Chapter 18

Global Architecture

A single user process (as shown in Chapter 16) "views" the CM-5 system as a set of processing nodes plus a partition manager, with I/O and other extra-partitional activities being provided by the operating system.

Supporting such processes, however, requires that the underlying system software make appropriate use of the global architecture provided by the CM-5's communications networks.

All the computational and I/O components of a CM-5 system interact through two networks, the Control Network and the Data Network. Every such component is connected through a standard CM-5 Network Interface. The NI presents a simple, synchronous 64-bit bus interface to a node or I/O processor, decoupling it both logically and electrically from the details of the network implementation.

The Control Network supports communication patterns that may involve all the processors in a single operation; these include broadcasting, reduction, parallel prefix, synchronization, and error signaling. The Data Network supports point-to-point communications among the processors, with many independent messages in transit at once.

18.1 The Network Interface

The CM-5 Network Interface provides a memory-mapped control-register interface to a 64-bit processor memory bus. All network operations are initiated by writing data to specific addresses in the bus address space.

Many of the control registers appear to be at more than one location in the physical address space. When a control register is accessed, additional information is conveyed by the choice of which of its physical addresses was used for the

Copyright © 1993 Thinking Machines Corporation

access; in other words, information is encoded in the address bits. For example, when the Control Network is to be used for a combining operation, the first — and perhaps only — bus transaction writes the data to be combined, and the choice of address indicates which combining operation is to be used. One of the address bits indicates whether the access has supervisor privileges; an error is signaled on an attempt to perform a privileged access using an unprivileged address. (Normally the operating system maps the unprivileged addresses into the address space of the user process, thereby giving the user program zero-overhead access to the network hardware while prohibiting user access to privileged features.)

The logical interface is divided into a number of functional interfaces. See Figure 33. Each functional interface presents two FIFO interfaces, one for outgoing data and one for incoming data. A processor writes messages to the outgoing FIFO and pulls messages from the incoming FIFO, using the same basic protocol for each functional interface. Different functional interfaces, however, respond in different ways to these messages. For example, a Data Network interface treats

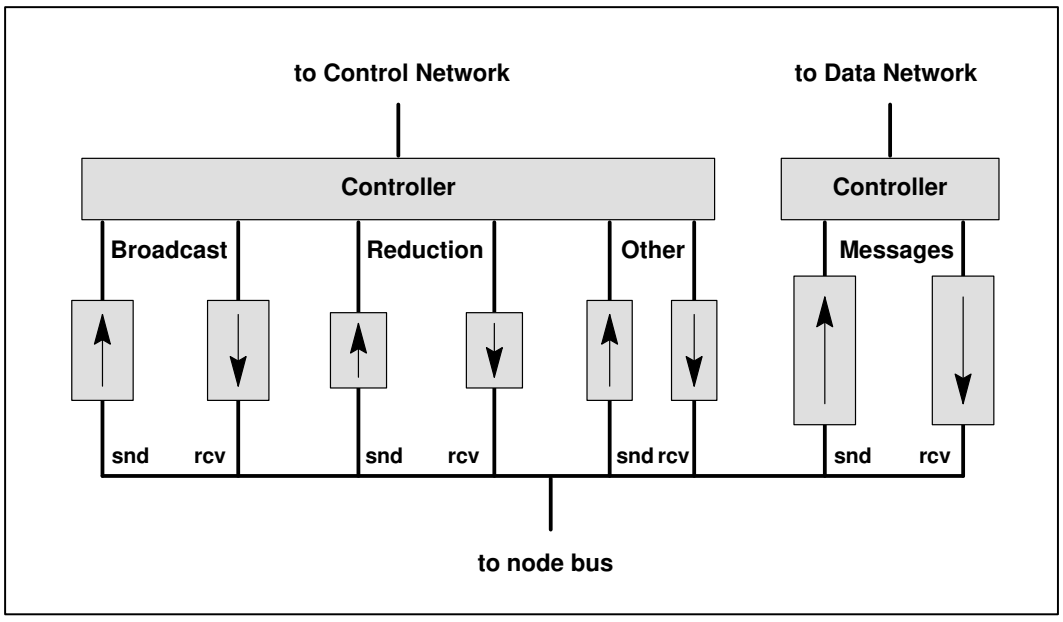

Figure 33. CM-5 Network Interface.

The Network Interface contains several functional interfaces. Each is a pair of FIFO buffers at particular memory locations. Through memory mapping, the Data Network and Control Network are directly accessible by user code.

Copyright © 1993 Thinking Machines Corporation

the first 32 bits of a message as a destination address to which to send the remainder of the message; a Control Network combining interface forwards the message to be summed (or otherwise combined) with similar messages from all the other processors.

Data is kept in each FIFO in 32-bit chunks. The memory-bus interface accepts both 32-bit and 64-bit bus transactions. Writing 64 bits thus pushes two 32-bit chunks onto an output FIFO; reading 64 bits pulls two chunks from an input FIFO.

For outgoing data, there are two control registers called **send** and **send_first**. Writing data to the **send_first** register initiates an outgoing message; address bits encode the intended total length of the message (measured in 32-bit chunks). Any additional data for that message is then written to the **send** register. After all the data for that message has been written, the program can test the **send_ok** bit in a third control register. If the bit is 1, then the network has accepted the message and bears all further responsibility for handling it. If the bit is 0, then the data was not accepted (because the FIFO overflowed) and the entire message must be re-pushed into the FIFO at a later time. The **send_space** control register may be checked before starting a message to see whether there is enough space in the FIFO to hold the entire message; this should be treated only as a hint, however, because supervisor operations (such as task switching) might invalidate it. In many situations throughput is improved by pushing without checking first, in the expectation that the FIFO will empty out as fast as new data is being pushed. It is also permissible to check the **send_ok** bit before all the data words for the message have been pushed; if it is 0, the message may be retried immediately.

For incoming data, a processor can poll the **receive_ok** bit until it becomes 1, indicating that a message has arrived; alternatively, it can request that certain types of messages trigger an interrupt on arrival. In either case, the program can then check the **receive_length_left** field to find out how long the message is and then read the appropriate number of data words from the **receive** control register.

The supervisor can always interrupt a user program and send its own message; this is done by deleting any partial user message, sending the supervisor message, and then forcing the **send_ok** bit for that unit to 0 before resuming the user program. To the user program it merely appears that the FIFO was temporarily full; the user program should then retry sending the message. The supervisor can also lock a send-FIFO, in which case it appears always to be full, or disable it, in which case user access will cause an interrupt. The supervisor can save and transparently restore the contents of any receive-FIFO.

Copyright © 1993 Thinking Machines Corporation

Each Network Interface records interrupt signals and error conditions generated within its associated processor, exchanges error and interrupt information with the Control Network, and forwards interrupt and reset signals to its associated processor.

18.2 The Control Network

Each Network Interface contains an assortment of functional interfaces associated with the Control Network. All have the same dual-FIFO organization but differ in detailed function.

Every Control Network operation potentially involves every processor. A processor may push a message into one of its functional interfaces at any time; shortly after all processors have pushed messages, the result becomes available to all processors. Messages of each type may be pipelined; a number of messages may be sent before any results are received and removed. (The exact depth of the pipeline varies from one functional interface to another.) The general idea is that every processor should send the same kinds of messages in the same order. The Control Network, however, makes no restrictions about when each processor sends or receives messages. In other words, processors need not be exactly synchronized to the Control Network; rather, the Control Network is the very means by which processors conduct synchronized communication *en masse*.

There are exceptions to the rule that every processor must participate. The functional interfaces contain mode bits for *abstaining*. A processor may set the appropriate mode bit in its Network Interface in order to abstain from a particular type of operation; each operation of that type will then proceed without input from that processor or without delivering a result to that processor. A *participating* processor is one that is not abstaining from a particular kind of Control Network operation.

Broadcasting

The *broadcast* interface handles broadcasting operations. There are actually three distinct broadcasting interfaces: one for user broadcast, one for supervisor broadcast, and one for interrupt broadcast. Access to the supervisor broadcast interface or interrupt broadcast interface is a privileged operation.

November 1993
Copyright © 1993 Thinking Machines Corporation

Only one processor may broadcast at a time. If another processor attempts to send a broadcast message before completion of a previous broadcast operation, the Control Network signals an error.

A broadcast message is one to fifteen 32-bit words long. Shortly after a message is pushed into the broadcast send-FIFO, copies of the message are delivered to all participating processors. The user broadcast and supervisor broadcast interfaces are identical in function except that the latter is reserved for supervisor use.

An interrupt broadcast message causes every processor to receive an interrupt or reset signal. A processor can abstain from receiving interrupts, in which case it ignores interrupt messages when it receives them; but a processor cannot abstain from a reset signal (which causes the receiving NI and its associated processor to be reset).

As an example of the use of broadcast interrupts, consider a partition manager coordinating the task-switching of user processes. When it is time to switch tasks, the PM uses the Control Network to send a broadcast interrupt to all nodes in the partition. This transfers control in each node to supervisor code, which can then read additional supervisor broadcast information about the task-switch operation (such as which task is up next).

Combining

The *combine* interface handles reduction and parallel prefix operations. A combine message is 32 to 128 bits long and is treated as a single integer value. There are four possible message types: reduction, parallel prefix, parallel suffix, and router-done. The router-done operation is simply a specialized logical OR reduction that assists the processors in a protocol to determine whether Data Network communications are complete. Reduction, parallel prefix, and parallel suffix may combine the messages using any one of five operators: bitwise logical OR, bitwise logical XOR, signed maximum, signed integer addition, and unsigned integer addition. (The only difference between signed and unsigned addition is in the reporting of overflow.) The message type and desired combining operation are encoded by address bits when writing the destination address to the **`send_first`** register.

As an example, every processor might write a 64-bit integer to the combine interface, specifying signed integer addition reduction. Shortly after the last participating processors write their input values, the signed sum is delivered to

Copyright © 1993 Thinking Machines Corporation

every participating processor, along with an indication of whether overflow occurred at any intermediate step.

As another example, every processor might write a 32-bit integer to the combine interface, specifying signed maximum parallel prefix. Shortly after the last participating processors write their input values, every participating processor receives the largest among the values provided by itself and all lower-numbered processors.

The combine interface also supports segmented parallel prefix (and parallel suffix) operations. Each combine interface contains a **`scan_start`** flag; when this flag is 1, that NI is considered to begin a new segment for purposes of parallel prefix operations.

Every participating processor must specify the same message type and combining operation. If, in the course of processing combine requests in order, the Control Network encounters different combine requests at the same time, it signals an error.

Global Operations

Global bit operations produce the logical OR of one bit from every participating processor. There are three independent global operation interfaces, one synchronous and two asynchronous, which may be used completely independently of each other and of other Control Network functions. This makes them useful for signaling conditions and exceptions.

The synchronous global interface is similar to the combine interface except that the operation is always a logical OR reduction and each message consists of a single bit. Processors may provide their values at any time; shortly after the last participating processors have written their input bits, the logical OR is delivered as a single-bit message to every participating processor.

Each asynchronous global interface produces a new value any time the value of any input is changed. Input values are continually transported, combined, and delivered throughout the Control Network without waiting for all processors to participate. Processors may alter their input bits at any time. These interfaces are best used to detect the transition from 0 to 1 in any processor or to detect the transition from 1 to 0 in all processors. (The NI signals an interrupt, if enabled, whenever a transition from 0 to 1 is observed.)

November 1993
Copyright © 1993 Thinking Machines Corporation

There are two asynchronous global interfaces, one for the user and one for the supervisor. Access to the supervisor asynchronous global interface is a privileged operation.

Synchronization

Both the synchronous global interface and the combine interface may be used to implement barrier synchronization: if every processor writes a message and then waits for the result, no processor passes the barrier until every processor has reached the barrier. The hardware implementation of this function provides extremely rapid synchronization of thousands of processors at once. Note that the router-done combine operation is designed specifically to support barrier synchronization during a Data Network operation, so that no processor abandons its effort to receive messages until all processors have indicated that they are no longer sending messages.

Flushing the Control Network

There is a special functional interface for clearing the intermediate state of combine messages, which may be required if an error or task switch occurs in the middle of a combine operation. A flush message behaves very much like a broadcast message: shortly after one processor has sent such a message, all processors are notified that the flush operation has completed. Access to the flush functional interface is a privileged operation.

Error Handling

The Control Network is responsible for detecting certain kinds of communications errors, such as an attempt to specify different combining operations at the same time. More important, it is responsible for distributing error signals throughout the system. Hard error signals are collected from the Data Network and all Network Interfaces; these error signals are combined by logical OR operations and the result is redistributed to every Network Interface.

Copyright © 1993 Thinking Machines Corporation

18.3 The Data Network

Each Network Interface contains one Data Network functional interface. The first 32-bit chunk of a message is treated as a destination address; it must be followed by one to five additional 32-bit chunks of data. This data is sent through the Data Network and delivered to the receive-FIFO of the Network Interface at the specified destination address. Each message also bears a 4-bit tag, which is encoded by address bits when writing the destination address to the **`send_first`** register. The tag provides a cheap way to differentiate among a small number of message types. The supervisor can reserve certain tags for its own use; any attempt by the user to send a message with a reserved tag signals an error. The supervisor also controls a 16-bit interrupt mask register; when a message arrives, an interrupt is signaled to the destination processor if the mask bit corresponding to the message's tag value is 1.

A destination address may be physical or relative. A physical address specifies a particular Network Interface that may be anywhere in the system and is not checked for validity. Using a physical address is a privileged operation. A relative address is bounds-checked, passed through a translation table, and added to a base register. A relative destination address is thus very much like a virtual memory address: it provides to a user program the illusion of a contiguous *address space* for the nodes running from 0 to one less than the number of processing elements. Access to the bounds register, translation table, or base register is a privileged operation; thus the supervisor can confine user messages within a partition.

While programs may use an interrupt protocol to process received messages, data parallel programs usually use a receiver-polls protocol in which all processors participate. In the general case, each processor has some number of messages to send (possibly none). Each processor alternates between pushing outgoing messages onto its Data Network send-FIFO and checking its Data Network receive-FIFO. If any attempt to send a message fails, that processor should then check the receive-FIFO for incoming messages. Once a processor has sent all its outgoing messages, it uses the Control Network combine interface to assert this fact; it then alternates between receiving incoming messages and checking the Control Network. When all processors have asserted that they are done sending messages and all outstanding messages have been received, the Control Network asserts the router-done signal to indicate to all the processors that the communications step is complete and they may proceed.

For task-switching purposes, the supervisor can put the Data Network into All Fall Down (AFD) mode. Instead of trying to route messages to their intended destinations, the Data Network drops each one into the nearest node. The advan-

November 1993
Copyright © 1993 Thinking Machines Corporation

tage of this strategy is that no node receives more than a few hundred bytes of AFD messages, even if they were all originally intended for a single destination. The supervisor can then read them from the Data Network receive-FIFO and save them in memory as part of the user task state, re-sending them when that user task is resumed.

Copyright © 1993 Thinking Machines Corporation