Distributed processing with the Z8000 family

by RICHARD MATEOSIAN and JANAK PATHAK
Zilog
Campbell, California

ABSTRACT

The Z8000 Family plan philosophy envisions a distributed processing approach to many Z8000 applications. The Z8000 Family consists of CPUs, CPU support circuits, and a full complement of VLSI peripherals. These components are all integrated by the Z-BUS, which defines the interconnections and transactions among them. The basic philosophy of the family plan is that of distribution of intelligence and function among complementary VLSI components. Of the several possible realizations of this philosophy, the one chosen has the following major aspects:

1. Synchronization primitives in bus and component architectures
2. Extensively programmable VLSI peripherals and CPU support circuits
3. Bus support for cooperative transactions
4. Built-in support for interprocess message passing

SYNCHRONIZATION PRIMITIVES

The Z-BUS has two features specifically designed for intercomponent synchronization in a distributed processing environment:

1. The "bus lock" status code
2. The resource request lines

Each of these bus features is designed to work with specific CPU instructions.

The "Bus Lock" Status Code

The "bus lock" status code is one of the 16 possible codes representable on the status lines ST_3-ST_0 of the Z-BUS. This status occurs during the fetch cycle of the Test and Set (TSET) instruction, which is available on all Z8000 CPUs. The TSET instruction is used to implement semaphores. Its job is to test a specified memory location for a predefined "available" code and to set the contents of the memory location to "not available." The inclusion of these two actions in a single instruction prevents any access to the specified location between the testing and the setting. That is, it prevents access by any other process running on the same CPU, which might happen if an interrupt occurred between separate testing and setting instructions. When other devices, such as another CPU or a DMA controller, have access to the same memory as the CPU executing the TSET instruction, the testing and setting operations must be inseparable at the bus transaction level. This inseparability is implemented through use of the "bus lock" status code.

The Resource Request Lines

In some distributed systems, several CPUs that do not share a common memory may need to share a common resource. In this case, the TSET instruction cannot be used. For such situations, the resource request lines of the Z-BUS have been provided. Figure 1 shows a prosaic example of their use: three CPUs sharing a line printer. When a CPU needs to use the line printer, it executes the MREQ instruction, which conducts a transaction on the four resource request lines; condition code settings indicate to the program whether or not the CPU gained control of the line printer through this transaction. If not, the MREQ instruction is executed again; if so, the line printer is used, then released through execution of the MRES instruction. If another CPU executes an MREQ instruction while the line printer is being held, the resource request transaction results in a "not available" indication.

PROGRAMMABLE VLSI COMPONENTS

The use of extensively programmable VLSI peripherals and CPU support circuits brings aspects of distributed processing into most Z8000 applications, even those with only a single CPU. The principal programmable VLSI components of the Z8000 Family are summarized below.

Memory Management Unit (MMU)

The MMU provides address translation and access protection, using internal tables transmitted from the CPU. Because of the Z8000's segmented addressing, which allows segment

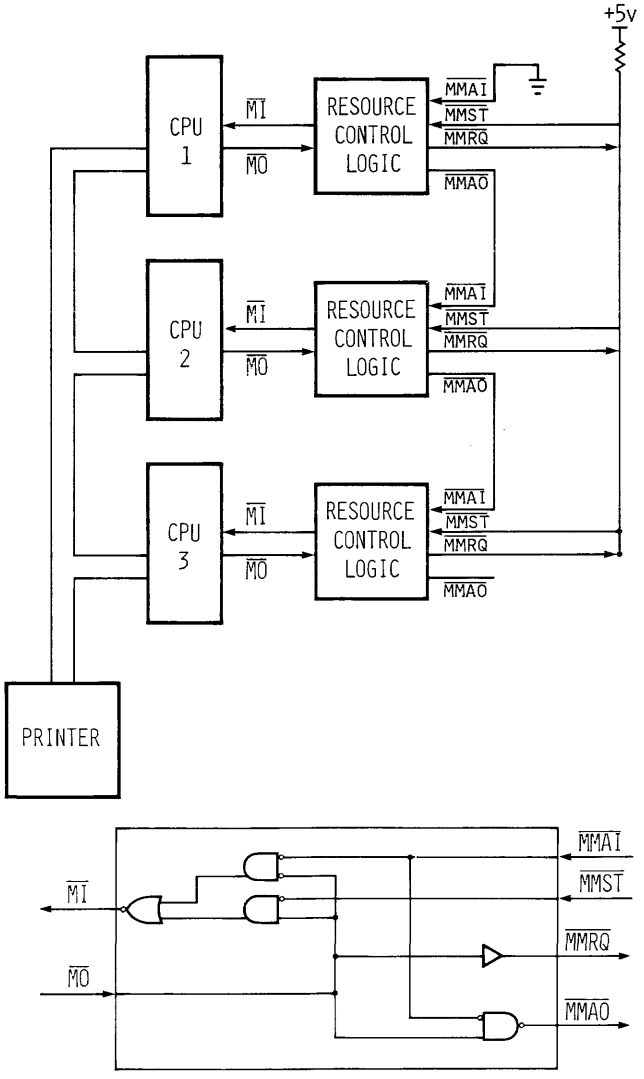

Figure 1—Resource lines provide non-memory-based synchronization

identity to be output by the CPU before completion of the indexing portion of address computations, the segment-related address processing done by the MMU occurs in parallel with the CPU's indexing. This parallel processing approach minimizes the overhead of external address translation and access protection.

DMA Transfer Controller (DTC)

The DTC can carry out high-speed block data transfers and searches independently of the CPU's operation. Control of the DTC by linked lists of command blocks in memory allows the DTC and CPU to carry out joint functions asynchronously. When an MMU is in the configuration, the DTC can work with logical or physical addresses. A special control line and a bit in the MMU access control registers allow the MMU to protect certain blocks of memory from DMA transfers and to prevent CPU access to blocks of memory while they are being changed by a DMA transfer.

FIFO Input/Output Interface Unit (FIO)

The FIO allows asynchronous parallel data transfers between processors, making it a key element in distributed multi-processor systems (see Figure 2).

The FIO is simply a 128-byte, first-in-first-out buffer, expandable in width and depth, equipped with bidirectional parallel interfaces at each "end" of the buffer and a set of message registers for interprocessor communications that bypass the buffer. The FIO is designed to cooperate with the DTC in "flyby" transfers (described below) to initiate DMA transfers without CPU involvement and to terminate DMA transfers on the basis of patterns recognized in the transferred data.

The Counter/Timer and Parallel I/O Unit (CIO)

The CIO has many functions related to real-time I/O processing. It is not a separate I/O processing CPU for use with the Z8000, but it does perform many of the same functions: bidirectional parallel I/O with a variety of handshake modes, counting and timing of external signals, and priority interrupt control.

The Serial Communications Controller (SCC)

The SCC, like the CIO, carries out many of the functions of a dedicated CPU working with the Z8000. It performs all of the tasks associated with serial communications on two independent 1Mbit/second channels, using any of a variety of protocols.

COOPERATIVE TRANSACTIONS

An essential element of the Z8000's distributed processing Family plan is the use of cooperative transactions. The principal examples are:

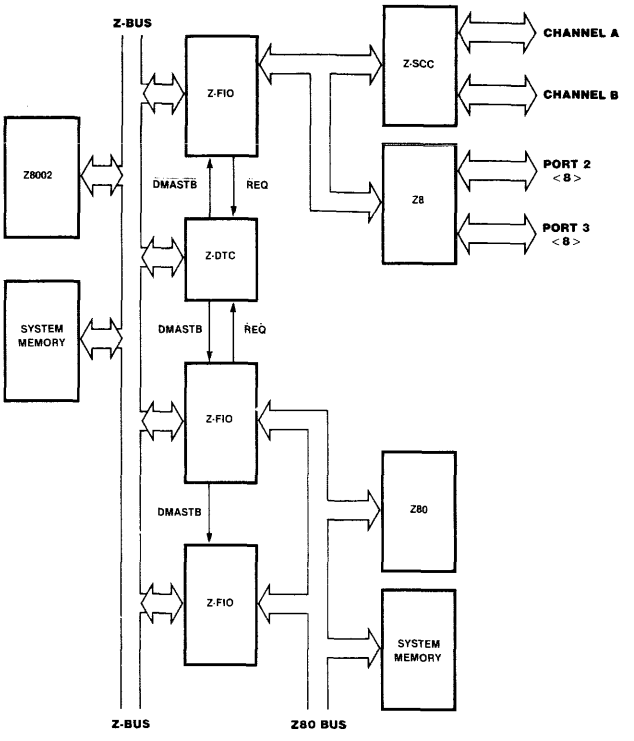

Figure 2—FIO links processors and cooperates in DMA transfers

1. CPU/MMU generation of physical addresses
2. Extended processing architecture
3. DTC/FIO "flyby" transfers

The common theme behind cooperative transfers is that each device has specific capabilities and that when a task requires a combination of capabilities, it is better to allow several devices to participate in the task than to replicate capabilities in several devices. Thus, for example, rather than equipping the FIO with DMA transfer capabilities, it was deemed more sensible to provide for joint DTC/FIO transfers.

Of the three examples of cooperative transfers listed above, CPU and MMU cooperation has already been discussed. The other two examples will now be described.

Extended Processing Architecture

An important goal of the Z8000 Family design was to accommodate additional processing capabilities (such as what would be provided by a floating point chip) with no redesign of the overall system or software. This goal was achieved with a scheme that allows certain CPU instructions either to cause traps (allowing simulation of an absent chip's function) or to be executed cooperatively by the CPU and an extended processing unit (EPU). With this cooperative approach, the CPU's addressing capabilities are used to fetch or store the arguments, and the EPU performs the operations. EPU operation can proceed in parallel with the execution of subsequent instructions by the CPU; synchronization is achieved by the EPU's assertion of the CPU's STOP line if the CPU fetches

another EPU instruction before the EPU is ready to execute it. Figure 3 illustrates the cooperation of the EPU and the CPU.

The Extended Processor Architecture gives designers a great deal of flexibility. For example, an EPU doing floating point operations could be used interchangeably with floating point software controlled by the same instruction stream; only a single bit in the CPU's Flag/Control Word (FCW) control register would need to change. Thus, a high-performance floating point chip could be an optional feature of a product that used floating point operations. The "slow" version would use software execution of the floating point instructions, and the "fast" version would use the chip to execute instructions. Both versions would have identical applications program code and circuitry.

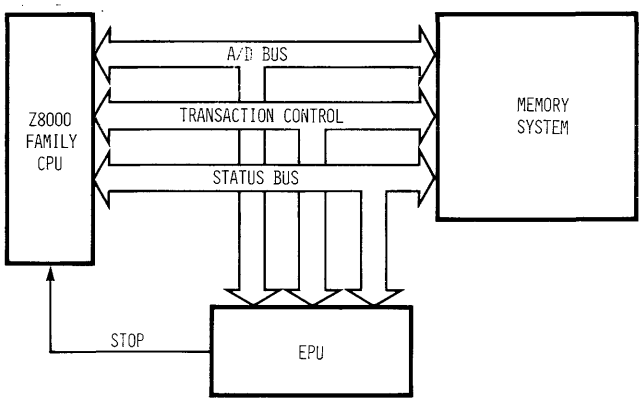

Figure 3—CPU and EPU cooperate to execute instructions

The EPU monitors the status lines, looking for "Instruction Fetch, First Word" status. When this occurs, it examines the instruction presented on the A/D bus. If the instruction is for that EPU, it either asserts STOP (if it is still busy executing a previous instruction) or initiates execution of the indicated instruction. The EPU instruction can be entirely internal to the EPU, or it can include one or more transfers of data between the EPU and CPU or EPU and memory. For each of these cases, the CPU generates the appropriate status signal (ST_3-ST_0) and transaction control (R/W, B/W, AS, MREQ, DS) lines, and the EPU takes or supplies data as appropriate.

Flyby Transfers

A "flyby" transfer is a DMA transfer in which the data never enters the DMA controller circuit. The DMA controller provides all necessary memory addressing, transfer counts, and bus control signals, but at the point in the transaction when data must pass from one component to another, an intelligent peripheral (like the FIO) supplies or takes the data. Flyby transfers are, therefore, approximately twice as fast as ordinary DMA transfers, in which one transaction is required to fetch the data from the source and to latch it in the DMA controller, and a second transaction is required to pass the data from the DMA controller to the destination.

SUPPORT FOR MESSAGE PASSING

The support for message passing in the Z8000 Family plan is predicated on the assumption that interprocess communication in Z8000 systems can be conducted effectively through messages. Other means of interprocess communication are not precluded, but message passing is the only interprocess communication method supported by special architectural features.

Since message passing is generally implemented through the movement of blocks of characters from one location to another, one of the principal means of supporting message passing in the Z8000 Family plan is the multi-level support of block data movement. The block I/O and memory transfer instructions of the CPU, the capabilities of the DTC, and the features of the FIO are all designed to complement each other in providing efficient, flexible block data movement throughout Z8000 systems.

Another instance of message passing occurs in the communication protocol defined between the Z8000 CPUs and the Universal Peripheral Controller (UPC). The UPC is a Z8-based single-chip microcomputer designed for use in device controllers. It functions as a slave processor to the CPU, and because it is directly tied to the operation of a physical device, it is essential that a faulty CPU program not cause the UPC to fail.

The fail-safe protocol for CPU/UPC communication calls for designation by the UPC of specific blocks of its internal memory for use as shared message buffers. The CPU has direct access to the designated buffer area but cannot access any other portion of the UPC's memory until the UPC designates that portion as the message buffer. The CPU always sees a single address in its I/O address space as "UPC message buffer," but the UPC maps this address internally into the desired area of its memory.

SUMMARY

Distributed processing with the Z8000 Family is not a special case. The distribution of function among CPU and extensively programmable VLSI components demands that the basic mechanisms of communication and synchronization be included in the design of the Z-BUS and all the Z8000 Family components. In addition, specific attention has been given to multi-CPU system problems through use of specific CPU instructions and bus protocols and through use of the First-In-First-Out Interface Unit (FIO) as a flexible buffer between asynchronously functioning systems. Cooperative transactions, in which the functions of several components must combine to carry out the desired action, bring distribution of function to the bus and component level. Finally, architectural features supporting message passing facilitate distributed processing at the software and application structuring level.