MC68030: The second generation 32-bit microprocessor

by MICHAEL RUHLAND
Motorola Microprocessor Products Division
Austin, Texas

ABSTRACT

The MC68030 is a virtual memory microprocessor based on an MC68020 core with additional enhanced performance features. Increased internal parallelism is provided by multiple internal data buses and address buses, and a versatile bus controller that supports synchronous burst cycle accesses in order to maximize performance with paged mode, nibble mode, and static column DRAM technology, or even external SRAM caches. A 256-byte on-chip instruction cache in addition to a 256-byte on-chip data cache improves data flow to the execution unit and further boosts performance regardless of the actual external memory configuration. On-chip paged memory management reduces the minimum physical bus cycle time to two clocks, and provides zero translation time to any bus cycle. The paged memory management structure can be enabled/disabled by software for applications not requiring the memory management feature. The rich instruction set and addressing modes of the MC68020 have been maintained allowing a clear migration path for M68000 systems.
INTRODUCTION

The MC68030 incorporates the capabilities of the MC68020 microprocessor, a data cache, an instruction cache, an improved bus controller, and an integrated memory management structure defined by the MC68851 Paged Memory Management Unit on one VLSI device. It maintains the 32-bit registers available with the entire M68000 family as well as the 32-bit address and data paths, rich instruction set, versatile addressing modes, and flexible coprocessor interface provided with the MC68020. In addition, the internal operations are designed to operate in parallel, allowing multiple instructions to be executed concurrently. It also allows instruction execution to proceed in parallel with accesses to the internal caches, the on-chip memory management unit, and the bus controller.

The MC68030 fully supports the non-multiplexed asynchronous bus of the MC68020 as well as a dynamic bus sizing mechanism that allows the processor to transfer operands to or from external devices while automatically determining device port size on a cycle-by-cycle basis. In addition to the asynchronous bus, the MC68030 also supports a fast and flexible synchronous bus. Using its synchronous bus capabilities, the MC68030 is capable of fetching up to four long words of data in a burst mode compatible with DRAM chips that have burst capability or SRAM. Burst mode can reduce, by up to 70%, the time necessary to fetch the four long words from physical memory. The four long words are used to prefill the on-chip instruction and data caches so that the hit ratio of the caches improves and the average access time is minimized.

The block diagram shown in Figure 1 depicts the major sections of the MC68030 and illustrates the autonomous nature of these blocks. The bus controller consists of the address and data pads, the multiplexors required to support dynamic bus sizing, and a macro bus controller which schedules the bus cycles on the basis of priority. The CPU contains the execution unit and all related control logic.

The instruction and data cache blocks operate independently from the rest of the machine, storing information read by the bus controller. Each cache resides on its own address and data buses, allowing simultaneous access to both. Both the caches are organized as 64 long word entries (256 bytes) with a block size of four long words. The data caches use a write-through policy.

Finally, the memory management unit controls the mapping of addresses for page sizes ranging from 256 bytes to 32K bytes. Mapping information stored in descriptors resides in translation tables in memory that are automatically searched by the MC68030 on demand. Recently used descriptors are maintained in a 22-entry fully associative cache called the Address Translation Cache (ATC) allowing address translation and other MC68030 functions to occur simultaneously. Additionally, the MC68030 contains two transparent translation registers that can be used to define a one-to-one mapping for two segments ranging in size from 16M bytes to 4G bytes each.

PROGRAMMING MODEL

As shown in the programming model (see Figure 2) the MC68030 has sixteen 32-bit general purpose registers, a 32-bit program counter, two 32-bit supervisor stack pointers, a 16-bit status register, a 32-bit vector base register, two 3-bit alternate function code registers, two 32-bit cache handling (address and control) registers, two 64-bit root pointer registers used by the MMU, a 32-bit translation control register, two 32-bit transparent translation registers, and a 16-bit MMU status register. Registers D0–D7 are used as data registers for bit and bit field (1 to 32 bit), byte (8 bit), word (16 bit), long word (32 bit), and quad word (64 bit) operations. Registers A0–A6 and the user, interrupt, and master stack pointers are address registers that may be used as software stack pointers or base address registers. In addition, the address registers may be used for word and long word operations. All of the 16 (D0–D7, A0–A7) registers may be used as index registers. The status register contains the interrupt priority mask as well as the condition codes. Additional control bits indicate that the processor is in a trace mode, supervisor/user state, and master/interrupt state.

The vector base register is used to determine the run-time location of the exception vector table in memory, hence it supports multiple vector tables so each process or task can properly manage exceptions independently of each other.

The M68000 Family processors distinguish address spaces as supervisor/user, program/data, and CPU space. These five
combinations are specified by the function code pins, FC0/FC1/FC2, during bus cycles, indicating the particular address space. Using the function codes, the memory subsystem (hardware) can distinguish between supervisor mode accesses and user accesses as well as program accesses, data accesses, and CPU space accesses. Additionally, the system software can configure the on-chip MMU so that supervisor/user privilege checking is performed by the address translation mechanism and the look-up of translation descriptors can be differentiated on the basis of function code. To support the full privileges of the supervisor, the alternate function code registers allow the supervisor to specify the function code for an access by preloading the SFC/DFC registers appropriately.

The cache registers (control-CACR, address-CAAR) allow supervisor software manipulation of the on-chip instruction and data caches. Control and status accesses to the caches are provided by the cache control register (CACR), while the cache address register (CAAR) specifies the address for those cache control functions that require an address.

All of the MMU registers (CRP, SRP, TC, TT0, TT1, and PSR) are accessible by the supervisor only. The CPU root pointer contains a descriptor for the first pointer to be used in the translation table search for page descriptors pertaining to the current task. If the SRE (Supervisor Root pointer Enable) bit of the translation control register is set, the supervisor root pointer is used as a pointer to the translation tables for all supervisor accesses. If the SRE bit is clear, this register is unused and the CPU root pointer is used for both supervisor and user translations. The translation control register configures the table look-up mechanism to be used for all table searches as well as the page size and any initial shift of logical address required by the operating system. In addition, this register has an enable bit that enables the MMU. The transparent translation registers can be used to define two transparent windows for transferring large blocks of data with untranslated addresses. Finally, the MMU status register (PSR) contains status information related to a specific address translation and the results generated by the PTEST instruction. This information can be useful in locating the cause of an MMU fault.

The MC68030 is upward source- and object-level code compatible with the M68000 Family because it supports all of the instructions that previous family members offer. Included in this set are the bit field operations, binary coded decimal support, bounds checking, additional trap conditions, and additional multi-processing support (CAS and CAS2 instructions) offered by the MC68020. Each instruction, with few exceptions, operates on bytes, words, and long words, and most instructions can use any of the 18 addressing modes. The new instructions supported by the MC68030 are a subset of the instructions introduced by the MC68851 paged memory management unit. The MMU instructions supported by the MC68030 are the PMOVE, PTEST, PLOAD, PFLUSH, and PFLUSH A instructions and they are completely compatible with the corresponding instructions on the MC68851 PMMU. Whereas the MC68851 required the coprocessor interface to execute its instructions, the MC68030 MMU instructions execute just like all other CPU instructions. All of the MMU instructions are privileged (can be executed by the supervisor only).

INSTRUCTION AND DATA CACHES

Studies have shown that typical programs spend most of their execution time in a few main routines or tight loops. This phenomenon is known as locality of reference, and has an impact on the performance of the program. The MC68010 takes limited advantage of this phenomenon with the loop mode of operation that can be used with the DBcc instruction. The MC68020 takes much more advantage of locality with its 256 byte on-chip instruction cache. The MC68030 takes fur-
ther advantage of cache technology to provide the system with two on-chip caches, one for instructions and one for data.

**MC68030 Cache Goals**

Similar to the MC68020, there were two primary goals for the MC68030 microprocessor caches. The first design goal was to reduce the processor external bus activity beyond what was accomplished with the MC68020. The second design goal was to increase effective CPU throughput as larger memory sizes or slower memories increased average access time. By placing a high speed cache between the processor and the rest of the memory system, the effective memory access time becomes:

\[
t_{\text{acc}} = h \cdot t_{\text{cache}} + (1 - h) \cdot t_{\text{rest}}
\]

where \(t_{\text{acc}}\) is the effective system access time, \(t_{\text{cache}}\) is the cache access time, \(t_{\text{rest}}\) is the access time of the rest of the system, and \(h\) is the hit ratio or the percentage of time that the data is found in the cache. Thus, for a given system design, two MC68030 on-chip caches provide an even more substantial CPU performance increase over that obtainable with the MC68020 with its instruction cache. Alternately, slower and less expensive memories can be used for the same processor performance.

The throughput increase in the MC68030 is gained in three ways. First, the MC68030 caches are accessed in less time than is required for external accesses, providing improvement in the access time for items residing in the cache. Secondly, the burst filling of the caches allows instruction and data words to be found in the on-chip caches the first time they are accessed by the micromachine, with the time required to bring those items into the cache minimized. This has the capability of lowering the average access time for items found in the caches even further.

Thirdly, and perhaps most importantly, the autonomous nature of the caches allows instruction stream fetches, data fetches, and a third external access to all occur simultaneously with instruction execution. For example, if the MC68030 requires both an instruction stream access and an external peripheral access, the peripheral access will proceed unimpeded rather than being queued behind the instruction fetch. Additionally, if a data operand was also required, and it was resident in the data cache, it could also be accessed without hindering either the instruction access from its cache or the peripheral access external to the chip. The parallelism designed into the MC68030 also allows multiple instructions to execute concurrently so that several internal instructions (those that do not require any external accesses) could execute while the processor is performing an external access for a previous instruction. The end result is that an MC68030 operating out of on-chip cache offers much better than “no wait state” performance regardless of the external memory system.

**Instruction Cache**

The instruction cache resident on the MC68030 is a 256-byte direct mapped cache organized as 16 blocks consisting of four long words per block. Each long word is independently accessible yielding 64 possible entries, with A1 selecting the correct word during an access. Thus each block has a tag field made up of the upper 24 address bits, the FC2 (supervisor/user) value, four valid bits (one for each long word entry), and the four long word entries (see Figure 3). The instruction cache is automatically filled by the MC68030 whenever a cache miss occurs and using the burst transfer capability, up to four long words can be filled in one burst. Neither the instruction or data caches can be manipulated directly by the programmer except by the use of the CACR register which provides cache clearing and cache entry clearing facilities. (The caches can also be independently enabled/disabled through the use of this register.) Finally, the system hardware can disable the on-chip caches at any time by the assertion of the CDIS signal.

**Data Cache**

The organization of the data cache is similar to that of the instruction cache as shown in Figure 4. However, the tag is composed of the upper 24 address bits, the four valid bits, and all three function code bits, explicitly specifying the address space associated with each block. The data cache employs a write-through policy with no write allocation for data writes. In other words, if a cache hit occurs on a write cycle, both the data cache and the external device are updated with the new data. If a write cycle generates a miss in the data cache, only the external device is updated and no data cache entry is replaced or allocated for that address.

![Figure 3—MC68030 on-chip instruction cache organization](image-url)
OPERAND TRANSFER MECHANISMS

The MC68030 offers three different mechanisms by which data can be transferred into and out of the chip.

1) Asynchronous bus cycles, compatible with the asynchronous bus on the MC68020, can transfer data in a minimum of three clock cycles and the amount of data transferred on each cycle is determined by the dynamic bus sizing mechanism on a cycle by cycle basis with the DSACKx signals.

2) Synchronous bus cycles are terminated with the assertion of STERM (Synchronous Termination) signal and always transfer 32-bits of data in a minimum of two clock cycles, increasing the bus bandwidth available for other bus masters, therefore increasing possible performance.

3) Burst mode transfers can be used to fill blocks of the instruction and data caches when the MC68030 asserts CBREQ (Cache Burst Request). After completing the first cycle with STERM, subsequent cycles may accept data on every clock where STERM is asserted until the burst is completed. Use of this mode can further increase the available bus bandwidth in systems that use DRAMs with page, nibble, or static column mode operation or SRAM configurations.

Asynchronous Transfers

Though the MC68030 has a full 32-bit data bus, it offers the ability to automatically and dynamically downsize its bus to 8 or 16 bits if peripheral devices are unable to accommodate the entire 32 bits. This feature allows the programmer the ability to write code that is not bus-width specific. For example, long word (32 bit) accesses to peripherals may be used in the code, yet the MC68030 will transfer only the amount of data that the peripheral can manage at one time. This feature allows the peripheral to define its port size as 8, 16, or 32 bits wide and the MC68030 will dynamically size the data transfer accordingly, using multiple bus cycles when necessary. Hence, programmers are not required to program for each device port size or know the specific port size before coding; hardware designers have flexibility to choose implementations independent of software prejudices. The MC68030, like the MC68020, offers a complete dynamic bus sizing mechanism which allows 8 or 16 bit ports or memory to be used without wasting any address space.

The dynamic bus sizing is invoked with the use of the DSACKx pins and occurs on a cycle by cycle basis. For example, if the processor is executing an instruction that requires the reading of a long word operand, it will attempt to read 32 bits during the first bus cycle to a long word address boundary. If the port responds that it is 32 bits wide, the MC68030 latches all 32 bits of data and continues. If the port responds that it is 16 bits wide, the MC68030 latches the 16 valid bits of data and runs another cycle to obtain the other 16 bits of data. An 8 bit port is handled similarly but with four bus read cycles. Each port is fixed in assignment to particular sections of the data bus. However, the MC68030 has no restrictions concerning the alignment of operands in memory; long word operands need not be aligned to long word address boundaries. When misaligned data requires multiple bus cycles, the MC68030 automatically runs the minimum number of bus cycles. Instructions must still be aligned to word boundaries.

The timing of asynchronous bus cycles is also determined by the assertion of the DSACKx signals on a cycle-by-cycle basis. If the DSACKx signals are valid 1.5 clocks after the beginning of the bus cycle (with the appropriate setup time), the cycle terminates in its minimum amount of time corresponding to three clock cycles total. The cycle can be lengthened by delaying DSACKx (effectively inserting wait states in one clock increments) until the device being accessed is able to terminate the cycle. This flexibility gives the processor the ability to communicate with devices of varying speeds while operating at the fastest rate possible for each device.

Use of the asynchronous transfer mechanism allows external errors to abort cycle upon the assertion of BERR (Bus Error), or individual bus cycles to be retried with the simultaneous assertion of BEFF and HALT, after the DSACKx signals have been asserted.

Synchronous Transfers

Synchronous bus cycles are terminated with the assertion of the STERM signal which automatically indicates that the bus transfer is for a 32 bit port. This input is not synchronized internally, thereby allowing two clock cycle bus accesses to be performed, if the signal is valid, one clock after the beginning of the bus cycle with the appropriate setup time. However, the bus cycle may be lengthened by delaying STERM (inserting wait states in one clock increments) until the device being...
accessed is able to terminate the cycle as in the case of asynchronous transfers. Additionally, these cycles may be aborted upon the assertion of BERR, or they may be retried with the simultaneous assertion of BERR and HALT, after the assertion of STERM. For systems operating at high clock frequencies STERM is easier to use than DSACK, because while DSACKx is asserted to the processor 1.5 clocks before the end of a bus cycle, STERM is asserted only one clock before the end of bus cycle (non-burst). This extra half clock allows control logic to be that much slower (or the clock frequency to be that much faster).

Burst Read Cycles

The MC68030 provides support for burst filling of its on-chip instruction and data caches, adding to the overall system performance. The on-chip caches are organized with a block size of four long words, so that there is only one tag for the four long words in a block. Since locality of reference is present to some degree in most programs, filling of all four entries when a single entry misses can be advantageous, especially if the time spent filling the additional entries is minimal. When the caches are burst-filled, data can be latched by the processor in as little as one clock for each 32 bits.

Burst read cycles can be performed when the MC68030 requests them with the assertion of CBREQ and only when the first cycle is a synchronous cycle as described above. If the CBACK (Cache Burst Acknowledge) input is valid at the appropriate time in the synchronous bus cycle, the processor will keep the original AS, DS, R/W, address, function code and size outputs asserted and will latch 32 bits from the data bus at the end of each subsequent clock cycle that has STERM asserted. This procedure continues until the burst is complete (the entire block has been transferred), BERR is asserted in lieu of STERM, or the CBACK input is negated. Figure 5 shows the minimum MC68030 burst cycle. To support slower memory systems, any number of wait states can be inserted before a unique long word of data is latched by negating STERM with the proper setup time. When compared to an MC68020/MC68851 system, the MC68030's burst mode provides a 220% (3.2x) improvement in the data transfer rate, assuming a four clock physical bus cycle (one long word of data, four bytes) for the 020/851 pair and a five clock burst cycle (four long words, 16 bytes) on the MC68030. This five clock burst cycle provides an effective transfer rate of one 32-bit long word every 1.25 clocks. Hence, performance of the MC68030 is better than “no wait states.” In addition, the MC68030 loads this data into its on-chip caches where it is needed again no external cycle may be required at all, improving further the better than “no wait state” performance.

Implementing an external memory system with burst capability presents several architectural options. DRAMs with page, nibble, or static column modes can be easily matched to the MC68030’s burst cycle. For higher performance systems, SRAM banks can be configured in 32, 64, or 128 bit widths. For banks that are 32 bits wide an external modulo-2 counter is required to provide the addresses of the three long words not directly addressed by the MC68030’s address lines. If the memory width is 128-bits, no additional addressing is necessary, but a counter will serve as a multiplexor control to gate the proper 32-bit long word onto the data bus during the burst. The 64 bit wide memory is a compromise between the two previous configurations, in that some additional addressing and multiplexor logic is required.

EXCEPTIONS

Exceptions can be generated by either internal or external causes. The externally generated exceptions are the interrupts, the bus error, and reset requests. The interrupts are requests from peripheral devices for processor action while the bus error and reset pins are used for access control and processor restart. The internally generated exceptions come from instructions, address errors, tracing, or breakpoints. The TRAP, TRAPcc, TRAPV, cpTRAPcc, CHK, CHK2, and DIV instructions can all generate exceptions as part of their instruction execution. Tracing behaves like a very high priority, internally generated interrupt whenever it is processed. The other internally generated exceptions are caused by illegal instructions, instruction fetches from odd addresses, and privilege violations. Finally, the MMU can generate exceptions when it detects an invalid translation in the Address Translation Cache (ATC) and an access to the corresponding address is attempted, or when it is unable to locate a valid translation for an address in the translation tables.

Bus exceptions are an important class of the possible M68000 exceptions. Included in this group are the bus error and retry operations, which are absent from many architectures outside the M68000 Family. Bus error exceptions for example, could be used to immediately indicate a parity error on data in the current cycle. This permits faster error detection and recovery, and prevents the processor from executing on bad data.

The retry mechanism can be used with external caches to repeat a particular bus cycle. With caches having high hit rates, this technique allows cache control logic to be slower than might be expected because hits are always assumed and signaled as such to the microprocessor by STERM or DSACK. If a miss does occur, a retry can be signaled within half a clock after STERM or one clock after DSACK. Thus with a 20 MHz MC68030, cache control logic must only be 5 ns faster than the cache data memory.
MC68030 ON-CHIP MEMORY MANAGEMENT UNIT

The full addressing range of the MC68030 is 4 gigabytes (4,294,967,296 bytes). However, most MC68030 systems implement a smaller physical memory. Nonetheless, by using virtual memory techniques, the system can be made to appear to have a full 4 gigabytes of physical memory available to each user program. In a similar fashion, a virtual system provides user programs access to other devices that are not physically present in the system such as tape drives, disk drives, printers, or terminals. The memory management unit (MMU) on the MC68030 provides the capability to easily support a virtual system and virtual memory. In addition, it provides protection of supervisor areas from accesses by user programs and also provides write protection on a page basis. All this capability is provided along with maximum performance as address translations occur in parallel with other processor activities.

Demand Paged Implementation

A typical system with a large addressing range such as one with the MC68030 provides a limited amount of high-speed physical memory that can be accessed directly by the processor while maintaining an image of a much larger "virtual" memory on secondary storage devices such as large capacity disk drives. When the processor attempts to access a location in the virtual memory map that is not resident in physical memory, the access to that location is temporarily suspended while the necessary data is fetched from secondary storage and placed in physical memory; the suspended access is then either restarted or continued.

A paged system is one in which the physical memory is subdivided into equal sized blocks called page frames and the logical (untranslated) address space of a task is divided into pages which have the same size as the page frames. The operating system controls the allocation of pages to page frames so that when data is needed from the secondary storage device, it is brought in on a page basis. The memory management scheme employed by the MC68030 is called a "demand" implementation because a process does not need to specify in advance what areas of its logical address space it requires. An access to a logical address is interpreted by the system as a request for the corresponding page.

The memory management unit on the MC68030 employs the same address translation mechanism introduced by the MC68851 Paged Memory Management Unit with possible page sizes ranging from 256 bytes to 32K bytes.

Translation Mechanism

Logical-to-physical address translation is the most frequently executed operation of the MC68030 MMU, so this task has been optimized and can function autonomously. The MMU initiates address translation by searching for a descriptor with the address translation information (a page descriptor) in the on-chip ATC. The ATC is a very fast fully-associative cache memory that stores recently used page descriptors. If the descriptor does not reside in the ATC then the MMU requests external bus cycles of the bus controller to search the translation tables in physical memory. After being located, the page descriptor is loaded into the ATC and the address is correctly translated for the access, provided no exception conditions are encountered.

The status of the page in question is easily maintained in the translation tables. When a page must be brought in from a secondary storage device, the table entry can signal that this descriptor is invalid so that the table search results in an invalid descriptor being loaded into the ATC. In this way, the access to the page is aborted and the processor initiates bus error exception processing for this address. The operating system can then control the allocation of a new page in physical memory and can load the page all within the bus error handling routine.

Address Translation Cache

An integral part of the translation function described above is the cache memory that stores recently used logical-to-physical address translation information, or page descriptors. This cache consists of 22 entries and is fully-associative. The ATC compares the logical address and function code of the incoming access against its entries. If one of the entries matches, there is a hit and the ATC sends the physical address to the bus controller, which then starts the external bus cycle (provided there was no hit in the instruction or data caches for the access).

The ATC is composed of three major components: the content-addressable memory (CAM) containing the logical address and function code information to be compared against incoming logical addresses, the physical address store that contains the physical address associated with a particular CAM entry, and the control section containing the entry replacement circuitry that implements the replacement algorithm (a variation of the least-recently-used algorithm).

Translation Tables

The translation tables supported by the MC68030 have a tree structure, minimizing the amount of memory necessary to set up the tables for most programs, since only a portion of the complete tree needs to exist at any one time. The root of a translation table tree is pointed to by one of two root pointer registers that are part of the MC68030 programmer's model; the CPU and supervisor. Table entries at the higher levels of the tree (pointer tables) contain pointers to other tables. Entries at the leaf level (page tables) contain page descriptors. The mechanism for performing table searches uses portions of the logical address as indices for each level of the lookup. All addresses contained in the translation table entries are physical addresses.

Figure 6 illustrates the structure of the MC68030 translation tables. Several determinants of the detailed table structure are software selectable. The first level of lookup in the table normally uses the function codes as an index but this may be suppressed if desired. In addition, up to 15 of the logical address lines can be ignored for the purposes of the table
searching. The number of levels in the table indexed by the logical address can be set from one to four, and up to 15 logical address bits can be used as an index at each level. A major advantage to using this tree structure for the translation tables is the ability to deallocate large portions of the logical address space with a single entry at the higher levels of the tree. Additionally, portions of the tree itself may reside on a secondary storage device or may not exist at all until they are required by the system.

The entries in the translation tables contain status information pertaining to the pointers for the next level of lookup or the pages themselves. These bits can be used to designate certain pages or blocks of pages as supervisor-only, write-protected, or non-cacheable. If a page is marked as non-cacheable, accesses within the page will not be cached by the MC68030 instruction or data caches and the CIOUT (cache inhibit out) signal is asserted for those accesses. In addition, the MMU automatically maintains history information for the pages and in the descriptors via the Used (U) and Modified (M) bits. CIOUT is particularly useful in systems accessing shared memory or I/O devices. CIOUT directly informs any external cache that current data should not be cached. Historically, some microprocessor architects recommended dedicating an address line to indicate non-cacheable areas, but the CIOUT signal provides more flexibility without affecting the processor’s addressing range.

Transparent Translation

Two transparent translation registers have been provided on the MC68030 MMU to allow portions of the logical address space to be transparently mapped and accessed without corresponding entries resident in the ATC. Each register can be used to define a range of logical addresses from 16M bytes to 4G bytes with a base address and a mask. All addresses within these ranges will not be mapped and protection is provided only on a basis of read/write and function code. These registers provide windows into memory that will never suffer from page faults regardless of previous memory activity. For example in a real time graphics application, this means that line drawing will be continuous and not jerk or step from delays caused by page faults and table searches. Many other applications will also benefit from the TT registers.

COPROCESSOR INTERFACE

The coprocessor interface is a mechanism for extending the instruction set of the M68000 family. The interface provided on the MC68030 is the same as that on the MC68020. Examples of these extensions are the addition of specialized data operands for the existing data types or, for the case of floating point, the inclusion of new data types and operations for them as implemented by the MC68881 and MC68882 floating-point coprocessors.

The communication protocol between the main processor and the coprocessor necessary to execute a coprocessor instruction is based on a group of coprocessor interface registers (CIRs) which have been defined for the M68000 family and are implemented on the coprocessor. The MC68030 incorporates the CIR protocol in hardware: the bus interface implemented by a coprocessor for its interface register set must only satisfy the MC68030 address, data, and control signal timing to guarantee proper communication with the CPU. The MC68030 implements the communication protocol with all coprocessors in hardware (and microcode) and handles all operations automatically so the programmer is only concerned with the instructions and data types provided by the coprocessor as extensions to the MC68030 instruction set and data types. Up to seven coprocessors are supported in a single MC68030 system with a system-unique coprocessor identifier encoded in the coprocessor instruction. When accessing a coprocessor, the MC68030 executes standard bus cycles in CPU address space, as encoded by the function codes, and places the coprocessor identifier on the address bus to be used by chip-select logic to select the particular coprocessor.

SUMMARY

The MC68030 provides increased system performance and reduced system costs through enhanced features not found on any other commercial microprocessors. Increased performance is derived from the MC68030’s on-chip caches, the on-chip memory management unit, multiple internal address and data buses, and a versatile bus controller. Reduced system cost is achieved by bringing all these features on-chip and through the improved bus controller’s interface to external memory. The MC68030 does indeed represent the second generation in 32-bit microprocessors.

ACKNOWLEDGEMENTS

The author wishes to thank Clara Serrano of Motorola’s M68000 Applications Group for her assistance in the preparation of this paper.