An inside look at the Z80,000 CPU:
Zilog's new 32-bit microprocessor

by ANIL PATEL
Zilog Corporation
Campbell, California

ABSTRACT

With recent advances in very large scale integrated circuit (VLSI) design, the once-distinct boundaries of micro-, mini-, and mainframe computer architectures are eroding. For example, the Motorola 68000, the Z8000 CPU, and the Intel 8086 have broken the once-distinct boundary between micro- and minicomputers. Now the Z80,000 CPU, Zilog’s new 32-bit processor chip, has broken the distinct boundary between mini- and mainframe computers by featuring a mainframe power on an integrated-circuit chip. The distinguishing features of the Z80,000 CPU—such as on-chip virtual memory management, on-chip cache memory, six-stage pipeline architecture, burst memory transfer, and multiprocessing support—put it ahead of any CPU in its class.

The CPU supports linear and segmented addressing. The regular instruction set and rich and powerful addressing modes are well suited to compilers and operating systems. The flexibility and simplicity of the Z80,000 provide an easy solution to hardware and software system design.
ARCHITECTURE

The Z80,000 CPU is a register-oriented machine that provides sixteen 32-bit general purpose registers (Figure 1). The registers are truly general purpose, with no restrictions on their use as accumulators, addressing registers, Stack Pointers, or data registers. Therefore, bottlenecks encountered with register organizations that dedicate specific registers for accumulators (or data) and addressing are eliminated. In addition, because any register can be used as a Stack Pointer, the registers lend themselves to high-level language support by providing the multiple Stack Pointers required for parsing operations.

The organization of the registers also provides for efficient handling of mixed data types. Registers can be used for 8- or 16-bit arithmetic and logical operations without loss of the high-order 24 and 16 bits, respectively, giving the effect of a much larger register space. In addition, 32-bit registers can be paired for 64-bit data.

The Z80,000 CPU uses 32-bit logical addresses to directly access up to 4 gigabytes of memory in each of 4 address spaces. Separate address spaces are provided for system and normal modes and for instructions and data. The programmer has available four different address representations in accessing the memory space (Figure 2), providing maximum flexibility in applying the processor to the specific requirements of the application environment. Two bits in the flag and control word (FCW) select compact, segmented, or linear address representation.

Compact mode uses a 16-bit address, which gets concatenated to the upper 16 bits of the Program Counter to form a 32-bit address. Programs operating within a 64K workspace can take advantage of the compact mode's dense code and efficient use of base registers.

Many applications lend themselves to the use of segmented mode, where individual objects are allocated to separate protected segments. The segment remains unchanged during address calculations; only the offset is affected. There are two segment sizes available with the Z80,000 CPU, controlled by the most significant bit of the address field. Thus, the programmer has the flexibility of having 128 segments of up to 16 megabytes, and 32K segments of up to 64K in size.

Applications requiring a large linear address space without the formal structure of segmentation include graphics and the processing of large arrays. Additionally, with the availability of 32 bits of addressing, certain application-specific implementations use address lines creatively and would otherwise be hampered by the structure imposed by segmentation. The Z80,000 CPU supports 32-bit linear addressing, as well as segmented and compact addressing, to provide maximum flexibility to the system designer.

Nine general addressing modes provide efficient access to the many types of data structures. A rich instruction set combines with the address modes to operate on a variety of data types, including 8-, 16-, and 32-bit integer and logical values, as well as bits, bit fields, packed decimal, and dynamic length strings. Additionally, high-level language support is enhanced by instructions for procedure linkage, array indexing, and integer conversion, as well as the more common operations.

A separate system mode, with its own Stack Pointer and protected address space, supports operating systems. Because some instructions are privileged, executing only in system mode, the operating system and system resources are protected from programs operating in normal mode. The System Call instruction is used by the normal mode program to communicate with the operating system through the Z80,000 CPU trap facility. The processor also includes an extensive trap mechanism for run-time error detection and software debugging.

MEMORY MANAGEMENT

The Z80,000 CPU memory management mechanism has been integrated with the CPU on-chip, offering two primary advantages to the system designer: a parts count reduction and faster access to memory. Memory access is faster because the CPU generates physical addresses, thus eliminating the delay of an external memory mapping device.

The CPU's Paged Memory Management Unit (PMMU) provides translation of logical addresses to physical addresses,
memory access protection, and protected access to memory mapped I/O devices. Demand-paged virtual memory is easily implemented without special software recovery routines or storage of the internal state following address translation faults. The implementation is accomplished through early detection of translation faults, resulting in the ability to restart all instructions efficiently. Besides providing access protection, the page attribute mechanism also contains referenced and modified bits that aid the operating system in determining which page in physical memory should be swapped with the required page from mass storage.

To manage the Z80,000 CPU’s 4G-byte logical address space, the translation scheme divides it into fixed-size, 1K-byte pages. The logical address’s 22 high-order bits select a page in the address space, while the 10 least significant bits select a byte within the page. Similarly, physical memory is divided into 1K-byte units, called frames. The memory management unit maps a logical page to a frame. Having both logical and physical units of the same size simplifies the operating system’s memory allocation problem.

The CPU and operating system cooperate to translate a program’s logical addresses into physical addresses that are used to access memory. The CPU’s paging scheme is similar to that of most mainframes and super-minicomputers. First, the operating system creates translation tables in memory, then initializes pointers to the tables in control registers. The CPU automatically references the tables to perform the address translation and access protection for each memory access. Delays that would be associated with referencing the translation tables are minimized by using an additional on-chip cache associated with the Memory Management Unit, the Translation Lookaside Buffer (TLB). Logical addresses, their corresponding physical addresses, and access attributes are stored in the TLB (Figure 3) as they are translated through
an inside look at the Z80,000 CPU

the translation tables. Subsequent accesses to the same page do not require access to the translation tables stored in memory; they are simply retrieved from the TLB. The tables are accessed only when an entry does not exist in the TLB, a TLB "miss." The least recently used entry is then replaced with the new address translation and access information. The TLB can hold the 16 most recently referenced pages, providing a TLB hit ratio that is typically over 96%.

The Z80,000 CPU implements a three-level address translation process. Once the operating system creates the translation tables and initializes the control registers, the CPU automatically references the level-1, level-2, and page tables to perform address translation and access protection (Figure 4). Access protection is encoded in a 4-bit field at any level of the translation process. This allows access protection to be accomplished at the page level, level-1, level-2, or a mixture of the 3. The use of 3 levels of translation is dictated by the 32-bit logical address of the Z80,000 CPU, whereas a 2-level translation mechanism would be appropriate for 24-bit logical addresses.

It is possible to reduce the number of levels of translation by specifying in the table descriptor registers (the control registers containing pointers to the translation tables) that either or both the level-1 and level-2 tables should be omitted during the translation process. Skipping level-1 tables is useful when only a 24-bit address space is required. Both tables can be skipped when 16-bit addressing is sufficient for the needs of the application. Additionally, the tables can be reduced in size by specifying in the table entries the size of the next level table in increments of 256 bytes. Thus, maximum flexibility in translation, access protection, and table organization is maintained by the Z80,000 CPU memory management implementation.

performance boosters: cache, six-stage pipeline, burst memory transfer

the Z80,000 CPU implementation includes a six-stage pipeline (Figure 5) supported by two 32-bit ALUs, one assigned to address calculation and the other associated with the execution stage. The pipeline allows concurrent operation of up to six instructions.

All pipeline stages can operate in a single processor cycle, which is composed of two clock cycles. The pipeline allows simple instructions, such as register-to-register Load and memory-to-register Add, to be executed at a rate of one instruction for each processor cycle, leading to a peak performance of 12.5 million instructions per second with a 25-MHz clock. In practice, the actual instruction rate is about one-third of the peak rate because of certain delays.

Because the pipeline may require two memory fetches during each processor cycle—one to fetch the instruction and the other for the operand fetch stage—it is necessary to buffer the high execution rate of the pipeline from the relatively slow memory access rate. Because memory fetches typically take three or more bus cycles, the pipeline would be idle most of the time if all references had to access main memory. By including an on-chip cache that can be accessed in one processor cycle, most memory references can be made without external bus transactions, allowing the pipeline to function at an extremely high level of performance.

The cache holds copies of the most recently accessed memory locations. On each memory fetch, the CPU examines the cache to determine if the data at that address is available on chip, in other words, a cache "hit." If available, the data is read from the cache rather than from external memory. If it is not available, a cache "miss," the CPU generates an
external memory transaction to fetch the data and then stores the fetched information in the cache. The Z80,000 CPU cache is organized as 16 lines, or blocks, of 16 bytes, for a total of 256 bytes of data (Figure 6). Each block is associated with a 28-bit tag that represents the most significant bits of the address of the block. The lower 4 bits of the address select the appropriate byte, word, or longword within the block. There are eight validity bits, each corresponding to a word within the block. This structure represents an optimum tradeoff between performance and silicon area (cost).

The Z80,000 CPU cache is mode programmable to best fit the requirements of the application. Modes include instruction only, data only, instruction/data (all mainframes implement instruction/data), and local memory. Whereas particular applications for the Z80,000 CPU may require instructions only to be cached, caching data along with instructions will typically increase cache performance by 20%. Local memory allows a specific address to be assigned to each block; thus, the cache takes on the form of an extremely fast 256-byte memory. For example, in a highly intensive interrupt driven environment, the interrupt service routines (ISR) may be allocated to the on-chip local memory to maximize ISR throughput.

For references requiring operand stores, the data is always written to main memory. The cache is also updated if it contains the addressed location; otherwise it is unaffected. This mechanism, called write-through, ensures that main memory represents the most recent value stored at any address. Without the ability to write through cache to main memory, the CPU would be required to update memory whenever the least recently used cache line is flushed to allow space for new code or data during a cache miss operation. The write-through mechanism allows processing to continue, concurrent with the bus activity associated with the write. The pipeline allows concurrent operation because the next instruction is most likely to be present in the CPU. Additionally, burst transfers into cache further increase the probability that instructions are present on-chip, minimizing the potential of write-through operations conflicting with bus read transactions.

Increased bus bandwidth can be achieved by taking advantage of the optional burst transfer capability of the Z80,000 CPU bus interface. Burst memory transactions use multiple data strobes following each address strobe to transfer consecutive memory locations. The CPU uses burst transactions to prefetch a cache block on an instruction fetch cache miss. A read transaction with a single data strobe and one wait state
requires three bus clocks. However, with burst transfers, a transaction with four data strobes and one wait state requires six bus clocks, resulting in twice the bus bandwidth of the single transfer transaction. With a 12.5-MHz bus clock (25-
MHz CPU clock), 32-bit data path, and 4 data transfer per transaction, with no wait state, the bus bandwidth is 40 mega­bytes a second. Burst transactions are also used for fetching and storing operands when multiple transfers are necessary, such as string operations, Load Multiple instructions, and loading of program status, and when unaligned accesses occur.

MULTIPROCESSING

The CPU provides support for interconnection in four types of multiprocessor configurations (Figure 7): coprocessor, slave processor, tightly coupled multiple CPUs, and loosely coupled multiple CPUs. Coprocessors work concurrently with the CPU to execute a single instruction stream using the Extended Processing Architecture (EPA) facility. This allows extension of the Z80,000 CPU architecture to include floating point operations and other specialized functions. Additionally, the processing speeds offered by extended processing units (EPUs) optimized for particular operations, such as the Z8070 Arithmetic Processing Unit, can provide significant performance improvements.

When the CPU encounters an EPU instruction (and the EPA bit in the FCW is set to 1), it begins a CPU-to-EPU instruction transaction that broadcasts the first two words of the EPU instruction to all (as many as four) EPUs in the system. If a data transfer is required, the CPU and the selected EPU conduct the appropriate data transfer transaction. The CPU is the bus master, handling address translations and bus transactions. The EPUBSY signal is used by the CPU and EPUs to synchronize transfers. EPU operations are efficient because the CPU is not required to wait for completion of the EPU operation, and no elaborate handshaking is necessary. In fact, up to four EPUs can be actively processing data while the CPU handles other chores. In systems supporting the functionality of an extended processing unit without the actual EPU present (the EPA bit in the FCW is cleared to 0), the EPU instructions are trapped and emulated in software.

Slave processors, such as the Z8016 DMA Transfer Controller, perform dedicated functions asynchronously to the CPU. The CPU and slave processor share a local bus, of which the CPU is the default master. When the slave wishes to use the bus, it requests the bus using the BUSREQ line. The CPU responds by asserting BUSACK and placing all other output signals in 3-state. The slave then gains control of the bus (and in the case of the Z8016, it provides DMA capabilities). When the slave no longer needs the bus, it relinquishes the control back to the CPU.

Tightly coupled, multiple CPUs execute independent instruction streams and generally communicate through shared memory located on a common (global) bus using the CPU’s GREQ and GACK lines. Each CPU is default master only of its local bus; an external arbiter chooses the global bus master. The CPU also provides status information about interlocked memory references so that bus control is not relinquished during an indivisible operation such as Test and Set or Increment Interlocked.

The Z80,000 CPU’s I/O and interrupt facilities support loosely coupled multiple CPUs, which generally communicate through a multi-ported peripheral, such as the Z8038 FIFO I/O Controller.

EXCEPTION PROCESSING

The Z80,000 CPU supports four types of exceptions: reset, bus error, interrupts, and traps. A reset exception occurs when the reset line is activated. In responding to a reset exception, the CPU fetches the program status (FCW and PC)
from physical address 2 and resets itself into the initialized state.

When external hardware indicates a bus error exception on the memory response lines RSPo-RSPn, the CPU terminates the transaction in progress. The CPU also terminates the instruction in execution. In processing bus error exception, the CPU saves the program status, the physical address for the transaction, and a word identifying the status and control signals used for the transaction.

Three types of interrupts are supported: vectored, nonvectored, and nonmaskable. The vectored and nonvectored interrupts have mask bits in the FCW. All interrupts read an identifier word from the bus during an interrupt acknowledge transaction and save the word on the system stack. Vectored interrupts use the lower byte of this word to select a unique PC value from the program status area.

The CPU supports 12 trap conditions: extended instruction, privileged instruction, system call, address translation, reserved instruction, odd PC, trace, breakpoint, conditional, integer overflow, bounds check, and subscript error.

In descending order, the priority of exceptions is: reset, bus error, trap, nonmaskable interrupt, vectored interrupt, and nonvectored interrupt.

Z80,000 CPU PERFORMANCE

Cache memory and the pipelined structure cause the performance evaluation of the Z80,000 CPU to be complex. The best approach is separation of instruction processing time into a sum of three components: execution time, pipeline delays, and memory delays. Performance was evaluated by statistically measuring activities of 10 C language programs and then performing a computer simulation of the cache, Translation Lookaside Buffer, and pipeline mechanisms.

The execution time for an instruction is the number of cycles required to execute the instruction if there are no other delays such as cache miss or register interlock. Common instructions, such as loading a register with a word operand specified by a base-register-plus-displacement addressing mode, execute in 1 processor cycle (2 clock cycles), but the average instruction execution time is 1.3 processor cycles.

Pipeline delays are caused by branch instructions, register interlocks, and other miscellaneous delays. The most significant of these is delay due to branch instructions. When a branch is taken, instructions in the pipe behind the branch instruction are flushed. Unconditional branches introduce a delay of two processor cycles. Conditional branches cause a three processor cycles delay if the condition is met and no delay if the condition is not met. The average delay due to branches is 0.5 processor cycles per instruction.

Another significant pipeline delay is register interlock. Whenever the execution stage modifies a register that is to be used in a subsequent instruction as an address register, the address calculation must be held up (interlocked) until the execution is complete. The interlock ensures that the proper register value is used in the address calculation. The average register interlock delay is 0.2 processor cycles per instruction.

All the other miscellaneous delays add up to 0.2 processor cycles. Therefore, the total average pipeline delay is 0.9 processor cycles per instruction.

Memory delays are caused by cache misses and TLB misses. When the processor fetches an instruction or operand for which a corresponding entry in the cache or TLB does not exist, a reference to main memory is generated. The average delay due to these memory transfers is 1.2 cycles per instruction. This delay calculation is based on a 32-bit data path, a memory cycle time of 3 processor cycles, and support of burst transfers.

Instruction processing time, \( T_i = \) Execution delay + Pipeline delay + Memory delay.

Therefore \( T_i = 1.3 + 0.9 + 1.2 = 3.4 \) processor cycles.

The total processing time is an average of 3.4 processor cycles per instruction. At 10 MHz, this corresponds to 1.5 MIPS; at 25 MHz, the instruction execution rate is 3.7 MIPS.

EASE OF SYSTEM DESIGN

The Z80,000 CPU allows particular cost and performance objectives to be met by allowing designers to balance memory access and bus bandwidth appropriately and to incorporate burst transfers into designs. The Hardware Interface Control register (HICR) defines the characteristics of the hardware configuration surrounding the CPU. By setting appropriate bits in the HICR, the system designer can specify bus speed, memory data path, and the number of wait states to be automatically inserted for different types of bus accesses.

The bus speed can be one-half or one-fourth the CPU's clock frequency. Because the cache effectively decouples the CPU from the external bus, high processing rates can be achieved on-chip supported by an external bus that is not only easier to design but also less costly than one operating at the high clock frequencies of the Z80,000 CPU. A performance of 1.5 MIPS can be achieved at 10 MHz, using slow and inexpensive memory of 600-nanosecond memory cycle time. Using 240-nanosecond memory cycle time, a performance of 3.7 MIPS can be achieved at 25 MHz. In addition, because the bus can operate at two frequencies relative to the processor's clock, design migration to faster versions of the CPU will not incur major redevelopment. For instance, a 10 MHz Z80,000 design using a 5 MHz bus can be increased to 20 MHz while maintaining the same external bus speed.

The memory data path width can be specified separately for the upper and lower portions of the memory space as either 16 or 32 bits. The number of wait states to be automatically inserted during bus accessces can be specific to the upper and lower portions of the memory and I/O spaces. Thus, a system can accommodate a slow, 16-bit-side ROM and a fast 32-bit-wide RAM.

CONCLUSION

The Z80,000 provides the following benefits:

1. High performance
   a. On-chip Memory Management Unit (MMU)
b. On-chip cache—instruction/data
  c. Six-stage pipeline architecture with two 32-bit ALUs
  d. Burst memory transfers
  e. EPU overlap (CPU is able to run while coprocessor is running)
2. Flexible architecture
   a. Available linear, segmented, and compact addressing
   b. Programmable hardware configuration (e.g., bus speed, wait states)
   c. Support for multiprocessing: tightly coupled, loosely coupled, slave processor, coprocessor
3. Simple and regular architecture
   a. Regular use of operations, addressing modes, and data types in instruction set
   b. Rich and powerful addressing modes
4. Miscellaneous benefits
   a. Instruction set well suited for high-level, structured languages like C, PASCAL
   b. Architecture well suited for operating systems
   c. On-chip MMU for easy and cost-effective hardware design
   d. Simple memory management and task switching for operating system
   e. Largest virtual memory available per task
   f. Largest register set
   g. Execution rate of up to 12.5 MIPS
   h. Memory mapped I/O
   i. Single phase clock

The Z80,000 CPU addresses a wide range of system applications including high-performance, desk-top general purpose computing, graphics, and array processing, wherever mainframe performance is at low cost.