(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number : **0 685 799 A1**

(12) EUROPEAN PATENT APPLICATION

(21) Application number : **95303697.7**

(51) Int. Cl.[6] : **G06F 13/40,** G06F 13/38

(22) Date of filing : **31.05.95**

(30) Priority : **03.06.94 US 253530**

(43) Date of publication of application :
06.12.95 Bulletin 95/49

(84) Designated Contracting States :
DE FR GB

(71) Applicant : **SYMBIOS LOGIC INC.**
2001 Danfield Court
Fort Collins, Colorado 80525 (US)

(72) Inventor : **Avery, James M.**
5231 McMurray Drive
Fort Collins, Colorado 80525 (US)
Inventor : **Isenberg, William D.**
1937 Sandalwood Lane
Fort Collins, Colorado 80526 (US)

(74) Representative : **Gill, David Alan**
W.H. Beck, Greener & Co.,
7 Stone Buildings,
Lincoln's Inn
London WC2A 3SZ (GB)

(54) Multi-device connector.

(57) The present invention provides for a multi-device connector (16) which allows a device (14) to be connected to a bus (12), such as a system bus in a computer. The invention can comprise a device adapter which interfaces the device (14) to the bus (12), so as to represent a customized device adapter (16) and in particular, the invention provides for a multi-device connector (16) comprising a bus interface for a computer comprising an adapter means for adapting a plurality of devices (14) to a computer bus (12), characterised by buffer means (22) for each device (14), for storing data, a data manager (20) for each buffer means (22), for controlling data transfers between said buffer means (22) and said bus (12), and control means (24) for each buffer means (22), for controlling data transfers between said buffer means (22) and a respective device (14) common design macro, which is programmable, a user can easily specify and generate custom device adapters for a plurality of dissimilar devices to be connected to the bus. A resulting adapter architecture allows for multiple, dissimilar devices to interface to a computer bus with a single device adapter integrated circuit or card.

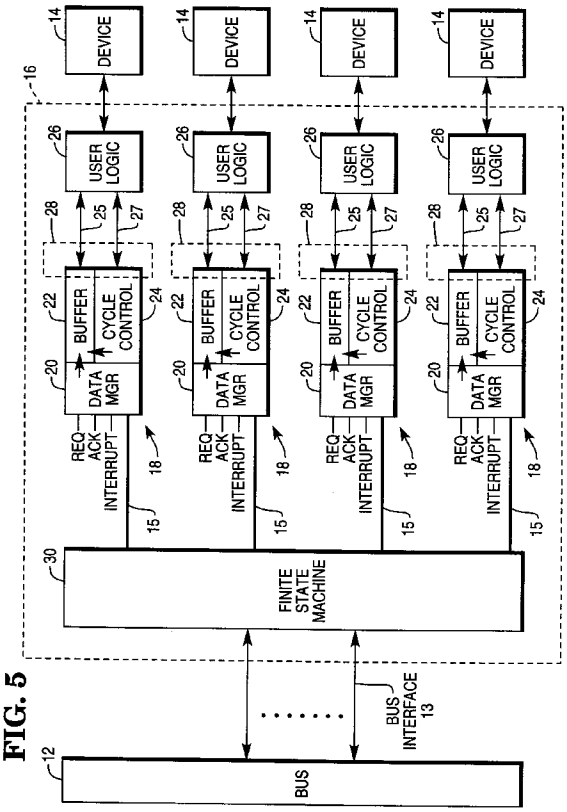

FIG. 5

EP 0 685 799 A1

The present invention relates to a multi-device connector apparatus and in particular, but not exclusively, to a configurable device adapter for connecting dissimilar types of devices to a data processing system bus.

This discussion will first explain a highly simplified data bus, and then illustrate how the simplified bus has evolved into a vast array of different types of data busses.

It is known that different types of devices can generally be connected to a bus by different types of device adapters. However, device adapters do not necessarily result in optimal matching to a particular device, as will be explained later with particular reference to Fig. 1 below.

The particular data-handling requirements of different devices, and the manner in which uncorrupted data transfer can occur from a first device which is ready to send data, and which knows that a second device is ready to receive that data, place particular requirements on device adapters which are disadvantageously not readily met.

It is apparent that there are numerous design constraints which exist and trade-offs which are made when attaching a device to a bus. When designing an adapter to allow such attachment, i.e. a device adapter, these design constraints and trade-offs result in a time consuming design effort for a particular type or class of device(s). When a subsequent type or class of device(s) is to attach to this bus, a similar intensive design effort is required to create a device adapter for this new device. There is a need for a system which allows for parameters, such as buffering specifications for a particular type or class of device, to be identified to a general purpose design macro, wherein a device adapter design is automatically generated which conforms to such user specified requirements. This device adapter design would thus be optimized for a given device which is to attach to a bus.

The invention seeks to provide for a multi-device connector having advantages over known such connectors.

According to one aspect of the present invention there is provided a multi-device connector comprising a bus interface for a computer for adapting a plurality of devices to a computer bus, characterised by buffer means for each device for storing data, a data manager for each buffer means for controlling data transfers between said buffer means and said bus, and control means for each buffer means, for controlling data transfers between said buffer means and a respective device.

According to another aspect of the present invention a multi-device connector comprising an adapter card for communication with an expansion slot of a computer which connects to a bus in the computer, characterised by a plurality of buffer means, for storing data for a respective device, and means for multiplexing said plurality of buffer means with said bus.

The invention can also provide for a multi-device connector for connecting multiple devices to a bus comprising back end interface means, having different characteristics than those of said bus, and interface logic means for connecting the back end interface to the bus, said interface logic means comprising a plurality of device sub-adapters which couple the back end interface means to the bus, at least one of said plurality of device sub-adapters comprising one or more buffers, a data manager for controlling data transfers between the bus and the one or more buffers, and a controller for controlling data transfers between the back end interface and the one or more buffers.

Further, the invention can provide for a multi-device connector comprising an electronic card for use in a computer with a bus characterised by a plurality of device sub-adapters, wherein at least one sub-adapter comprises a buffer, a data manager for transferring data from the bus into the buffer, and a controller for transferring data from the respective buffer to a back end interface, and by an arbitration circuit which allows only a single data manager to communicate with the bus at a given time.

The invention is advantageous in providing for an improved device adapter.

Further the invention can also provide a device adapter which can be optimized, or tuned, to a given environment.

The invention can advantageously optimally match a bus with a plurality of different devices.

Also, the present invention can advantageously provide a general purpose device adapter macro which can be customized, via user specified parameters, for a particular attaching device.

Preferably, in one form of the invention, configurable device adapters are provided. The invention provides circuitry, for a particular type or class of device, which adapts the device to a bus. The individual circuitry for each device can be optimized, or "tuned," within certain parameter limits. A general purpose macro is used in conjunction with user specified parameters for a particular device, to generate a customized, or tuned, device adapter design based on such user specified parameters. Accordingly, a large degree of specific design information that would otherwise have to be manually provided by a user is automatically generated by use of the general purpose macro. The macro embodies a high degree of configuration to meet a wide variety of device support requirements. The user can thus configure the macro in such a way as to be tuned to meet particular system performance requirements.

The tunable architecture provides a basic architectural framework with contractual, or design-rules based, interfaces. By using contractual interfaces, the internals of a specific block can be configured without changing the interface design rules. With this concept, multiple behaviours and sub-block architectures can be implemented without changing the basic configurable macro concept.

The invention is described further hereinafter, by way of example only, with reference to the accompanying drawings in which :

Fig. 1 illustrates a simplified bus for data transmission;

Fig. 2 illustrates a simple way of transferring data from a 32-bit data bus to a receiver having an 8-bit data interface;

Fig. 3 illustrates how the approach of Fig. 2 can be improved by the use of a buffer;

Fig. 4 is an overview of one form of the invention;

Fig. 5 is a more detailed overview of one form of the invention;

Fig. 6 shows internal interfaces having separated data and control;

Fig. 7 shows a buffer configured as unidirection read/write;

Fig. 8 shows buffers configured as separate read and write buffers;

Fig. 9 shows multiple buffers configured as ping-pong or circular buffers;

Fig. 10 illustrates a system board, for a computer, which contains the device adapter apparatus of Fig. 5;

Fig. 11 illustrates an expansion card, for a computer, which contains the device adapter apparatus of Fig. 5; and

Fig. 12 is a process flow diagram for the design and manufacture of integrated circuits using the tunable macro.

Fig. 1 illustrates a highly simplified example of data transfer over a bus. The transfer sequence is the following, beginning at the top left part.

1. The Sender places data onto the Data Lines, as indicated.
2. The Sender pulls a Ready line HIGH, as indicated, thereby telling the Receiver that the Data is ready.
3. In response, the Receiver collects the Data.
4. The Receiver pulls the Acknowledge line HIGH, thereby telling the Sender that the Receiver has taken the data, and that another piece of Data can be placed on the Data Lines.
5. The Sender detects the signal on the Acknowledge line, and pulls the Ready line LOW.
6. The Receiver responds by removing the signal on the Acknowledge line.

At this time, all lines are in their original conditions. The Sender may repeat the process by placing a second piece of Data onto the Data Lines.

The exemplary transaction given above is a highly simplified example. Several examples of ways to improve and complicate the bus are outlined in the following:

1. The Receiver may detect an error in the data, and may wish to request that a given piece of data be transmitted again. To accommodate this wish, an additional Repeat line can be provided, by which the Receiver requests the repeated transmission.

   Thus, when the Sender receives a signal on the Repeat line, the Sender repeats the transmission. Conversely, if the Sender receives no such signal, but receives an acknowledge signal instead, the Sender sends the next piece of data.

2. Assume that the Sender places a piece of data on the Data Lines, and pulls the Ready line HIGH. How long should it wait for the acknowledge signal? What does the Sender do if it receives no acknowledgment signal within the allotted time?

   One common approach to the problem is the following. If the Sender receives no acknowledgment signal within a specified time, then the Sender abandons the attempt to send data.

   However, another approach is based on the supposition that, if the Receiver is busy performing another task when the Sender wants to send data, nevertheless, the Receiver is willing to accept the data when the task terminates. Under this approach, a Wait line can be provided. When the Sender sees a Wait signal, it does not abandon the attempt to send data, but waits for the Wait signal to disappear.

3. Suppose that other devices besides the Receiver are connected to the data bus. However, the Sender wishes to send data to a specific device, and wishes others to ignore the data. Selection lines can be added which instruct a specific device to listen to the data.

4. As in case 3, above, multiple devices are connected to the data bus. However, in contrast to the examples given above, now the devices wish to initiate communication: The devices now become Senders. Further, the devices simultaneously wish to send data. However, to allow them to do so would result in confusion. A solution is to add Interrupt Lines. Each device asks permission, prior to sending data, by using the interrupt lines.

There are numerous additional examples of additional types of lines which can be added to a bus, to pro-

vide added features. Therefore, it is not surprising that numerous types of device interfaces exist, each with its own particular combination of lines and data transfer sequences.

Further, a device interface to a bus does not operate in isolation: if the device and bus attain the highest possible speed between themselves, it is possible that this speed was obtained at a cost to some other components or devices.

For example, assume that a bus couples together a processor and a printer. It is easy to see that very fast data transmission can occur if the processor devotes its full attention to the printer, and ignores all other devices on the bus, such as disc drives. However, under this scenario, the processor becomes unavailable for other tasks while tending the printer. Thus, while throughput of printed data is very high, other tasks have been brought to a standstill.

Viewed another way, in this example, the processor is capable of transmitting data at a far higher rate than the printer can print it. Thus, in fact, during the full-attention printing, the processor will be spending idle time while the printer prints. The processor could be performing other tasks during this idle time.

One solution to this type of problem is the use of buffering. A simple example will illustrate.

Assume that the Sender's data bus is 32 bits wide, as shown in Fig. 2. Assume that the Receiver's data bus is smaller, at 8 bits. A very crude device adapter could operate as STEP 1 through STEP 4 indicate. The Adapter transfers data in eight-bit chunks. However, a problem with this approach is immediately apparent. While the Receiver may be receiving data as fast it can, the Sender is tied to the bus during the Receiver's activity. The Sender can do nothing else. This approach is inefficient from the Sender's view.

As shown in Fig. 3, the Adapter can grab the entire 32-bit word from the Sender, and place the data in a Buffer. Now, the Buffer holds the data while the Adapter transmits it to the Receiver in eight-bit chunks. Now the Sender has been freed from waiting for the Receiver to receive the data.

For sending data in the opposite direction, wherein the Receiver sends data to the Sender, the Adapter would perform the opposite sequence. The Adapter would collect data from the Receiver in eight-bit chunks. When 32 bits have accumulated in the Buffer, the Adapter sends the 32 bits to the Sender.

While the buffering approach represents an improvement over the non-buffering approach, the buffering approach itself can be improved, by choosing the proper size of the buffers under the circumstances, as well as other parameters.

The invention includes a method for dynamically configuring a device adapter for interconnecting devices to a common bus. Figs. 4 and 5 are views of the resulting apparatus for such connection. Several features are significant.

1. A bus 12 is shown. Typically, the bus is a system bus in a micro-computer. Devices 14 do not directly communicate with the bus. Instead, the devices 14 communicate with a device adapter 16. The invention provides a method and apparatus for generating a device adapter 16 which allows a given device 14 and the bus 12 to communicate with each other. A configurable macro can be produced that has the flexibility of generating a device adapter that supports multiple devices all of the same type or class, or that supports multiple, dissimilar types of devices. Alternatively, and as a further exemplification of the flexibility of the present invention, the macro can be configured to provide separate device adapters for each interconnecting device (whether similar or dissimilar).

2. These devices 14 can include normal peripheral devices, such as disc drives and printers, which are normally connected to or within a computer. In addition, these devices can include other apparatus, such as the following:

-- video adapters,
-- keyboards and pointing devices,
-- multimedia adapters,
-- memory controllers,
-- memory and I/O cache,
-- communication interfaces, and
-- interfaces to other busses.

In principle, there is no fundamental limit to the types of apparatus which can act as a device 14 in Fig. 4.

3. In one embodiment, the generated device adapter presents a single load to the bus. One ramification of the single load concept is that the bus treats the device adapter, in general, as a single device. One exception is addressing: to select one of the four devices shown, the processor (not shown, but which is also connected to the bus) must supply an address to the device adapter which, in turn, selects the corresponding device. If the device adapter were, in all respects, viewed as a single device by the processor, then the device adapter would only possess a single address.

As a single load, the device adapter simplifies some overhead which the bus would otherwise incur.

4. The generated device adapter is "tunable". Further, the tuning is done for each type or class of device.

In the preferred embodiment, within the device adapter, there are buffers (BUF) devoted to each device, as indicated in Fig. 5. Because, in general, each device has different buffering needs, during design the device adapter is optimized for the best, or at least a good, buffer design for each device.

Further, because of particular characteristics of the device adapter, to be discussed later, certain features of the buffers can be changed, without requiring alteration of other circuitry within the device adapter.

5. Data and control are separated within the macro. Interface control mechanisms are specified independently from the data paths in the macro. By specifying the data paths independently, the control signal interfaces between functional blocks can operate on a variety of data interchanges without special control structures. As an example, the transfer of data to/from an interface can be controlled in a similar fashion regardless of the length of the data transfer. The control interfaces operating in contractual agreement provide the mechanisms to initiate, terminate and monitor the data interchange without regard to the data contents.

A computer listing for the software (i.e. macro) used in creating and generating a configurable device adapter can be achieved. The macro requires a VHDL compiler. VHDL compilers are commercially available. VHDL is an acronym referring to VHSIC Hardware Description Language. VHSIC is an acronym referring to Very High Speed Integrated Circuit.

A commercially available VHDL compiler known as V-System VHDL Simulator and Compiler Platform Revision 2.5, available from Model Technology, located Beaverton, Oregon can be used for this purpose.

To design a customized device adapter, the macro is used in two different ways, namely, (1) in synthesizing a gate-level logic diagram, and (2) in simulating the circuit.

Synthesis

To perform synthesis, a user does the following:

1. The user specifies the following parameters:
   -- The width of the device interface (e.g., 8, 16, or 32 bits).
   -- The address size (i.e. number of bits) of the device interface.
   -- The depth of the buffer contained in the device adapter (number of bits). The buffers are explained in greater detail later.
   -- Buffer packing.

These parameters are specified by inserting them into the file of the macro labeled PCIMACRO_USER_PARAMS.VHD.

2. The user can compile the design, using a commercially available synthesis engine such as Design Analyzer 3.0c, available from Synopsis Incorporated, Mountain View, California. For compilation, the user should load the macro files, described later in Table 2 in the following order:
pcimacro_user_params.vhd, macro_pkg.vhd, configurable_bit_reg.vhd, configurable_dword_reg.vhd, clktree.vhd, signaltree.vhd, burst_size_cntr.vhd, next_addr_cmpr.vhd, new_transfer_counter.vhd, control_reg.vhd, dm_bk_sm.vhd, dm_fe_sm.vhd, data_manager.vhd, address_counter.vhd, devsel_timer.vhd, master_latency_timer.vhd, parity36.vhd, target_latency_timer.vhd, master_state_machine.vhd, slave_state_machine.vhd, pci_fsm.vhd, cycler.vhd, cc_bk_sm.vhd, cc_bypass_sm.vhd, cc_fe_sm.vhd, cycle_controller.vhd, dpr_gen.vhd, pipeline_block.vhd, valid_data_counter.vhd, address_pointer.vhd, buffer_element.vhd, memory_element.vhd, output_latch.vhd, new_fifo.vhd, fifo.vhd, byte_steering_logic.vhd, bist_reg.vhd, class_code_reg.vhd, command_reg.vhd, device_id_reg.vhd, header_reg.vhd, interrupt_line_reg.vhd, interrupt_pin_reg.vhd, max_lat_reg.vhd, min_gnt_reg.vhd, revision_id_reg.vhd, status_reg.vhd, vendor_id_reg,vhd, high_bit_reg.vhd, base_address_reg.vhd, rom_base_address_reg.vhd, miscelaneous_configuration_regs.vhd, config.vhd, bk_end.vhd, pci_macro.vhd, nan2.vhd, dffpq.vhd, dffrpq.vhd, or2.vhd, hbuf.vhd, sbuf.vhd, inbuf.vhd, inpd.vhd, inv.vhd, iobuf.vhd, iobufpci.vhd, iopdl6.vhd, iopd12s1.vhd, iobuf_iopd12s1.vhd, opdl6.vhd, otpdl6.vhd, bus_hbuf.vhd, bus_inbuf.vhd, iopdpci.vhd, bus_inv.vhd, bus_opd16.vhd, iobuf_iopd16.vhd, configured_pci_macro.vhd

The product of the synthesis is a gate level diagram or netlist. A netlist is a description of how gate cells are connected together. An exemplary netlist is the following:

```
AND1 2 4 10
OR1 10 12 14
```

This list indicates that AND gate 1 has two inputs. One is connected to node 2, and the other is connected to node 4. The output is connected to node 10. The list indicates that OR gate 1 has two inputs. One is connected to node 10 (which is the output of the AND gate) and the other is connected to node 12. The output is connected to node 14.

Therefore, the synthesis engine, together with the macro and specified parameters, synthesize a gate level

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.