes the raw data by a factor of approximately $n+1$ as compared to a single-processor machine (not shown in **FIG. 1**).

[0013] Unfortunately, although the computing machine **10** can process data more quickly than a single-processor computing machine (not shown in **FIG. 1**), the data-processing speed of the machine **10** is often significantly less than the frequency of the processor clock. Specifically, the data-processing speed of the computing machine **10** is limited by the time that the master processor **12** and coprocessors **14** require to process data. For brevity, an example of this speed limitation is discussed in conjunction with the master processor **12**, although it is understood that this discussion also applies to the coprocessors **14**. As discussed above, the master processor **12** executes a program that controls the processor to manipulate data in a desired manner. This program includes a sequence of instructions that the processor **12** executes. Unfortunately, the processor **12** typically requires multiple clock cycles to execute a single instruction, and often must execute multiple instructions to process a single value of data. For example, suppose that the processor **12** is to multiply a first data value A (not shown) by a second data value B (not shown). During a first clock cycle, the processor **12** retrieves a multiply instruction from the memory **22**. During second and third clock cycles, the processor **12** respectively retrieves A and B from the memory **26**. During a fourth clock cycle, the processor **12** multiplies A and B, and, during a fifth clock cycle, stores the resulting product in the memory **22** or **26** or provides the resulting product to the remote device (not shown). This is a best-case scenario, because in many cases the processor **12** requires additional clock cycles for overhead tasks such as initializing and closing counters. Therefore, at best the processor **12** requires five clock cycles, or an average of 2.5 clock cycles per data value, to process A and B..

[0014] Consequently, the speed at which the computing machine **10** processes data is often significantly lower than the frequency of the clock that drives the master processor **12** and the coprocessors **14**. For example, if the processor **12** is clocked at 1.0 Gigahertz (GHz) but requires an average of 2.5 clock cycles per data value, then the effective data-processing speed equals $(1.0 \text{ GHz})/2.5=0.4 \text{ GHz}$. This effective data-processing speed is often characterized in units of operations per second. Therefore, in this example, for a

clock speed of 1.0 GHz, the processor **12** would be rated with a data-processing speed of 0.4 Gigaoperations/second (Gops).

[0015] **FIG. 2** is a block diagram of a hardwired data pipeline **30** that can typically process data faster than a processor can for a given clock frequency, and often at substantially the same rate at which the pipeline is clocked. The pipeline **30** includes operator circuits **32₁-32_n**, which each perform a respective operation on respective data without executing program instructions. That is, the desired operation is “burned in” to a circuit **32** such that it implements the operation automatically, without the need of program instructions. By eliminating the overhead associated with executing program instructions, the pipeline **30** can typically perform more operations per second than a processor can for a given clock frequency.

[0016] For example, the pipeline **30** can often solve the following equation faster than a processor can for a given clock frequency:

$$Y(x_k)=(5x_k+3)2^{xk}$$

[0017] where x_k represents a sequence of raw data values. In this example, the operator circuit **32₁** is a multiplier that calculates $5x_k$, the circuit **32₂** is an adder that calculates $5x_k+3$, and the circuit **32_n** ($n=3$) is a multiplier that calculates $(5x_k+3)2^{xk}$.

[0018] During a first clock cycle $k=1$, the circuit **32₁** receives data value x_1 and multiplies it by 5 to generate $5x_1$.

[0019] During a second clock cycle $k=2$, the circuit **32₂** receives $5x_1$ from the circuit **32₁** and adds 3 to generate $5x_1+3$. Also, during the second clock cycle, the circuit **32₁** generates $5x_2$.

[0020] During a third clock cycle $k=3$, the circuit **32₃** receives $5x_1+3$ from the circuit **32₂** and multiplies by 2^{x1} (effectively left shifts $5x_1+3$ by x_1) to generate the first result $(5x_1+3)2^{x1}$. Also during the third clock cycle, the circuit **32₁** generates $5x_3$ and the circuit **32₂** generates $5x_2+3$.

[0021] The pipeline **30** continues processing subsequent raw data values x_k in this manner until all the raw data values are processed.

[0022] Consequently, a delay of two clock cycles after receiving a raw data value x_1 —this delay is often called the latency of the pipeline **30**—the pipeline generates the result $(5x_1+3)2^{x1}$, and thereafter generates one result—e.g., $(5x_2+3)2^{x2}$, $(5x_3+3)2^{x3}$, . . . , $(5x_n+3)2^{xn}$ —each clock cycle.

[0023] Disregarding the latency, the pipeline **30** thus has a data-processing speed equal to the clock speed. In comparison, assuming that the master processor **12** and coprocessors **14** (**FIG. 1**) have data-processing speeds that are 0.4 times the clock speed as in the above example, the pipeline **30** can process data 2.5 times faster than the computing machine **10** (**FIG. 1**) for a given clock speed.

[0024] Still referring to **FIG. 2**, a designer may choose to implement the pipeline **30** in a programmable logic IC (PLIC), such as a field-programmable gate array (FPGA), because a PLIC allows more design and modification flexibility than does an application specific IC (ASIC). To configure the hardwired connections within a PLIC, the designer merely sets interconnection-configuration registers disposed within the PLIC to predetermined binary states.

The combination of all these binary states is often called “firmware.” Typically, the designer loads this firmware into a nonvolatile memory (not shown in **FIG. 2**) that is coupled to the PLIC. When one “turns on” the PLIC, it downloads the firmware from the memory into the interconnection-configuration registers. Therefore, to modify the functioning of the PLIC, the designer merely modifies the firmware and allows the PLIC to download the modified firmware into the interconnection-configuration registers. This ability to modify the PLIC by merely modifying the firmware is particularly useful during the prototyping stage and for upgrading the pipeline **30** “in the field”.

[0025] Unfortunately, the hardwired pipeline **30** may not be the best choice to execute algorithms that entail significant decision making, particularly nested decision making. A processor can typically execute a nested-decision-making instruction (e.g., a nested conditional instruction such as “if A, then do B, else if C, do D, . . . , else do n”) approximately as fast as it can execute an operational instruction (e.g., “A+B”) of comparable length. But although the pipeline **30** may be able to make a relatively simple decision (e.g., “A>B?”) efficiently, it typically cannot execute a nested decision (e.g., “if A, then do B, else if C, do D, . . . , else do n”) as efficiently as a processor can. One reason for this inefficiency is that the pipeline **30** may have little on-board memory, and thus may need to access external working/program memory (not shown). And although one may be able to design the pipeline **30** to execute such a nested decision, the size and complexity of the required circuitry often makes such a design impractical, particularly where an algorithm includes multiple different nested decisions.

[0026] Consequently, processors are typically used in applications that require significant decision making, and hardwired pipelines are typically limited to “number crunching” applications that entail little or no decision making.

[0027] Furthermore, as discussed below, it is typically much easier for one to design/modify a processor-based computing machine, such as the computing machine **10** of **FIG. 1**, than it is to design/modify a hardwired pipeline such as the pipeline **30** of **FIG. 2**, particularly where the pipeline **30** includes multiple PLICs.

[0028] Computing components, such as processors and their peripherals (e.g., memory), typically include industry-standard communication interfaces that facilitate the interconnection of the components to form a processor-based computing machine.

[0029] Typically, a standard communication interface includes two layers: a physical layer and a services layer.

[0030] The physical layer includes the circuitry and the corresponding circuit interconnections that form the interface and the operating parameters of this circuitry. For example, the physical layer includes the pins that connect the component to a bus, the buffers that latch data received from the pins, and the drivers that drive signals onto the pins. The operating parameters include the acceptable voltage range of the data signals that the pins receive, the signal timing for writing and reading data, and the supported modes of operation (e.g., burst mode, page mode). Conventional physical layers include transistor-transistor logic (TTL) and RAMBUS.

[0031] The services layer includes the protocol by which a computing component transfers data. The protocol defines

the format of the data and the manner in which the component sends and receives the formatted data. Conventional communication protocols include file-transfer protocol (FTP) and transmission control protocol/internet protocol (TCP/IP).

[0032] Consequently, because manufacturers and others typically design computing components having industry-standard communication interfaces, one can typically design the interface of such a component and interconnect it to other computing components with relatively little effort. This allows one to devote most of his time to designing the other portions of the computing machine, and to easily modify the machine by adding or removing components.

[0033] Designing a computing component that supports an industry-standard communication interface allows one to save design time by using an existing physical-layer design from a design library. This also insures that he/she can easily interface the component to off-the-shelf computing components.

[0034] And designing a computing machine using computing components that support a common industry-standard communication interface allows the designer to interconnect the components with little time and effort. Because the components support a common interface, the designer can interconnect them via a system bus with little design effort. And because the supported interface is an industry standard, one can easily modify the machine. For example, one can add different components and peripherals to the machine as the system design evolves, or can easily add/design next-generation components as the technology evolves. Furthermore, because the components support a common industry-standard services layer, one can incorporate into the computing machine’s software an existing software module that implements the corresponding protocol. Therefore, one can interface the components with little effort because the interface design is essentially already in place, and thus can focus on designing the portions (e.g., software) of the machine that cause the machine to perform the desired function(s).

[0035] But unfortunately, there are no known industry-standard services layers for components, such as PLICs, used to form hardwired pipelines such as the pipeline **30** of **FIG. 2**.

[0036] Consequently, to design a pipeline having multiple PLICs, one typically spends a significant amount of time and exerts a significant effort designing “from scratch” and debugging the services layer of the communication interface between the PLICs. Typically, such an ad hoc services layer depends on the parameters of the data being transferred between the PLICs. Likewise, to design a pipeline that interfaces to a processor, one would have to spend a significant amount of time and exert a significant effort in designing and debugging the services layer of the communication interface between the pipeline and the processor.

[0037] Similarly, to modify such a pipeline by adding a PLIC to it, one typically spends a significant amount of time and exerts a significant effort designing and debugging the services layer of the communication interface between the added PLIC and the existing PLICs. Likewise, to modify a pipeline by adding a processor, or to modify a computing machine by adding a pipeline, one would have to spend a

significant amount of time and exert a significant effort in designing and debugging the services layer of the communication interface between the pipeline and processor.

[0038] Consequently, referring to **FIGS. 1 and 2**, because of the difficulties in interfacing multiple PLICs and in interfacing a processor to a pipeline, one is often forced to make significant tradeoffs when designing a computing machine. For example, with a processor-based computing machine, one is forced to trade number-crunching speed and design/modification flexibility for complex decision-making ability. Conversely, with a hardwired pipeline-based computing machine, one is forced to trade complex-decision-making ability and design/modification flexibility for number-crunching speed. Furthermore, because of the difficulties in interfacing multiple PLICs, it is often impractical for one to design a pipeline-based machine having more than a few PLICs. As a result, a practical pipeline-based machine often has limited functionality. And because of the difficulties in interfacing a processor to a PLIC, it would be impractical to interface a processor to more than one PLIC. As a result, the benefits obtained by combining a processor and a pipeline would be minimal.

[0039] Therefore, a need has arisen for a new computing architecture that allows one to combine the decision-making ability of a processor-based machine with the number-crunching speed of a hardwired-pipeline-based machine.

SUMMARY

[0040] According to an embodiment of the invention, a pipeline accelerator includes a bus and a plurality of pipeline units each coupled to the bus and each including at least one respective hardwired-pipeline circuit.

[0041] By including a plurality of pipeline units in the pipeline accelerator, one can increase the accelerator's data-processing performance as compared to a single-pipeline-unit accelerator. Furthermore, by designing the pipeline units so that they communicate with each other and with other peers via a common bus, one can alter the number of pipeline units, and thus alter the configuration and functionality of the accelerator, by merely coupling or uncoupling pipeline units to or from the bus. This eliminates the need to design or redesign the pipeline-unit interfaces each time one alters one of the pipeline units or alters the number of pipeline units within the accelerator.

BRIEF DESCRIPTION OF THE DRAWINGS

[0042] **FIG. 1** is a block diagram of a computing machine having a conventional multi-processor architecture.

[0043] **FIG. 2** is a block diagram of a conventional hardwired pipeline.

[0044] **FIG. 3** is a block diagram of a computing machine having a peer-vector architecture according to an embodiment of the invention.

[0045] **FIG. 4** is a block diagram of a pipeline unit of the pipeline accelerator of **FIG. 3** according to an embodiment of the invention.

[0046] **FIG. 5** is a block diagram of a pipeline unit of the pipeline accelerator of **FIG. 3** according to another embodiment of the invention.

[0047] **FIG. 6** is a block diagram of the pipeline accelerator of **FIG. 3** including multiple pipeline units according to an embodiment of the invention.

[0048] **FIG. 7** is a block diagram of the pipeline accelerator of **FIG. 3** including multiple pipeline units according to another embodiment of the invention.

[0049] **FIG. 8** is a block diagram of the pipeline accelerator of **FIG. 3** including groups of multiple pipeline units according to an embodiment of the invention.

DETAILED DESCRIPTION

[0050] **FIG. 3** is a schematic block diagram of a computing machine **40**, which has a peer-vector architecture according to an embodiment of the invention. In addition to a host processor **42**, the peer-vector machine **40** includes a pipeline accelerator **44**, which performs at least a portion of the data processing, and which thus effectively replaces the bank of coprocessors **14** in the computing machine **10** of **FIG. 1**. Therefore, the host-processor **42** and the accelerator **44** (or units thereof as discussed below) are "peers" that can transfer data vectors back and forth. Because the accelerator **44** does not execute program instructions, it typically performs mathematically intensive operations on data significantly faster than a bank of coprocessors can for a given clock frequency. Consequently, by combining the decision-making ability of the processor **42** and the number-crunching ability of the accelerator **44**, the machine **40** has the same abilities as, but can often process data faster than, a conventional computing machine such as the machine **10**. Furthermore, as discussed below, providing the accelerator **44** with a communication interface that is compatible with the communication interface of the host processor **42** facilitates the design and modification of the machine **40**, particularly where the processor's communication interface is an industry standard. And where the accelerator **44** includes multiple pipeline units (e.g., PLIC-based circuits), providing each of these units with the same communication interface facilitates the design and modification of the accelerator, particularly where the communication interfaces are compatible with an industry-standard interface. Moreover, the machine **40** may also provide other advantages as described below and in the previously cited patent applications.

[0051] Still referring to **FIG. 3**, in addition to the host processor **42** and the pipeline accelerator **44**, the peer-vector computing machine **40** includes a processor memory **46**, an interface memory **48**, a pipeline bus **50**, one or more firmware memories **52**, an optional raw-data input port **54**, a processed-data output port **58**, and an optional router **61**.

[0052] The host processor **42** includes a processing unit **62** and a message handler **64**, and the processor memory **46** includes a processing-unit memory **66** and a handler memory **68**, which respectively serve as both program and working memories for the processor unit and the message handler. The processor memory **46** also includes an accelerator-configuration registry **70** and a message-configuration registry **72**, which store respective configuration data that allow the host processor **42** to configure the functioning of the accelerator **44** and the format of the messages that the message handler **64** sends and receives.

[0053] The pipeline accelerator **44** is disposed on at least one PLIC (**FIG. 4**) and includes hardwired pipelines **74₁-**

74_n, which process respective data without executing program instructions. The firmware memory 52 stores the configuration firmware for the accelerator 44. If the accelerator 44 is disposed on multiple PLICs, then these PLICs and their respective firmware memories may be disposed in multiple pipeline units, which are discussed further below in conjunction with FIGS. 4-8. Alternatively, the accelerator 44 may be disposed on at least one ASIC, and thus may have internal interconnections that are unconfigurable once the ASIC is formed. In this alternative, the machine 40 may omit the firmware memory 52. Furthermore, although the accelerator 44 is shown including multiple pipelines 74₁-74_n, it may include only a single pipeline. In addition, although not shown, the accelerator 44 may include one or more processors such as a digital-signal processor (DSP). Moreover, although not shown, the accelerator 44 may include a data input port and/or a data output port.

[0054] And although the host processor 42 and pipeline accelerator 44 are discussed as being disposed on different ICs, the host processor and pipeline accelerator may be disposed on the same IC.

[0055] The general operation of the peer-vector machine 40 is discussed in previously cited U.S. patent app. Ser. No. _____ entitled IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-11-3), the structure and operation of the host processor 42 is discussed in previously cited U.S. patent app. Ser. No. _____ entitled COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-12-3), and the structure and operation of the pipeline accelerator 44 is discussed in previously cited U.S. patent app. Ser. No. _____ entitled PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-13-3) and below in conjunction with FIGS. 4-8.

[0056] FIG. 4 is a block diagram of a unit 78 of the pipeline accelerator 44 of FIG. 3 according to an embodiment of the invention.

[0057] The accelerator 44 includes one or more such pipeline units 78 (only one shown in FIG. 4), each of which includes a pipeline circuit 80, such as a PLIC or an ASIC. As discussed further below and in previously cited U.S. patent app. Ser. No. _____ entitled PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-13-3), each pipeline unit 78 is a "peer" of the host processor 42 (FIG. 3) and of the other pipeline units of the accelerator 44. That is, each pipeline unit 78 can communicate directly with the host processor 42 or with any other pipeline unit. Thus, this peer-vector architecture prevents data "bottlenecks" that otherwise might occur if all of the pipeline units 78 communicated through a central location such as a master pipeline unit (not shown) or the host processor 42. Furthermore, this architecture allows one to add or remove peers from the peer-vector machine 40 (FIG. 3) without significant modifications to the machine.

[0058] The pipeline circuit 80 includes a communication interface 82, which transfers data between a peer, such as the host processor 42 (FIG. 3), and the following other components of the pipeline circuit: the hardwired pipelines

74₁-74_n via a communication shell 84, a pipeline controller 86, an exception manager 88, and a configuration manager 90. The pipeline circuit 80 may also include an industry-standard bus interface 91 and a communication bus 93, which connects the interface 82 to the interface 91. Alternatively, the functionality of the interface 91 may be included within the communication interface 82 and the bus 93 omitted.

[0059] By designing the components of the pipeline circuit 80 as separate modules, one can often simplify the design of the pipeline circuit. That is, one can design and test each of these components separately, and then integrate them much like one does when designing software or a processor-based computing system (such as the system 10 of FIG. 1). In addition, one can save in a library (not shown) hardware description language (HDL) that defines these components—particularly components, such as the communication interface 82, that designers will probably use frequently in other pipeline designs—thus reducing the design and test time of future pipeline designs that use the same components. That is, by using the HDL from the library, the designer need not redesign previously implemented components from scratch, and thus can focus his efforts on the design of components that were not previously implemented, or on the modification of previously implemented components. Moreover, one can save in the library HDL that defines multiple versions of the pipeline circuit 80 or multiple versions of the entire pipeline accelerator 44 so that one can pick and choose among existing designs.

[0060] Still referring to FIG. 4, the communication interface 82 sends and receives (via the bus interface 91 of present) data in a format recognized by the message handler 64 (FIG. 3), and thus typically facilitates the design and modification of the peer-vector machine 40 (FIG. 3). For example, if the data format is an industry standard such as the Rapid I/O format, then one need not design a custom interface between the host processor 42 and the pipeline unit 78. Furthermore, by allowing the pipeline unit 78 to communicate with other peers, such as the host processor 42 (FIG. 3), via the pipeline bus 50 instead of via a non-bus interface, one can change the number of pipeline units by merely connecting or disconnecting them (or the circuit cards that hold them) to the pipeline bus instead of redesigning a non-bus interface from scratch each time a pipeline unit is added or removed.

[0061] The hardwired pipelines 74₁-74_n perform respective operations on data as discussed above in conjunction with FIG. 3 and in previously cited U.S. patent app. Ser. Nos. _____ entitled IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-11-3) and _____ entitled PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-13-3), and the communication shell 84 interfaces the pipelines to the other components of the pipeline circuit 80 and to other circuits (such as a data memory 92 discussed below) of the pipeline unit 78.

[0062] The controller 86 synchronizes the hardwired pipelines 74₁-74_n in response to SYNC signals and special pipeline-bus communications (i.e., "events") from other peers, and monitors and controls the sequence in which the

pipelines perform their respective data operations. For example, a peer, such as the host processor 42, may pulse a SYNC signal or send an event to the pipeline unit 78 via the pipeline bus 50 to indicate that the peer has finished sending a block of data to the pipeline unit and to cause the hardwired pipelines 74₁-74_n to begin processing this data. Typically, a SYNC signal is used to synchronize a time-critical operation, and an event is used to synchronize a non-time-critical operation. Typically, an event is a data-less communication that is often called a “doorbell”. But an event may include data, in which case it is often called an “event message”. SYNC signals and events are discussed further in previously cited U.S. patent app. Ser. No. _____ entitled PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-13-3).

[0063] The exception manager 88 monitors the status of the hardwired pipelines 74₁-74_n, the communication interface 82, the communication shell 84, the controller 86, and the bus interface 91 (if present), and reports exceptions to the host processor 42 (FIG. 3). For example, if a buffer in the communication interface 82 overflows, then the exception manager 88 reports this to the host processor 42. The exception manager 88 may also correct, or attempt to correct, the problem giving rise to the exception. For example, for an overflowing buffer, the exception manager 88 may increase the size of the buffer, either directly or via the configuration manager 90 as discussed below.

[0064] The configuration manager 90 sets the soft configuration of the hardwired pipelines 74₁-74_n, the communication interface 82, the communication shell 84, the controller 86, the exception manager 88, and the interface 91 (if present) in response to soft-configuration data from the host processor 42 (FIG. 3)—as discussed in previously cited U.S. patent app. Ser. No. _____ entitled IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-11-3), the hard configuration denotes the actual topology, on the transistor and circuit-block level, of the pipeline circuit 80, and the soft configuration denotes the physical parameters (e.g., data width, table size) of the hard-configured components. That is, soft configuration data is similar to the data that can be loaded into a register of a processor (not shown in FIG. 4) to set the operating mode (e.g., burst-memory mode) of the processor. For example, the host processor 42 may send soft-configuration data that causes the configuration manager 90 to set the number and respective priority levels of data and event queues in the communication interface 82. The exception manager 88 may also send soft-configuration data that causes the configuration manager 90 to, e.g., increase the size of an overflowing buffer in the communication interface 82.

[0065] The industry-standard bus interface 91 is a conventional bus-interface circuit that reduces the size and complexity of the communication interface 82 by effectively offloading some of the interface circuitry from the communication interface. Therefore, if one wishes to change the parameters of the pipeline bus 50 or router 61 (FIG. 3), then he need only modify the interface 91 and not the communication interface 82. Alternatively, one may dispose the interface 91 in an IC (not shown) that is external to the pipeline circuit 80. Offloading the interface 91 from the pipeline circuit 80 frees up resources on the pipeline circuit

for use in, e.g., the hardwired pipelines 74₁-74_n and the controller 86. Or, as discussed above, the bus interface 91 may be part of the communication interface 82.

[0066] Still referring to FIG. 4, in addition to the pipeline circuit 80, the pipeline unit 78 of the accelerator 44 includes the data memory 92, and, if the pipeline circuit is a PLIC, a firmware memory 52.

[0067] The data memory 92 buffers data as it flows between another peer, such as the host processor 42 (FIG. 3), and the hardwired pipelines 74₁-74_n, and is also a working memory for the hardwired pipelines. The communication interface 82 interfaces the data memory 92 to the pipeline bus 50 (via the communication bus 94 and industry-standard interface 91 if present), and the communication shell 84 interfaces the data memory to the hardwired pipelines 74₁-74_n.

[0068] The data memory 92 (or other portion of the pipeline unit 78) may also store a profile of the pipeline unit. The profile describes the pipeline unit 78 sufficiently for the host processor 42 (FIG. 3) to appropriately configure itself, the pipeline unit, and other peers of the peer-vector machine 40 (FIG. 3) for intercommunication. For example, the profile may identify the data operations and communication protocols that the pipeline unit 78 is capable of implementing. Consequently, by reading the profile during initialization of the peer-vector machine 40, the host processor 42 can properly configure the message handler 64 (FIG. 3) to communicate with the pipeline unit 78. This technique is analogous to the “plug and play” technique by which a computer can configure itself to communicate with a newly installed peripheral such as a disk drive. Configuration of the host processor 42 and pipeline unit 78 are discussed further in previously cited U.S. patent app. Ser. Nos. _____ entitled IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-11-3) and _____ entitled PROGRAMMABLE CIRCUIT AND RELATED COMPUTING MACHINE AND METHOD (Attorney Docket No. 1934-14-3).

[0069] As discussed above in conjunction with FIG. 3, where the pipeline circuit 80 is a PLIC, the firmware memory 52 stores the firmware that sets the hard configuration of the pipeline circuit. The memory 52 loads the firmware into the pipeline circuit 80 during the configuration of the accelerator 44, and may receive modified firmware from the host processor 42 (FIG. 3) via the communication interface 82 during or after the configuration of the accelerator. The loading and receiving of firmware is further discussed in previously cited U.S. patent app. Ser. Nos. _____ entitled IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-11-3), _____ entitled COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-12-3), and _____ entitled PROGRAMMABLE CIRCUIT AND RELATED COMPUTING MACHINE AND METHOD (Attorney Docket No. 1934-14-3).

[0070] Still referring to FIG. 4, the pipeline unit 78 may include a circuit board or card 98 on which are disposed the pipeline circuit 80, data memory 92, and firmware memory 52. The circuit board 98 may be plugged into a pipeline-bus

connector (not shown) much like a daughter card can be plugged into a slot of a mother board in a personal computer (not shown). Although not shown, the pipeline unit **78** may include conventional ICs and components such as a power regulator and a power sequencer; these ICs/components may also be disposed on the card **98** as is known.

[0071] Further details of the structure and operation of the pipeline unit **78** are discussed in previously cited U.S. patent app. Ser. No. _____ entitled PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-13-3) in.

[0072] FIG. 5 is a block diagram of a pipeline unit **100** of the pipeline accelerator **44** of FIG. 3 according to another embodiment of the invention. The pipeline unit **100** is similar to the pipeline unit **78** of FIG. 4 except that the pipeline unit **100** includes multiple pipeline circuits **80**—here two pipeline circuits **80a** and **80b**. Increasing the number of pipeline circuits **80** typically allows an increase in the number *n* of hardwired pipelines **74₁-74_n**, and thus an increase in the functionality of the pipeline unit **100** as compared to the pipeline unit **78**. Furthermore, the pipeline unit **100** includes a firmware memory **52a** for the pipeline circuit **80a** and a firmware memory **52b** for the pipeline circuit **80b**. Alternatively, the pipeline circuits **80a** and **80b** may share a single firmware memory.

[0073] In the pipeline unit **100**, the services components, i.e., the communication interface **82**, the controller **86**, the exception manager **88**, the configuration manager **90**, and the optional industry-standard bus interface **91**, are disposed on the pipeline circuit **80a**, and the pipelines **74₁-74_n** and the communication shell **84** are disposed on the pipeline circuit **80b**. By locating the services components and the pipelines **74₁-74_n** on separate pipeline circuits **80a** and **80b**, respectively, one can include a higher number *n* of pipelines and/or more complex pipelines than he/she can where the service components and the pipelines are located on the same pipeline circuit. Alternatively, the portion of the communication shell **84** that interfaces the pipelines **74₁-74_n** to the interface **82** and to the controller **86** may be disposed on the pipeline circuit **80a**.

[0074] Further details of the structure and operation of the pipeline unit **100** are discussed in previously cited U.S. patent app. Ser. No. PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-13-3).

[0075] FIG. 6 is a block diagram of the accelerator **44** of FIG. 3 having multiple pipeline units **78** (FIG. 4) or **100** (FIG. 5) according to an embodiment of the invention. For clarity, the accelerator **44** is discussed as having multiple pipeline units **78₁-78_n**, it being understood that the accelerator may include multiple pipeline units **100** or a combination of units **78** and **100**. By including multiple pipeline units **78**, one can increase the functionality and processing power of the accelerator **44** as compared to an accelerator having only one pipeline unit. Furthermore, because each pipeline unit **78** typically has a common industry-standard interface, one can easily modify the accelerator **44** by adding or removing pipeline units.

[0076] In one implementation of the multiple-pipeline accelerator **44**, the industry-standard bus interface **91** is

omitted from each pipeline unit **78₁-78_n**, and a single external (to the pipeline units) interface **91** and communication bus **94** are common to all of the pipeline units. Including a single external bus interface **91** frees up resources on the pipeline circuits **80** (FIG. 4) as discussed above in conjunction with FIG. 4. The pipeline units **78₁-78_n** may all be disposed on a single circuit board (not shown in FIG. 6), each pipeline unit may be disposed on a respective circuit board, or groups of multiple pipeline units may be respectively disposed on multiple circuit boards. In the latter two implementations, the bus interface **91** is disposed on one of the circuit boards. Alternatively, the pipeline units **78₁-78_n** may each include a respective industry-standard bus interface **91** as discussed above in conjunction with FIG. 4, and thus may each communicate directly with the pipeline bus **50** or router **61** (FIG. 3). In this implementation, the pipeline units **78₁-78_n** may be disposed on a single or multiple circuit boards as discussed above.

[0077] Each of the pipeline units **78₁-78_n** is a peer of the host processor **42** (FIG. 3) and of each other. That is, each pipeline unit **78** can communicate directly with any other pipeline unit via the communication bus **94**, and can communicate with the host processor **42** via the communication bus **94**, the bus interface **91**, the router **61** (if present), and the pipeline bus **50**. Alternatively, where the pipeline units **78₁-78_n** each include a respective bus interface **91**, then each pipeline unit can communicate directly with the host processor **42** via the router **61** (if present) and the pipeline bus **50**.

[0078] The operation of the multi-pipeline-unit accelerator **44** is now described by way of two examples.

[0079] In a first example, the pipeline unit **78₁** transfers data to the pipeline **78_n**, which processes the data in a time-critical manner; thus, the pipeline units **78₁** and **78_n** use one or more SYNC signals to synchronize the data transfer and processing. Typically, a SYNC signal is fast enough to trigger a time-critical function, but requires significant hardware resources; comparatively, an event typically is not fast enough to trigger a time-critical function, but requires significantly fewer hardware resources. As discussed in previously cited U.S. patent app. Ser. No. _____ entitled PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-13-3), because a SYNC signal is routed directly from peer to peer, it can trigger a function more quickly than an event, which traverses, e.g., the pipeline bus **50** (FIG. 3), and the communication bus **94**. But because they are separately routed, the SYNC signals require dedicated circuitry, such as routing lines and buffers of the pipeline circuit **80** (FIG. 4). Conversely, because they use the existing data-transfer infrastructure (e.g. the pipeline bus **50** and the communication bus **94**), the events require fewer dedicated hardware resources. Consequently, designers tend to use events to trigger all but the most time-critical functions.

[0080] First, the pipeline unit **78₁** sends data to the pipeline unit **78_n** by driving the data onto the communication bus **94**. Typically, the pipeline unit **78₁** generates a message that includes the data and a header that contains the address of the pipeline unit **78_n**. If the pipeline unit **78₁** were to send the data to multiple pipeline units **78**, then it may do so in one of two ways. Specifically, the pipeline unit **78₁** may sequen-

tially send separate messages to each of the destination pipeline units **78**, each message including a header containing the address of a respective destination unit. Alternatively, the pipeline unit **78₁** may simultaneously send the data to each of the destination pipeline units **78** by including in single message the data and a header containing the addresses of each destination pipeline unit. The sending of data is further discussed in previously cited U.S. patent app. Ser. No. _____ entitled PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-13-3).

[0081] Next, the pipeline unit **78_n** receives the data. Because the pipeline units **78₁-78_n** are each coupled to the common communication bus **94**, each pipeline unit **78₂-78_n** determines whether or not it is an intended recipient of the data. For example, each pipeline unit **78₂-78_n** determines whether its address is included in the header of the message. In this example, the units **78₂-78_{n-1}** determine that they are not intended recipients of the data, and thus ignore the data, i.e., do not load the data into their data memories **92** (FIG. 4). Conversely, the pipeline unit **78_n** determines that it is an intended recipient of the data, and thus loads the data into its data memory **92**. The receiving of data is discussed further in U.S. patent app. Ser. No. _____ entitled PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-13-3).

[0082] Then, when the pipeline unit **78_n** is ready to process the received data, a peer, such as the pipeline unit **78₁**, or an external device (not shown) pulses a SYNC signal to cause the pipeline unit **78_n** to process the data in a timely manner. There exist a number of techniques by which the peer/device that pulses the SYNC signal may determine when the pipeline unit **78_n** is ready to process the received data. For example, the peer/device may merely pulse the SYNC signal a predetermined time after the pipeline unit **78₁** sends the data. Presumably, the predetermined time is long enough to allow the pipeline unit **78_n** to receive and load the data into its data memory **92** (FIG. 4). Alternatively, the pipeline unit **78_n** may pulse a SYNC signal to inform the peer/device that it is ready to process the received data.

[0083] Next, in response to the pulsed SYNC signal, the pipeline unit **78_n** processes the received data. The processing of data by a pipeline unit is discussed further in previously cited U.S. patent app. Ser. No. _____ entitled PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-13-3).

[0084] Then, when the pipeline unit **78_n** is finished processing the data, a peer, an external device (not shown), or the unit **78_n** itself may pulse a SYNC signal to notify the pipeline unit **78₁** to send more data.

[0085] In a second example, the host processor **42** (FIG. 3) transfers data to the pipeline **78_n**, which processes the data in a non-time-critical manner; thus the host processor and the pipeline unit **78_n** use one or more events to synchronize the data transfer and processing for the reasons discussed above.

[0086] First, the host processor **42** (FIG. 3) sends data to the pipeline unit **78_n** by driving the data onto the pipeline bus

50 (FIG. 3). Typically, the host processor **42** generates a message that includes the data and a header containing the address of the pipeline unit **78_n**. If the host processor **42** were to send the data to multiple pipeline units **78**, then it may do so in one of the two ways discussed above in conjunction with the first example.

[0087] Next, the pipeline unit **78_n** receives the data from the pipeline bus **50** (FIG. 3) via the industry-standard bus interface **91** and the communication bus **94**. Because the pipeline units **78₁-78_n** are each coupled to the common communication bus **94**, each pipeline unit determines whether it is an intended recipient of the data in the manner discussed above in conjunction with the first example.

[0088] Then, when the pipeline unit **78_n** is ready to process the received data, a peer, such as the host processor **42** (FIG. 3), or an external device (not shown), generates an event on the pipeline bus **50** or directly on the communication bus **94** to cause the pipeline unit **78_n** to process the data in a timely manner. There exist a number of techniques by which the peer/device that generates the event may determine when the pipeline unit **78_n** is ready to process the received data. For example, the peer/device may merely generate the event a predetermined time after the host processor **42** sends the data. Presumably, the predetermined time is long enough to allow the pipeline unit **78_n** to receive and load the data into its data memory **92** (FIG. 4). Alternatively, the pipeline unit **78_n** may generate an event to inform the peer/device that it is ready to process the received data.

[0089] Next, the pipeline unit **78_n** receives the event. The receiving of an event is discussed in previously cited U.S. patent app. Ser. No. _____ entitled PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-13-3).

[0090] Then, in response to the received event, the pipeline unit **78_n** processes the received data. The processing of data by a pipeline unit **78_n** is discussed further in previously cited U.S. patent application Ser. No. _____ entitled PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-13-3).

[0091] Next, when the pipeline unit **78_n** is finished processing the data, a peer, an external device (not shown), or the unit **78_n** itself may generate an event to notify the host processor **42** (FIG. 3) to send more data.

[0092] Still referring to FIG. 6, alternative implementations of the accelerator **44** are contemplated. For example, although the above-discussed first and second examples of operation respectively use SYNC signals and events exclusively, it is contemplated that the accelerator **44** can use both SYNC signals and events in combination. Furthermore, other peers can use one or more of the multiple pipeline units **78** or **100** merely for bulk storage of data in their respective data memories **92**. In addition, a designer may replace the host processor **42** (FIG. 3) with one or more of the pipeline units **78** or **100**, which together form a "host" peer that performs the functions of the host processor. Moreover, one or more of the pipeline units **78** or **100** may act as one or more message-distribution peers. For example, suppose that the host processor **42** generates a message for transmission to multiple subscriber peers. The host processor **42** may send

the message to a message-distribution peer, which then distributes the message to each of the subscriber peers. Consequently, the message-distribution peer, not the host processor 42, undertakes the burden of distributing the message, and thus allows the host processor to devote more time and resources to other tasks.

[0093] FIG. 7 is a block diagram of the accelerator 44 (FIG. 3) having multiple pipeline units 78 (FIG. 4) or 100 (FIG. 5) according to another embodiment of the invention.

[0094] The accelerator 44 of FIG. 7 is the same as the accelerator 44 of FIG. 6, except that the accelerator 44 of FIG. 7 includes a communication-bus router 110 for routing data between the pipeline units 78₁-78_n and other peers, such as the host processor 42 (FIG. 3), and devices (not shown) that are coupled to the pipeline bus 50 (FIG. 3). For clarity, the accelerator 44 of FIG. 7 is discussed as having multiple pipeline units 78₁-78_n, it being understood that the accelerator may include multiple pipeline units 100 or a combination of units 78 and 100.

[0095] The communication-bus router 110 is coupled to the pipeline units 78₁-78_n via respective branches 94₁-94_n of the communication bus 94, and is coupled to the industry-standard bus interface 91 (if present) via a bus 112. Alternatively, as discussed above in conjunction with FIG. 6, each pipeline unit 78₁-78_n may include a respective interface 91 on board, and thus the external interface 91 can be omitted such that the router 110 is directly coupled to the pipeline bus 50 (or router 61 if present) of FIG. 3.

[0096] The router 110 routes signals from the pipeline bus 50 (FIG. 3) to the respective destination pipeline unit or units 78₁-78_n, and also routes signals from a source pipeline unit to one or more destination pipeline units or to the pipeline bus. Consequently, the router 110 offloads from each of the pipeline units 78₁-78_n the function of determining whether a signal on the communication bus 94 is intended for that pipeline unit. This offloading may free up resources on the pipeline circuit 80 (FIG. 4) of each pipeline unit 78₁-78_n, and thus may allow an increase in the functionality of each pipeline unit.

[0097] Still referring to FIG. 7, the operation of the multi-pipeline-unit accelerator 44 with the router 110 is now described. Because the operation is similar to that described above for the accelerator 44 of FIG. 6, the below description highlights the operational differences between the accelerators of FIGS. 6 and 7.

[0098] In a first example, the pipeline unit 78₁ transfers data to the pipeline 78_n, which processes the data in a time-critical manner; thus, the pipeline units 78₁ and 78_n use one or more SYNC signals to synchronize the data transfer and processing as discussed above in conjunction with the first example of FIG. 6.

[0099] First, the pipeline unit 78₁ sends data to the pipeline unit 78_n by driving the data onto the branch 94₁ of the communication bus. Typically, the pipeline unit 78₁ generates a message that includes the data and a header that contains the address of the pipeline unit 78_n.

[0100] Next, the router 110 receives the data, determines that the destination of the data is the pipeline unit 78_n, and drives the data onto the branch 94_n of the communication bus. Typically, the router 110 determines the destination of

the data by analyzing the header of the message containing the data and extracting the destination address from the header. Consequently, because the router 110 determines the proper destination of the data, the pipeline unit 78_n can merely accept the data from the router without determining whether it is an intended recipient of the data. Alternatively, the pipeline 78_n may determine whether it is an intended recipient of the data, and generate an exception (discussed in previously cited U.S. patent app. Ser. Nos. _____ entitled IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-11-3), _____ entitled COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-12-3), and _____ entitled PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-13-3) if it is not an intended recipient. The pipeline unit 78_n can send this exception to the host processor 42 (FIG. 3) via the router 110, the industry-standard bus interface 91 (if present), the router 61 (if present), and the pipeline bus 50 (FIG. 3).

[0101] Then, the pipeline unit 78_n loads the data from the bus branch 94_n. The loading of data by a pipeline unit is further discussed in previously cited U.S. patent app. Ser. No. _____ entitled PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-13-3).

[0102] Next, when the pipeline unit 78_n is ready to process the received data, a peer, such as the pipeline unit 78₁, or an external device (not shown) pulses a SYNC signal to cause the pipeline unit 78_n to process the data in a timely manner as discussed above in conjunction with the first example of FIG. 6.

[0103] Then, in response to the pulsed SYNC signal, the pipeline unit 78_n processes the received data as discussed above in conjunction with the first example of FIG. 6.

[0104] Next, when the pipeline unit 78_n is finished processing the data a peer, an external device (not shown), or the unit 78_n itself may pulse a SYNC signal to notify the pipeline unit 78₁ to send more data.

[0105] In a second example, the host processor 42 (FIG. 3) transfers data to the pipeline 78_n, which processes the data in a non-time-critical manner; thus the host processor and the pipeline unit 78_n use one or more events to synchronize the data transfer and processing for the reasons discussed above in conjunction with FIG. 6.

[0106] First, the host processor 42 (FIG. 3) sends data to the pipeline unit 78_n by driving the data onto the pipeline bus 50 (FIG. 3). Typically, the host processor 42 generates a message that includes the data and a header containing the address of the pipeline unit 78_n.

[0107] Next, the router 110 receives the data from the pipeline bus 50 (FIG. 3) via the industry-standard bus interface 91 (if present) and the bus 112.

[0108] Then, the router 110 determines that the destination of the data is the pipeline unit 78_n, and drives the data onto the branch 94_n of the communication bus. Typically, the router 110 determines the destination of the header as

discussed above in conjunction with the first example of FIG. 7. Consequently, because the router 110 determines the proper destination of the data, the pipeline unit 78_n can merely accept the data from the router without determining whether it is an intended recipient of the data. Alternatively, the pipeline 78_n may determine whether it is an intended recipient of the data, generate an exception (discussed in previously cited U.S. patent app. Ser. Nos. _____ entitled IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-11-3), _____ entitled COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-12-3), and _____ entitled PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-13-3) if it is not an intended recipient, and send the exception to the host processor 42 (FIG. 3) as discussed above in conjunction with the second example of FIG. 6.

[0109] Next, the pipeline unit 78_n loads the data from the bus branch 94_n. The loading of data by a pipeline unit is further discussed in previously cited U.S. patent app. Ser. No. _____ entitled PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-13-3).

[0110] Then, when the pipeline unit 78_n is ready to process the received data, a peer, such as the host processor 42 (FIG. 3), or an external device (not shown), generates an event on the pipeline bus 50 or on one of the branches 94₁-94_{n-1} of the communication bus to cause the unit 78_n to process the data in a timely manner as discussed above in conjunction with the second example of FIG. 6.

[0111] Next, the router 110 receives the event, determines that it is intended for the pipeline unit 78_n, and drives the event onto the bus branch 94_n.

[0112] Then, the pipeline unit 78_n loads the event from the bus branch 94. The loading of an event by a pipeline unit 78 is discussed in previously cited U.S. patent app. Ser. No. _____ entitled PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD (Attorney Docket No. 1934-13-3).

[0113] Next, in response to the received event, the pipeline unit 78_n processes the received data.

[0114] Then, when the pipeline unit 78_n is finished processing the data, a peer, an external device (not shown), or the unit 78_n itself may generate an event to notify the host processor 42 (FIG. 3) to send more data.

[0115] Still referring to FIG. 7, although the first and second examples of operation respectively use SYNC signals and events exclusively, it is contemplated that the accelerator 44 can use a combination of both SYNC signals and events.

[0116] FIG. 8 is a block diagram of the accelerator 44 of FIG. 3 including multiple groups 120 of multiple pipeline units 78 (FIG. 4) or 100 (FIG. 5) according to an embodiment of the invention. Including multiple groups 120 of pipeline units increases the functionality of the accelerator

44, and allows a designer to increase the accelerator's efficiency by grouping pipeline units that perform related operations. For clarity, the accelerator 44 of FIG. 8 is discussed as having multiple pipeline units 78, it being understood that the accelerator may include multiple pipeline units 100 or a combination of units 78 and 100. Furthermore, the pipeline units 78 do not include industry-standard bus interfaces 91 (this interface is external in this embodiment), although they may in another embodiment.

[0117] The accelerator 44 includes six groups 120₁-120₆ of pipeline units 78, each group having three pipeline units and a respective intra-group communication-bus router 110₁-110₆ that interconnects the pipeline units to each other and to the other pipeline-unit groups. Although the accelerator 44 is discussed as including six groups 120₁-120₆ of three pipeline units 78 each, other implementations of the accelerator may include virtually any number of groups of any number pipeline units, and not all the groups need have the same number of pipeline units. Furthermore, the communication-bus routers 110₁-110₆ may be omitted as discussed above in conjunction with the accelerator 44 of FIG. 6.

[0118] The pipeline-unit group 120₁ includes three pipeline units 78₁-78₃, which are connected to an intra-group communication-bus router 110₁ via branches 94₁, 94₂, and 94₃, respectively, of a communication bus in a manner similar to that discussed above in conjunction with FIG. 7. The other groups 120₂-120₆ are similar.

[0119] The communication-bus routers 110₁-110₃ of the groups 120₁-120₃ are connected to a first-level router 122₁ via respective branches 124₁-124₃ of a first-level bus 126₁. The router 122₁ and bus 126₁ allow the pipeline units 78₁-78₉ to communicate with each other.

[0120] Similarly, the communication-bus routers 110₄-110₆ are connected to a first-level router 122₂ via respective branches 128₁-128₃ of a first-level bus 126₂. The router 122₂ and bus 126₂ allow the pipeline units 78₁₀-78₁₈ to communicate with each other.

[0121] The first-level routers 122₁ and 122₂ are connected to a second-level router 130 via respective branches 132₁-132₂ of a second-level bus 134. The router 130 and bus 134 allow the pipeline units 78₁-78₁₈ to communicate with each other and with other peers/devices as discussed below.

[0122] The pipeline bus 50 and a secondary pipeline bus 136 are coupled to the second-level router 130 via the respective industry-standard bus interfaces 91₁ and 91₂. The secondary pipeline bus 136 may be connected to peers, such as the host processor 42 (FIG. 3), or peripherals, such as a hard-disk drive (not shown), that are not coupled to the pipeline bus 50. Furthermore, either or both of the busses 50 and 136 may be coupled to peers or peripherals via a network or the internet (neither shown) such that the accelerator 44 can be remotely located from other peers, such as the host processor 42 (FIG. 3).

[0123] A bus 138 directly connects one or more SYNC signals to all of the pipeline units 78₁-78₁₈ and to other peers, such as the host processor 42 (FIG. 3), or devices (not shown).

[0124] Still referring to FIG. 8, in one example of operation, each group 120₁-120₆ of pipeline units 78 processes data from a respective sensor of a sonar array (not shown)

that is coupled to the secondary pipeline bus 136. Because the pipeline units 78₁-78₃ of the group 120₁ are interconnected by a single router 110₁, these pipeline units can communicate with each other more quickly than they can with the pipeline units 78₄-78₁₈ of the other groups 120₂-120₆. This higher communication speed is also present in each of the other groups 120₂-120₆. Consequently, a designer can increase the processing speed of the accelerator 44 by grouping together pipeline units that frequently transfer data or otherwise communicate among themselves.

[0125] In general, the pipeline units 78₁-78₁₈ communicate with each other and with peers, such as the host processor 42 (FIG. 3), and devices coupled to the buses 50 and 136 in a manner similar to that discussed above in conjunction with FIG. 7. For example, a sensor (not shown) coupled to the bus 136 communicates with the pipeline unit 78₁ via the industry-standard bus interface 91₁, the second-level router 130, the first level router 122₁, and the intra-group router 110₁. Similarly, the pipeline unit 78₁ communicates with the pipeline unit 78₇ via the routers 110₁, 122₁, and 110₃, and communicates with the pipeline unit 78₁₀ via the routers 110₁, 122₁, 130, 122₂, and 110₄.

[0126] The preceding discussion is presented to enable a person skilled in the art to make and use the invention. Various modifications to the embodiments will be readily apparent to those skilled in the art, and the generic principles herein may be applied to other embodiments and applications without departing from the spirit and scope of the present invention. Thus, the present invention is not intended to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles and features disclosed herein.

What is claimed is:

1. A pipeline accelerator, comprising:
 - a communication bus; and
 - a plurality of pipeline units each coupled to the communication bus and each comprising a respective hardwired-pipeline circuit.
2. The pipeline accelerator of claim 1 wherein each of the pipeline units comprises:
 - a respective memory coupled to the hardwired-pipeline circuit; and
 - wherein the hardwired-pipeline circuit is operable to,
 - receive data from the communication bus,
 - load the data into the memory,
 - retrieve the data from the memory,
 - process the retrieved data, and
 - drive the processed data onto the communication bus.
3. The pipeline accelerator of claim 1 wherein each of the pipeline units comprises:
 - a respective memory coupled to the hardwired-pipeline circuit; and
 - wherein the hardwired-pipeline circuit is operable to,
 - receive data from the communication bus,
 - process the data,
 - load the processed data into the memory,

- retrieve the processed data from the memory, and
 - load the retrieved data onto the communication bus.
4. The pipeline accelerator of claim 1 wherein each of the hardwired-pipeline circuits is disposed on a respective field-programmable gate array.
 5. The pipeline accelerator of claim 1, further comprising:
 - a pipeline bus; and
 - a pipeline-bus interface coupled to the communication bus and to the pipeline bus.
 6. The pipeline accelerator of claim 1, further comprising:
 - wherein the communication bus comprises a plurality of branches, a respective branch coupled to each pipeline unit; and
 - a router coupled to each of the branches.
 7. The pipeline accelerator of claim 1, further comprising:
 - wherein the communication bus comprises a plurality of branches, a respective branch coupled to each pipeline unit;
 - a router coupled to each of the branches;
 - a pipeline bus; and
 - a pipeline-bus interface coupled to the router and to the pipeline bus.
 8. The pipeline accelerator of claim 1, further comprising:
 - wherein the communication bus comprises a plurality of branches, a respective branch coupled to each pipeline unit;
 - a router coupled to each of the branches;
 - a pipeline bus;
 - a pipeline-bus interface coupled to the router and to the pipeline bus; and
 - a secondary bus coupled to the router.
 9. The pipeline accelerator of claim 1 wherein:
 - the communication bus is operable to receive data addressed to one of the pipeline units; and
 - the one pipeline circuit is operable to accept the data; and
 - the other pipeline circuits are operable to reject the data.
 10. The pipeline accelerator of claim 1, further comprising:
 - wherein the communication bus comprises a plurality of branches, a respective branch coupled to each pipeline unit;
 - a router coupled to each of the branches and operable to,
 - receive data addressed to one of the pipeline units, and
 - provide the data to the one pipeline unit via the respective branch of the communication bus.
 11. A computing machine, comprising:
 - a processor;
 - a pipeline bus coupled to the processor; and

a pipeline accelerator comprising,
 a communication bus,
 a pipeline-bus interface coupled between the pipeline bus and the communication bus, and
 a plurality of pipeline units each coupled to the communication bus and each comprising a respective hardwired-pipeline circuit.

12. The computing machine of claim **11** wherein:

the processor is operable to generate a message that identifies one of the pipeline units and to drive the message onto the pipeline bus;
 the pipeline-bus interface is operable to couple the message to the communication bus;
 the pipeline units are each operable to analyze the message;
 the identified pipeline unit is operable to accept the message; and
 the other pipeline circuits are operable to reject the message.

13. The computing machine of claim **11**, further comprising:

wherein the communication bus comprises a plurality of branches, a respective branch coupled to each pipeline unit;
 wherein the processor is operable to generate a message that identifies one of the pipeline units and to drive the message onto the pipeline bus; and
 a router coupled to each of the branches and to the pipeline-bus interface and operable to receive the message from the pipeline-bus interface and to provide the message to the identified pipeline unit.

14. The computing machine of claim **11**, further comprising:

wherein the communication bus comprises a plurality of branches, a respective branch coupled to each pipeline unit;
 a secondary bus; and
 a router coupled to each of the branches, to the pipeline-bus interface, and to the secondary bus.

15. A method, comprising:

sending data to first of a plurality of pipeline units via a communication bus, each pipeline unit including a respective hardwired pipeline; and
 processing the data with the first pipeline unit.

16. The method of claim **15** wherein sending the data comprises:

sending the data to a router; and
 providing the data to the first pipeline unit with the router via a respective first branch of the communication bus.

17. The method of claim **15** wherein sending the data comprises sending the data to the first pipeline unit with a processor.

18. The method of claim **15** wherein sending the data comprises sending the data to the first pipeline with a second of the plurality of pipeline units.

19. The method of claim **15**, further comprising driving the processed data onto the communication bus with the first pipeline unit.

20. The method of claim **15** wherein processing the data with the first pipeline unit comprises:

receiving the data from the communication bus with a hardwired-pipeline circuit,
 loading the data into a memory with the hardwired-pipeline circuit,
 retrieving the data from the memory with the hardwired-pipeline circuit, and

processing the retrieved data with the hardwired-pipeline circuit.

21. The method of claim **15**, further comprising:

wherein processing the data with the first pipeline unit comprises,
 receiving the data from the communication bus with a hardwired-pipeline circuit,
 processing the received data with the hardwired-pipeline circuit, and
 loading the processed data into a memory with the hardwired-pipeline circuit; and

retrieving the processed data from the memory and driving the processed data onto the communication bus with the hardwired-pipeline circuit.

22. The method of claim **15**, further comprising:

generating a message that includes the data and that identifies the first pipeline unit as a recipient of the message; and

wherein sending the data to the first pipeline unit comprises determining from the message that the first pipeline is a recipient of the message.

* * * * *