Real-Time Computing: Implications for General Microprocessors

Chip Weems Steve Dropsho

University of Massachusetts

In this document we describe the issues facing hardware design for general-purpose real-time processors. This document is divided into three parts, of which the first is a summary of the material in the second and third sections. The second section discusses performance features of current processors that make them unsuitable for real-time computing and proposes alternative features to achieve high performance while supporting real-time needs. The third section discusses the merits of using intermediate level vision as the driving application for the design of a true real-time processor.

Summary

The demands of real-time computing with hard deadlines are not well met by today's microprocessor-based systems. The problem arises from design choices to maximize *average* performance in current microprocessors and tolerate high latency on low probability events (such as TLB misses). In contrast, real-time applications must be able to ensure upper bounds on execution times. Thus, a large potential variance in the performance of the processor can force the upper bound to be so high that the throughput of the system is strangled as a result of the weak performance guarantees. Thus, real-time systems need fast hardware with worst case timings closely matching the average case.

Variance is probably inevitable in any software system due to paging, interrupts, data dependent branching, etc. The key to designing for real-time is to provide the programmer with the capability of controlling points in the system that contribute to variance in execution times. Having this control allows the programmer to account for the variance with respect to the real-time operating system or application.

There are many areas in current processor designs where variance is added. Conditional branches have different execution times depending on whether or not the correct path is predicted. Instructions with execution times that are data dependent add variance between runs on different data sets. Asynchronous coprocessors have variable communication delays. Interrupts change the control flow of the program usually by asynchronous events. Fast address translation suffers from variance due to translation cache (TLB) misses. Fast instruction and data accesses through caching adds variance for similar reasons. Using intermediate storage buffers to decouple writes from time-critical reads adds uncertainty during execution when coherency conflicts force buffer flushes.

The solutions to these issues involve either removing the source of variance or giving the programmer control to explicitly manage the offending resource. For example, branch penalties are eliminated with additional hardware to track both possible outcomes, while uncertainty due to intermediate storage buffers, data dependent instructions, and asynchronous coprocessors is solved by simply not allowing them. Sometimes removing a feature may have an unacceptable performance penalty, this is the case with caches. The variance due to caching is best served by letting the user explictly manage the resources so misses are not taken or they can be planned. This serves two purposes: it provides the performance enhancement of the feature while letting the user minimize variance in its operation.

No current processor satisfactorily addresses all these issues. We recommend that a new processor be designed to address the problems of hardware support for real-time systems. This processor must be efficient in the many real-time application domains. To ensure its versatility, we propose using intermediate-level vision as a driving application for the processor design. Intermediate-level vision relies on a core set of diverse techniques that are used in many other fields: Kalman filters, singular value decomposition, database operations, object programming, graphics, neural nets, dynamic programming, and multiple models of parallelism such as shared memory and message passing. These techniques define a set of general capabilities that a real-time processor must support.

At the end of this report many of the common processors available today are surveyed with respect to how well each satisfies real-time computing requirements. A table at the end condenses these summaries into a quick reference form for easy comparisons.

Real-time Computing

Today's microprocessors are inappropriate for complex real-time applications because their designs strive to improve the *average* performance with technology whose benefits are probabilistic, thus increasing the variance in execution time predictability or making reasonable timing predictions impossible altogether.

A major issue in real-time (RT) computing is the predictability of the execution time of an application. System factors that make predictability difficult or impossible are: branch instructions, asynchronous coprocessors, variable length instructions, interrupts, the degree of flexibility of the operating system, and the memory system.

Branch instructions cause variance in execution time due to pipelines in the processors and the need to predict the outcome of a conditional branch many cycles before the result of a test is known. If an incorrect prediction is made then a processor must flush its pipeline and restore its state to that at the point where the branch was taken. Depending on the depth and complexity of the pipeline this penalty for an incorrect branch can be significant. In [1] Hennessy and Patterson show that conditional branches are 11%-17% of the instructions. In a worst case analysis, such as that employed in a real-time system, this penalty must be assumed on every branch, thus each cycle of the potential penalty adds significant variance to the prediction of the application execution time. The variance can be eliminated by duplicating early stages in the pipeline as proposed in the Y-Pipe design[2].

The problems of variable length instructions and asynchronous processors are easily avoided. Variable length instructions add variance because their operation is data dependent. In today's RISC instruction sets only shift-type instructions fall into this category. If they are implemented with a full barrel shifter then the operations take constant time and are of no concern. CISC processors are more prone to this problem, sometimes implementing complete procedure calls within a single machine instruction.

Asynchronous coprocessors create the problem of imprecise interrupt timing during an exception which adds to the variance of exception handling. A simple solution is to make all coprocessors synchronous even if there is a performance penalty in doing so. Out-of-order execution should also be avoided for a similar reason because undoing operations after an exception may require a variable amount of time dependent on the combination of operations in the pipeline.

To limit variance due to interrupts, low overhead interrupt processing and context switch hardware are needed if real-time applications share a processor with the system software. This may be a less critical issue if the real-time system adopts an architecture like the Spring Architecture[3], where the system has both application processors and a system processor. The system processor is responsible for all interrupts allowing the application to run in a more predictable environment on the application processor. Of course, this is an expensive solution. It also implies that the processor must support parallel configurations. For less costly systems a low-overhead interrupt capability or possibly an interrupt coprocessor will be necessary.

The Spring Architecture and ART, a Mach-based system, are two powerful RT operating systems which overcome restrictions of simpler, more rigid embedded RT kernels. A major feature these operating systems provide is support for logical addressing compared with strict physical addressing of embedded systems. This capability allows easy implementation of process protection which is critical in a multi-tasking environment. The drawback is that address translation must be performed efficiently, but the standard methods in current microprocessors using TLBs have high variance (due to TLB misses). Solutions to this problem involve the memory hierarchy and are discussed below.

Probably the largest source of variance comes from modern memory hierarchies. In order of decreasing severity these sources are: page faults, TLB misses, and cache misses. Together they force an unmanageable amount of variance in a memory access. The current solutions are brute force. Expensive page faults are prevented by loading the entire application and system code into physical memory before execution. TLB misses are prevented by loading the complete address mapping of application and system code before execution. Cache misses can be excluded by bypassing the cache, but this has an unacceptable performance penalty. Current RT operating system research analyzes the assembly code to determine lower bounds on the cache hit rate to better estimate the worst case execution time.

The need to prevent TLB misses and predict cache hit rates has implications on the processor hardware. The TLB must be able to be read and written directly by the system software. Since an entire application must be mapped into a limited TLB, the TLB must be flexible in the page size that each entry can map. Research indicates that the page size should be allowed to range from as small as 2Kb to as large as 16Mb and have a relatively small increment size, a factor of two appears acceptable (e.g., 2Kb, 4Kb, 8Kb,...).

Analysis of cache hit rate behavior is done on the static code only if a loop fits entirely within the cache, including subroutines called. Although it is primarily the responsibility of the compiler to place related code into memory regions that do not overlap in the cache, certain hardware features can make the analysis easier. One feature is to have a virtually addressed cache since this is the address space the compiler has access to. This is common in today's first-level caches but not in the increasingly necessary secondary caches which use physical addresses. To gain a predictable performance advantage from physically addressed second level caches the OS must be able to intelligently allocate memory with hints from the compiler. Without such OS involvement second-level caches lose their advantage, indicating that either large primary caches will be necessary or the use of virtual secondary caches. Virtual secondary caches, however, impact the address translation mechanism by requiring translation at that level of the hierarchy. Another feature of RT systems is that caches are flushed during context switches so a fast flushing mechanism is mandatory for high performance.

There are various other features to consider. One is intermediate storage such as writebuffers. This type of storage delays operations in order to minimize read access times. However, limited buffer sizes and the need to maintain consistency of the data in the system force periodic *unanticipated* delays to memory accesses due to emptying the contents of buffers. Buffers added for performance purposes must have their value reassessed for a real-time environment. Another feature is the ability to synchronize a processor to external events to minimize the contention for resources and, hence, variance in execution time (e.g., contention for memory cycles between the processor and the DRAM refresh circuitry).

Finally, to help analyze the hardware system, very high speed counters should be available to the software that are accurate to within a single cycle. Many systems place high-speed clocks on a remote device so a multiprocessor system will have a consistent view of the global counter. In real-time systems, processes do not generally migrate and the developers need a high-speed counter locally on every processor ideally with an access time of a single cycle. Other features such as an external time-boundary warning signal may be worth considering. This feature could be implemented with a count-down counter that provides an external signal when a preset time boundary is reached. This signal can then be fed to an external processor or returned as an interrupt to the application processor to notify the system software that the application has passed a soft-deadline and corrective action may be necessary in the near future.

As a general rule, the more predictable each piece of the system is the easier it is to predict a system's overall behavior. To recap, the desired features in a real-time system include:

- Support to minimize time penalties due to incorrect branching.
- Fixed length instructions.
- Synchronous coprocessors and in-order instruction execution only.
- Low-overhead interrupts and context switching.
- Logical addressing to support process protection.
- A TLB with flexible page sizes that can be read and written directly by software and quickly flushed.
- Support for large primary caches and/or virtual secondary caches that can be flushed quickly.
- No hidden buffering that affects memory access time.
- Capability of the processor to tightly synchronize to external events.
- A high-speed local clock on the processor with a very small access time (1 cycle).

Vision Processing

A processor for complex real-time computing must be efficient in a variety of processing domains. Intermediate-level vision is an ideal driving application for the design of such a processor. The most compelling applications for vision, such as vehicle control, have strict real-time constraints and, thus, require real-time support. These applications incorporate a diverse set of computational characteristics, elements of which are shared in many standard non-vision applications. In addition, the larger vision applications need parallel systems to satisfy their tremendous computational and IO demands, so common models of parallelism have to be supported efficiently. By designing to the requirements of intermediate-level vision, a powerful real-time processor will result that will be ideal for small, inexpensive real-time systems with minimal support logic. And, with support for various models of parallelism, the same processor can be the building block for larger multiprocessor systems that need self-contained processing units for compact and reliable designs.

We have identified a set of generic high-level operations in intermediate-level vision which define the common capabilities that should be supported efficiently. These operations are: error minimization, database manipulations, projection, and functional approximation. These operations can be considered general categories of problems to which specific techniques are applied. These techniques did not originate in vision, but were adopted from other areas: scientific numerical methods, database methods, graphics processing, and areas of AI, specifically machine learning. A processor designed to be versatile should be efficient in all these domains of processing.

Error minimization is an integral component in vision algorithms as in most other scientific computing. The inherently noisy environment in which vision data is gathered makes exact results impossible in many vision tasks such as model matching or depth from motion. Two common techniques used in error minimization are Kalman filters and singular value decompostion (SVD). Both of these techniques rely heavily on the matrix operations of addition, multiplication, inverse, and transpose. High precision is required, especially in SVD, so double precision operations should be fast. The matrices involved range from small 4x4 matrices (in 3D translations) to larger matrices on the order of 1000x1000 (more general linear equation manipulations). The implications on hardware design is that traditional high-precision scientific computation must be supported, however, the computational units should efficiently support small data sets. Decoupled integer and floating point units should be incorporated to allow simultaneous calculation of array indices with floating point results. Vector processing capability will be useful if the start up overhead is small.

Database capabilities are part of vision *systems* that incorporate and orchestrate many vision algorithms in order to perform higher level vision. These systems must organize the data elements (e.g., line segments, regions, structures, etc.), called tokens and usually encapsulated in objects, that are generated and consumed by the various algorithms. Current work in this area allows for tokens to be gathered into a database, or token set, with support for any number of token sets in a system. The structure of the sets are independent of each other to allow each set to be optimized for its particular purpose. The sets are structured as arrays of tokens or lists of tokens. While the array structure provides a very efficient access method the list structure is often used because of its generality. Token sizes range from very small (representation of a line segment) to very large (an entire image plane). The token sets also vary considerably in their sizes from a few elements to potentially thousands. The basic operations on token sets are set operations. Token sets are used to communicate within the multi-tasking vision system and have been implemented in both a shared memory model and a message passing model. Minimal information is kept with the token to reduce memory requirements; thus, complex properties tend to be generated when needed. This type of data organization requires efficient Lisp-like processing to handle the objects, meaning very low overhead context switching for frequent (member) function calls. Also required are low latency memory access times to retrieve widely dispersed information (having a relatively low locality of reference profile which reduces cache efficiency). And for communication, hardware support for low latency communication to allow efficient processing of small token sets in a scalable distributed muliprocessing environment; and high bandwidth communication support for large token sets.

Just as user interfaces and other applications are increasing their use of graphics, so are developers of vision applications relying more and more heavily on basic graphics capabilities. In vision systems,

graphics are used in the user interface and also to do projection, generation of a 2D image from a 3D model. Projection is used in two respects. One, the important task of model matching transforms hypothesized models into proper orientations and then performs hidden line removal before matching actual tokens to the model. And two, an image generated through graphics in many cases is the goal of the processing in order to aid humans in areas such as medical treatment. Real-time vision processing will require real-time generation of graphics. The graphics may be generated elsewhere if there is the capability to move large amounts of data rapidly out of the system, or the graphics may be generated locally if the task is relatively small and there is sufficient local graphics support. Today the level of sophistication of the graphics used is relatively low with hidden line removal being one of the most common operations. Taking hidden line removal as a starting point to define local graphics needs, support for fast z-buffering using ray tracing must be included. However, in some instances researchers are finding that the limited accuracy of z-buffering (precision is at pixel granularity) is insufficient and they are forced to use the much more computationally intensive line intersection method. If this is an indication of the general trend in graphics requirements, applications will be demanding support for more sophisticated graphics operations in the future.

Functional approximation is becoming an important method in the development of vision algorithms and other areas of AI research. Across AI, while the algorithms are improving, the degrees of freedom for manipulating parameters to optimize performance (in terms of accuracy) is increasing drastically. Functional approximation in the form of supervised learning is one method to manage this explosion of parameter dimensions. Learning techniques using backpropagation networks, temporal difference (TD) methods, instance-based recognition, and traditional dynamic programming are all being applied. These involve medium-precision floating point operations (mostly requiring only scaled integer precision), table lookups, and low latency communication of small data packets (for distributed backpropagation networks). Large backpropagation networks place interesting demands on network topology and consequently the processor to interface efficiently with the desired topologies.

To summarize the above computation and communication requirements, they are:

- Efficient high precision floating point operations.
- Efficient vector processing for both small and large arrays.
- Fast context switching for frequent function calls such as found in object oriented programming.
- Low latency memory accesses on cache misses.
- Low latency, high bandwidth communication support for scalable, distributed computing.
- Basic graphics support with some consideration towards more complex graphics operations.

Conclusion

Efficient real-time processing can only be attained on hardware that is designed to satisfy its unique need for low variance on all fundamental operations in the system. Current high performance processors have not been designed with real-time processing as a goal and thus are not a solution. While the weaknesses in these processors are understood and solutions have been proposed, a satisfactory processor design which accumulates this work has not been developed. We recommend that this research be brought together in a new processor design that emphasizes extremely low variance in execution time. To ensure the processor will be efficient for the many applications in real-time computing, intermediate-level vision should provide the benchmark environment. The occurrence of intermediate-level vision's fundamental techniques in so many other important fields make it an ideal benchmark for processor design.

A summary of current microprocessors follows. A table at the end condenses this information.

Alpha 21064-AA[4, 5, 6]

Noteworthy instructions. *Privileged architecture code (PALcode)* are instructions that match specific operating system functions to the underlying hardware and abstract architectural features from implementation specifics. Their basic function is to emulate CISC instructions on a RISC machine to ease operating system ports. Context switches as well as access to privileged registers are handled in PALcode. Upon a call to a PALcode routine the pipeline is flushed and the current PC saved, hence PALcode instructions for small tasks, such as loading a TLB register, are costly.

All *arithmetic exceptions* cause asynchronous traps. This means that additional instructions may have executed after the exception occurred (possibly generating additional exceptions). Software can make these exceptions precise via the trap barrrier instruction, TRAPB, which prevents subsequent instructions from being issued during the window that the current instruction might trap. In [6]. McLellan states that ensuring precise exceptions at all times resulted in less than one percent degradation in integer performance and between 3 and 25 percent in floating point performance. Other processors, such as Motorola's 88110, have non-IEEE compliant modes in which reasonable values are returned (e.g., zero if underflow occurs) instead of generating a trap.

Incorrectly predicted *branches* result in a 4 cycle penalty. The high penalty is due to the superpiplining employed. The pipeline depth is 7 stages for non-floating point instructions and 10 stages for floating point.

Coprocessor Support. Only through shared memory multiprocessor model and interprocessor interrupt capability.

Interrupts and context switches. Interrupts are implemented through PALcode. Before an interrupt can be serviced all currently executing instructions must finish without exception and all outstanding data cache fills must complete; if an exception arises it will be serviced before the interrupt. Thus, interrupts have high potential variance.

Swap Privileged Context instruction (SWPCTX) is a PALcode instruction that swaps out the current Hardware Privileged Context Block (HWPCB) and swaps in a new one during a context switch. In the HWPCB 72 bytes are saved/restored as well as the 31 integer and 31 floating point registers, for a total transfer of 2x568 bytes or 1136 bytes per context switch.

Addressing. The instruction TLB has 12 entries, 8 of which map 8KB pages and 4 that map 8KB, 64KB, 512KB, or 4MB pages. The data TLB has 32 entries to map pages of 8KB, 64KB, 512KB, or 4MB. This is a wide range but considered too coarse. The TLB entries can be directly read/written but only through PALcode (and its high overhead relative to a single register write). PALcode can quickly flush all or *selective* entries.

Caches and buffers. Supports on-chip virtually addressed first level cache and off-chip physically addressed second level cache. The Harvard architecture primary cache has an 8KB instruction cache and an 8KB write-through or writeback data cache. A 128KB to 16MB second level cache is supported. There is full support for shared memory multiprocessing and PALcode instructions to flush both the I-cache and D-cache.

There is a 4 by 32 entry write buffer to condense stores off-chip. This buffer can be forcibly flushed using the memory barrier (MB) instruction between every store. Its execution time depends on the contents.

Monitoring support. A cycle counter is accessible through PALcode (hence, with relatively high overhead for a single register access). Other counters are available that can be used to track instruction issue, pipeline stalls, instruction mix, cache misses, and branch misprediction. There are also two input pins for counting external events.

Integer unit. The integer register file has 32 general purpose 64-bit registers. The logic unit, the adder, and the shifter can retire one operation per cycle, the multiplier is not pipelined and takes 16 cycles per result. There is no hardware for direct division, but a suggested method has a best case of 1 multiply (16 cycles) and a worst case of 9 multiplies (144 cycles) and is data-dependent on the divisor. The DEC 10000 is rated at SPECint89 of 116.2.

Floating point unit. The floating point register file has 32 entries of 64-bits. The adder and multiplier can retire one instruction per cycle with dependencies adding a 6 cycle latency. The DEC 10000 is rated at SPECfp89 of 275.8 and SPECfp92 in excess of 200.

External data paths. At a bus clock of 75MHz (cpu at 150MHz), the 128-bit bus electrical limit is 1200MB/sec.

Additional pipeline capabilities. Condition codes are sent to any general purpose register rather than a dedicated location on compares. This simplifies saving results of comparisons for later use.

Dual instruction issue capability allows the following combinations: an integer with a floating point operation; a load or store with an integer or float operation; and a branch with a load, store, integer, or float operation. Notice that only a subset of the possible combinations of functional units are supported. McLellan states this results in only a 2 percent degradation in performance over a more flexible (and more complex) design. The Alpha cannot cascade functional units for multiply-accumulate operations.

Vector processing is possible due to the separate address unit, so arithmetic calculations (int and float) can occur simultaneously with operand fetches. However, loop unrolling is necessary on small loops to amortize the cycle of overhead of the (correctly predicted) branch; this increases code size. Also, branches cannot issue with stores of the same format (integer/integer or float/float) because they share a register file port. Thus, a loop of integer arithmetic using an integer counter for the branch control cannot pair the branch with an integer operation and will likely have to issue the branch alone.

Graphics support. None.

Communication support. No message passing support for distributed memory systems.

Alpha 21164[7]

Noteworthy instructions. Insufficient information.

Coprocessor Support. Insufficient information.

Interrupts and context switches. Insufficient information.

Addressing. Insufficient information.

Caches and buffers. 8 KB instruction cache and 8 KB data cache, both direct mapped. Sports a 96 KB *on-chip* second level unified cache that is three-way set-associative. The processor has support for a third level of caching.

Monitoring support. Insufficient information.

Integer unit. SPECint92 of 330.

Floating point unit. SPECfp92 of 500.

External data paths. The bus is 128-bits wide and can be run at frequencies of the system clock divided by an integral value between 1 and 15.

Additional pipeline capabilities. Superscalar pipeline can issue up to four instructions per cycle- two integer and two floating point. Maintains in-order execution so does need complex register renaming. The pipeline is 7 stages for integer operations and 9 for floating point.

Graphics support. None.

Communication support. No message passing support for distributed memory systems.

Technology. Packaged in a 499-pin ceramic PGA and fabricated in 0.5 micron CMOS technology with 9.3 million transistors.

Pentium[8]

Noteworthy instructions. Branch penalties are at least 3 cycles. It is not clear what is the maximum penalty.

The safe instruction recognition capability adds variance to floating point operations. Since in order execution is maintained as well as precise interrupts, floating point calculations that may cause an exception force the pipe to stall for 3 cycles until the exception status of the operation is known. However, to speed execution, if the operation can be dynamically determined to be "safe" (i.e., cannot cause an exception) then the pipe does not have to be stalled. Logic in the Pentium detects conditions early for the six possible exceptions: invalid operation, divide by zero, denormal operand, overflow, underflow, and inexact. Obviously, the safe status of an operation is data dependent, so the execution time of a given operation may not be known a priori. The last three exceptions are not detected in the safe instruction logic because they require floating point operations, but the software has the ability to not take the exception by setting the precision mask bit. Most processors have this problem with arithmetic exceptions but do not add detection logic up front, instead logic is added to clean up the state of the processor after an exception so that it appear precise.

The Pentium supports the 486's CISC instruction set.

Coprocessor Support. Insufficient information.

Interrupts and context switches. Insufficient information.

Addressing. The reference has insufficient information on the organization of the TLB.

Caches and buffers. The Harvard architecture has an 8KB I-cache and an 8KB D-cache. The D-cache uses a writeback protocol.

Monitoring support. Insufficient information.

Integer unit. The register file has 8 registers. The SPECint92 rating is 64.5.

Floating point unit. The 8 stage floating point pipeline can retire an add or multiply of any precision (single, double, or extended (80 bits)) every cycle. The divider unit calculates two bits of the quotient per cycle; thus, single precision requires 16 cycles and double precision 32 cycles. The SPECfp92 rating is 56.9.

The register file has 8 floating point registers.

External Data paths. At 66MHz the 64-bit system bus has an electrical limit of 528MB/sec.

Additional pipeline capabilities. The pipeline is superscalar and has two integer units, U and V. Floating point operations use the U pipeline during the execution stage, so all units are not totally independent. Floating point operations cannot be paired with other floating point or integer operations, in general. However, the integer units can be used in parallel with the floating point unit to fetch operands for the floating point calculations, resulting in vector processing of floating point operations.

Graphics support. None.

Communication support. Basic multiprocessor protocol support and interrupts.

HP PA7100[9, 10]

Noteworthy instructions. The integer multiply and divide execution times are data dependent, since only minimal hardware support is provided. Lee states that most of these operations as they appear in applications are variables paired with constants; in this case the execution time can be determined during the compilation phase.

The small 5 stage pipeline helps the processor minimize the *branch* penalty to 1 cycle.

Coprocessor Support. Although the Precision Architecture supports coprocessors it is unclear whether the PA7100 implementation provides hardware for these additional coprocessors beyond the floating point unit. In the architecture, two types of assist units are are defined: special functional units (SFUs) and coprocessors. SFUs are tightly coupled to the main processor buses and serve as alternate functional units to the ALU, receiving operands from the processor's general registers and returning results there. Coprocessors are hardware assists with their own registers and interfacing with the processor at the data cache or memory level. The architecture definition allows up to 8 SFUs and 8 coprocessors to be added. The units are synchronous with the main processor.

Interrupts and context switches. All interrupts generated by instructions are synchronous. There are six registers that are continuously updated with the critical processor state so that on an interrupt only the current processor status word has to be saved before control is transferred via a vectored location. This is an extremely low latency handler for interrupts requiring only a small amount of processing (i.e. do not have to store much additional processor state).

During a context switch, to be saved/restored are 128 bytes of control register information, the processor status word, eight space registers, thirty-two 32-bit general purpose registers, and thirty-two 64-bit floating point registers. The total data movement is 2x545 bytes for a total of 1090 bytes.

Addressing. A single TLB is used that has 120 entries for 4KB pages and 12 entries to map pages from 512KB to 64MB. The range is large but the increment is too coarse. The TLB can be purged entry by entry and each entry can be loaded quickly.

Caches and buffers. *External* caches allow sizes of up to 1 MB instruction cache and 2 MB data cache. This is the largest primary cache capability of all the processors in this survey. The data paths to both caches are 64 bits wide. A read is one cycle and a write is two cycles. Instructions exist to both purge and flush the caches. Details on these instructions were not available in the references, but with external caches we surmise that flushing the instruction cache is done entry by entry. The data cache has a writeback policy so the dirty bit of each entry must be tested and the cache line written back if set, before the entry can be invalidated. Due to the large primary caches, there is no logic for second level cache support.

The processor has a single entry write buffer on writes to the data cache. There is a maximum store penalty of 1 cycle (due to the two cycle cache write) which is deterministically avoided by simply not scheduling back-to-back stores in the code.

Monitoring support. Insufficient information.

Integer unit. There are thirty-two 32-bit general registers. The integer unit retires an add every cycle. The references have insufficient information on multiplication and division other than that the operations are multi-cycle and data dependent in their execution times. The processor has a SPECint89 rating of 88.1.

Floating point unit. The processor has been designed with floating point as the critical focus. The register file has 32 entries of 64 bits each. Floating point adds or multiplies can be issued every cycle with data dependencies incurring a 1 cycle penalty. Divides and square roots have hardware support and take 8 cycles for single-precision and 15 cycles for double-precision. Divides and square roots are calculated outside the main pipeline path so additions or multiplications can operate in parallel. The processor has a SPECfp89 rating of 206.2.

Multiply and accumulate operations are supported. These instructions multiply two values and store the result in a register while simultaneously adding two values.

A hardware underflow mode treats denormalized operands as signed zero values and eliminates the need to trap. This does not comply with the IEEE standard. The references do not state methods of managing other arithmetic exceptions, but we assume that they cause synchronous traps.

External Data paths. The data path to main memory is only 32 bits wide for a maximum bandwidth of 400MB/sec.

Additional pipeline capabilities. Floating point vector processing is supported in the dual issue pipeline. An integer operation or a load/store can operate in parallel with a floating point operation. As mentioned above, multiply-accumulate operations are efficiently supported and can be issued with an integer or load/store operation, in effect, issuing 3 operations. It is not clear that branch instructions may be paired with floating point operations to minimize loop overhead.

The pipeline is fully interlocked, so stalls and slips are automatically controlled by the hardware.

Graphics support. Floating point multiply-accumulate function for fast 3D transformations. A 3D transformation requires 12 multiplies and 9 additions, the multiply-accumulate functionality allows this set of operations to be completed in 12 cycles vs. 21 (excluding pipeline latency in both instances).

Graphics support includes instructions for block move, z-buffering, color interpolation, graphics clip test instructions and floating point STORE and LOAD instructions to/from I/O space. Using the CPU alone for graphics, the PA7100 is able to draw 19,400 quadrilaterals/sec.[11].

Communication support. Shared memory model supported. No message passing support provided.

MIPS R4400[12, 13]

Noteworthy instructions. Incorrect *branch* predictions have a 2 cycle penalty.

Coprocessor Support. The architecture has definitions for four coprocessors, C0-C3. C0 is for address translation, C1 is floating point, and C2 and C3 are reserved by MIPS for future use. In effect, there is no support for adding a tightly coupled coprocessor other than through a standard shared memory multiprocessor configuration.

Interrupts and context switches. Insufficient information.

Addressing. The TLB has 48 pairs of entries for a total of 96 entries, however, each pair maps two consecutive virtual pages so the two entries are not independent. The page size range is from 4KB to 16MB with increments coming in a factor of four (4KB, 16KB, 64KB, ... , 16MB). This is the finest grain of increment for such a range in any of the processors in this survey, but [14] suggests an increment factor of only 2 as more useful and more efficient in terms of memory allocation. The TLB entries are directly readable and writeable as special registers. Entries must be invalidated individually, there is no fast flush mechanism.

Caches and buffers. The primary caching scheme is a Harvard architecture with a 16KB I-cache and a 16KB D-cache. The primary caches are virtually addressed, but the second level cache (if added) is physically addressed. The writeback policy of the D-cache prevents a fast flush mechanism, so the flushing execution time depends on the number of dirty entries.

There is a two entry writebuffer to the data cache to buffer writes until a time slot is available when a cache read is not taking place.

The processor has very broad support for implementing most cache coherency protocols on a page by page basis. Complete control logic for the primary and second level caches resides in the processor, however, a bus controller must be implemented with external logic.

Monitoring support. There is an on-chip cycle counter that is quickly readable and writeable as a special register. The counter is incremented every two cycles.

Integer unit. There are 32 general purpose registers of 64 bits each. A 32-bit multiply requires 10 cycles and a 64-bit multiply requires 20. Divide requires 69 cycles for a 32 bit word and 133 cycles for a 64 bit word. A 100MHz R4000 integer unit has a simulated SPECint89 of 55 (4MB second level cache). Linearly scaling this to a 150MHz system implies a rating of 82.5.

Floating point unit. The floating point register file is 32 entries of 64 bits each and two read ports and two write ports. A new floating point add can be issued every 3 cycles for both single- and double-precision values. The floating point unit can accept a new multiply every 3 cycles for single-precision and every 4 cycles for double-precision. A new divide can be issued every 22 cycles for single-precision and 35 cycles for double-precision. The square-root takes 54 and 112 cycles for single- and double-precision values, respectively. A 100MHz R4000 floating point unit has a simulated SPECfp89 of 69 (4MB second level cache). Linearly scaling this to a 150MHz system implies a rating of 103.5.

External Data paths. The path to main memory is 64 bits wide and if run at 75MHz (cpu at 150MHz) provides 600MB/sec. as the electrical limit.

Additional pipeline capabilities. The pipeline is superpipelined with 8 stages, but not superscalar. Only a single instruction can be issued per cycle. However, during a multicycle floating point coprocessor operation, the CPU pipeline can continue in parallel until a data or resource dependency is detected.

The pipeline is fully interlocked, so pipeline stalls and slips are controlled by hardware. Although a fully interlocked pipeline allows more compact code by removing the need for NOP padding, a compiler of real-time applications must be able to identify dependencies and accurately calculate instruction latencies.

Graphics support. None.

Communication support. No support for message passing in a distributed memory model.

MIPS R10000[15]

Noteworthy instructions. Branch penalty *varies* between one and four cycles. This bad for real-time.

Coprocessor Support. The architecture has definitions for four coprocessors, C0-C3. C0 is for address translation, C1 is floating point, and C2 and C3 are reserved by MIPS for future use. In effect, there is no support for adding a tightly coupled coprocessor other than through a standard shared memory multiprocessor configuration.

Interrupts and context switches. No details.

Addressing. 64 entry TLB with 44-bit virtual addressing.

Caches and buffers. The primary caching scheme is a Harvard architecture with a 32KB I-cache two-way set-associative and a 32 KB data cache two-way set-associative and *two-way interleaved* to hide SRAM access delays. Both caches are virtually addressed. Cache line size on the I-cache is 64 bytes, but 32 bytes on the D-cache. The secondary cache interface can operate at 200 MHz for a peak transfer rate of 3.2 GB/sec. Minimum size is 512 KB, max is 16 MB.

Monitoring support. No details.

Integer unit. There are 64 registers. Other than multiply and divide integer operations require one cycle. Multiply is 6 cycles for single-precision and 10 for double-precision. Divide is 35 cycles for single precision and 67 for double-precision. The SPECint92 rating is 250.

Floating point unit. The floating point register file is 64 entries. Multiply/add instructions are available. Operations are single cycle except division that requires 11/18 cycles for single/double precision, square root with 17/32 cycles. The SPECfp92 rating is 350.

External Data paths. The path to main memory is 64 bits wide with multiplexed address/data and can run at 200, 133, 100, 80, 67, 57, and 50 MHz. The bus is split transaction to improve performance allowing up to four outstanding requests to be issued. The 128-bit path to secondary cache provides 3.2 GB of bandwidth.

Additional pipeline capabilities. The pipeline is superscalar and can issue up to 4 instructions per cycle that can execute out-of-order. Up to four outstanding branch instructions can be in the pipe at once.

Graphics support. None.

Communication support. No support for message passing in a distributed memory model.

Technology. Available in a 527 pin PGA, fabricated with CMOS 0.5 micron technology with 6 million transistors and runs at 3.3V.

i860[16, 17, 18]

Noteworthy instructions. A *FLUSH* instruction empties the writeback data cache by testing the dirty bit of each cache line and writing to memory the lines with their bit set. The execution time of this instruction depends on the number of dirty lines in the cache.

Prefetching of floating point values is possible through the PFLD instruction. The PFLD instruction returns data from the address generated by the third previous PFLD. This instruction uses a three-stage data FIFO and is used during vector processing to hide the memory latency. The data fetched through the PFLD instruction is not cached. This feature is handy when streaming through large arrays for scientific computation or pixel graphics operations.

An incorrectly predicted *branch* has a 1 cycle penalty.

Coprocessor Support. None.

Interrupts and context switches. Insufficient information.

Addressing. The TLB has 64 entries to map fixed size pages of 4KB. The TLB can be quickly flushed, but not selectively (e.g., flush application pages but not system pages). The reference has insufficient information on the ability of software to directly read and write the TLB registers.

Caches and buffers. The i860XP has a 16KB instruction cache and a 16KB writeback data cache. Since the I-cache is read-only it can be flushed quickly. The writeback D-cache must be flushed by cycling through each entry.

A two entry write buffer is present between the cpu and main memory.

Monitoring support. Insufficient information in the references.

Integer unit. The integer pipeline is 4 stages and is used for integer arithmetic, loads/stores, and branches. There are thirty-two 32-bit general purpose registers. An add can be started every cycle. The references had insufficient information on the execution time of multiply and divide.

Floating point unit. The floating point register file has thirty-two 64-bit registers. Floating point calculations go through an additional two stages in the pipe for a total of 6 stages. Both the add and multiply can be retired in a one cycle for single-precision, while in double-precision an add can be retired every cycle and a multiply every other cycle. The references have insufficient information about the performance of divides.

The multiply and add units can be cascaded to implement both multiply-before-add and add-before-multiply operations. This is more flexibility than most other processors which are restricted to the multiply-before-add combination.

External Data paths. The data path to memory is 64 bits. Although the electrical limit is 320MB/sec at 40MHz, the bus protocol halves this to a sustainable 160MB/sec.

Additional pipeline capabilities. The pipeline is not fully interlocked. Compilers must take into account the pipeline structure and the dependencies in the code when ordering instructions. This should result in larger binaries than an interlocked design due to added NOPs, however, instruction latencies are now explicit in the code. While the hardware design is simplified, the basic compiler must be relatively knowledgeable to attain correct operation with good performance.

The floating point and integer units can run in parallel when the processor is used in dual-instruction mode. The integer unit performs all the memory accesses for the floating point unit so vector processing of floating point operations can be sustained. Branch instructions can also be issued with floating point operations minimizing looping overhead. Dual instruction mode is distinct from superscalar designs in that the compiler explicitly programs the dual-instruction mode where the usual superscalar designs use hardware to determine if multiple instructions may be issued together. Here again, the i860 simplifies its hardware by complicating the compiler. Although a high-end optimizing compiler for any processor will likely incorporate hardware knowledge to improve performance, the i860 requires a high degree of sophistication for correct operation.

Graphics support. Floating point multiply-accumulate function for fast transformations.

Graphics operations are performed in the floating point unit. Operations explicitly supported are pixel and z interpolation as well as z-buffer checking. The dual instruction mode allows very efficient 3D tranformations to be implemented in software (a 4x4 matrix multiply and accumulate). At 40MHz 500,000 transformations per second have been achieved.

Pixel data may be 8, 16, or 32 bits for intensity or color and 16 or 32 bits for depth (z direction). The graphics unit uses the floating point data paths and works on 64 bits of pixel data at a time (8, 4, or 2 pixels at a time). The 16 or 32 bit z-buffer check compares the z value of a pixel in a triangle with the current z-buffer value for that pixel position and stores the lesser of the two values (i.e. the closer pixel). The *faddp* instruction (add with pixel merge) combined with the *form* instruction (OR with merge register) aids in doing color interpolation (Gouraud shading) on each of the three color bands and then quickly recombining the results. For a 40MHz processor, rendering 40,000 100-pixel triangles per second is expected. This is double the performance of the HP PA7100.

Communication support. No support for message passing in a distributed memory model. Basic shared memory multiprocessing is supported.

Other information. Intel is dropping support for the i860.

PowerPC 601[19]

Noteworthy instructions. Due to the separate branch unit, branches dependent on a count register present no delay in the instruction stream, effectively executing in zero cycles. For other types of conditional branches the reference has insufficient information to determine the cost of a branch penalty.

Although floating point unit and fixed point unit instructions can proceed out of order, the 601 enforces precise exception reporting. Overhead, if any, to do so is not discussed.

Coprocessor Support. Insufficient information.

Interrupts and context switches. Insufficient information.

Addressing. The TLB has 256 entries to map 4KB pages and up to 4 blocks of sizes from 128KB to 8MB. The reference had insufficient information about the control of the TLB by software.

The instruction fetcher has a translation shadow array (TSA) which holds the 4 most recent translations. It is basically a cache to the TLB which resides in the MMU. A miss in the TSA forces the instruction fetcher to arbitrate for the primary MMU for translation, adding variance to the execution time of the translation. Real-time applications need a single, large TLB or need to manage this small cache explicitly.

Caches and buffers. The 32KB unified cache uses the low order 12 address bits to index into the 8-way set associative cache. These address bits are the same for the logical and physical address.

The multiprocessor coherency hardware supports the many common protocols and the PowerPC cache control operations allow explicit cache management.

The write buffer holds three 32-byte sectors to decouple the processor from the memory system.

Monitoring support. An on-chip real-time clock exists, but its precision and access time are not discussed.

Integer unit. The integer pipeline is four stages and most instructions (add, logicals, shifts, etc.) execute in a single cycle with the exception of mulitply and divide. Since the 601 is a 32-bit machine, double precision integer operations may require multiple cycles; however, double precision performance is not explicitly discussed. The SPECint92 rating is 60.

Floating point unit. The register set has thirty-two 64-bit registers. All denormalized numbers, NANs, QNANs, and infinities, are automatically handled in the hardware. There is no discussion of the time overhead in handling a denormalized number, but we can assume it is small if not zero cycles. All single precision operations, except division, can be retired one per cycle; division requires 17 cycles. Double precision operations can be retired one per cycle, except multiplication (2 cycles) and division (31 cycles). The SPECfp92 rating is 80.

Multiply-accumulate operations are supported.

External Data paths. The memory data bus is 64 bits wide with an electrical limit of 528MB/sec and a sustainable limit of 422MB/sec.

Additional pipeline capabilities. The processor is superscalar and up to 3 instructions per cycle may be issued, one to each of the units, branch, fixed point (integer), and floating point. A common mode will have the fixed point unit calculating next addresses while floating point operations run in parallel. This is a popular method of performing vector processing in the microprocessors surveyed.

Graphics support. Floating point multiply-accumulate function for fast transformations.

Communication support. No message passing support.

PowerPC 604[20]

Noteworthy instructions. Due to the separate branch unit, branches dependent on a count register present no delay in the instruction stream, effectively executing in zero cycles. There is a branch penalty of one cycle on a misprediction.

Although floating point unit and fixed point unit instructions can proceed out of order, the 604 enforces precise exception reporting.

A count register (CTR) is used by some branch instructions to count down the iterations of a loop. A shadow CTR with the future CTR value is used to predetermine the end condition of a loop and branch out with no penalty.

Coprocessor Support. Insufficient information.

Interrupts and context switches. Insufficient information.

Addressing. 128 two-way set-associative TLB.

Caches and buffers. Has a 16 KB data cache and 16 KB instruction cache, both are 4 way set associative and are *physically addressed* with 32 byte blocks. Nonblocking cache access (up to 2 requests outstanding). The PowerPC architecture allows some control over loading and invalidating cache lines.

There is a cache lock feature that allows the cache to be accessed but not modified.

The multiprocessor coherency hardware supports the many common protocols and the PowerPC cache control operations allow some explicit cache management.

Monitoring support. The 604 has hardware to report on instruction execution rate, branch prediction rate, cache hit rates, average cache miss latency and allows single-stepping, placing breakpoints, and branch instruction tracing.

Integer unit. Integer multiply (32x32) can retire one result every 2 cycles. Division is 19 cycles. The SPECint92 rating is 160.

Floating point unit. The register set has thirty-two 64-bit registers. All denormalized numbers, NANs, QNANs, and infinities, are automatically handled in the hardware. All single precision operations, except division, can be retired one per cycle; division requires 18 cycles. Double precision operations can be retired one per cycle, except division (31 cycles). The SPECfp92 rating is 165.

Multiply-accumulate operations are supported.

External Data paths. The memory data bus is 64 bits wide with an electrical limit of 528MB/sec and a sustainable limit of 422MB/sec.

Additional pipeline capabilities. The processor is superscalar and up to 4 instructions per cycle may be issued and execute out-of-order but with precise interrupts. A common mode will have the fixed point unit calculating next addresses while floating point operations run in parallel. This is a popular method of performing vector processing in the microprocessors surveyed. The branch history table (BHT) has 512 entries, the branch target address cache (BTAC) has 64 entries.

On a branch dispatch other instructions in the other three dispatch units are delayed and then dispatched in the following cycle. This is to simplify state saving to recover from a mispredicted branch.

Graphics support. Floating point multiply-accumulate function for fast transformations.

Communication support. No message passing support.

Technology. 0.5 micron CMOS, 4 metal layers, 3.6 million transistors, 3.3V, 304-pin CQFP, less than 10W at 100 MHz.

PowerPC 620[21]

The 620 is essentially a 64-bit version of the 604, differences are noted below.

Noteworthy instructions.
Coprocessor Support.
Interrupts and context switches.
Addressing.
Caches and buffers. Has a 32 KB data cache and 32 KB instruction cache and are *physically addressed* (like the 604). Configurable second level cache from 1 to 128 MB.
Monitoring support.
Integer unit. The SPECint92 rating is 225.
Floating point unit. Single precision division is 18 cycles and square root is 22. The SPECfp92 rating is 300.
External Data paths. The memory data bus is 128 bits wide with an electrical limit of 1064MB/sec.
Additional pipeline capabilities. Can execute up to 4 unresolved branch instructions vs 2 for the 604.
Graphics support.
Communication support.
Technology. 0.5 micron CMOS, 4 metal layers,7 million transistors, 3.3V, 482-pin BGA, 30W worst case at 133 MHz

Motorola 88110[22]

Noteworthy instructions. Incorrectly predicted branches have a 2 cycle penalty.

Coprocessor Support. The 88110 has support for running in lockstep with another 88110, once synchronized. It is not clear how general this synchronization feature is and if it could be used for tight coupling with a system processor.

Interrupts and context switches. On exceptions, instruction issue is halted until the pipe empties. Then, the machine is backed up to the state that existed before the instruction which caused the exception occurred. Due to the different latencies of the various functional units, the time to empty the pipe is variable.

On an external interrupt, the processor halts, aborts all unfinished instructions, and backs out of any instructions that completed out of order. This minimizes interrupt response latency.

Addressing. The TLB has 32 entries for 4KB pages and 8 entries for blocks of 512KB to 64MB.

Caches and buffers. The Harvard cache architecture has an 8KB I-cache and an 8KB D-cache both of which are logically indexed. The D-cache uses a writeback policy. A fast flush capability of the I-cache and D- cache (5 cycles) is available. To write back dirty lines from the D-cache requires 1 cycle per cache line plus the memory transfer time per dirty line.

The *touch-load* instruction brings in a line of data into the cache before it is actually needed. A line flush instruction can force a cache line out to memory.

Some instructions can be specially marked to "store-through" the cache. Although these stores bypass the cache, if there is a cache hit then the line is updated; but if there is a miss then the line is not brought into the cache.

The logic for a second-level cache is provided by the 88410 cache controller.

Monitoring support. Insufficient information.

Integer unit. The general purpose register file has thirty-two 32-bit registers that can be used by any functional unit in the processor, including floating point.

Both the integer unit and the multiplier can retire an instruction every cycle. The multiplier is used on floating point and graphics values as well as integer.

The SPECint89 rating of the processor is 51.

Floating point unit. The register file has thirty-two 80-bit registers to support IEEE double-extended values. Single- and double floating point numbers are supported in the general purpose register file, also.

Adds can be started every cycle, as well as can multiplies. Divides depend on the operand type and precision, but since this information is known at compile time division is not data dependent. Divides require 13 cycles for single-precision floating point. The SPECfp89 rating is 73.9.

External Data paths. The system bus is 64 bits wide with an electrical limit of 400MB/sec.

Additional pipeline capabilities. There are 10 independent execution units: branch, integer (two), bit field, multiplier, floating-point adder, divider, graphics (two), and data or load/store.

A time-critical floating point mode supports real-time applications. In this mode, the hardware delivers arithmetically sensible results rather than trapping on exceptional conditions such as underflow, overflow, and NAN (not a number). This mode reduces data-dependent variations in execution time.

There is no ability to directly cascade functional units (such as for the useful multiply-add function).

Graphics support. There is hardware support for floating point infinities, to eliminate the need to trap to IEEE software handlers. This a frequent occurrence in some graphics algorithms.

The processor can work on 64 bits of pixel data at a time. Pixel information can be 4-, 8-, 16-, or 32-bits, so 16, 8, 4, or 2 pixels can be operated on simultaneously. The arithmetic unit supports modulo and saturation arithmetic. Modulo arithmetic allows wrap-around to occur when the minimum/maximum supported value is surpassed. Saturation arithmetic clamps values to the minimum/maximum supported value when it is exceeded. Saturation arithmetic prevents visual anomalies occurring and saves on explicit intensity value boundary checks.

General instructions for unpacking, truncating, packing, and rotating 4-, 8-, 16, and 32-bit fields manipulate specific fields within the pixel data. Combinations of these instructions perform the same functions as the *faddp* and *form* instructions of the i860.

There are two graphics units, one for arithmetic operations and one for bit-field manipulations.

The line flush instruction for the caches (see above) is stated to be efficient for updating video frame buffers using the burst-mode capability of the system bus. Defining the frame buffer memory as write-through would result in a less efficient transfer across the bus.

Communication support. There is support for shared memory multiprocessing, but not a distributed message passing system.

Other information. Motorola is dropping support for the 88110.

SuperSPARC (TMS390Z50)[23, 24]

Noteworthy instructions. Instructions are issued in order, however, floating point *results* may be out of order relative to the integer instructions. This is due to the decoupling of the integer and floating point pipelines.

Precise interrupts are supported by having the hardware track the address of each instruction in the pipeline (as many as 16 instructions).

The references do not have enough information to determine the branch penalty.

Coprocessor Support. Insufficient information.

Interrupts and context switches. The register windowing scheme (see integer unit entry below) makes procedure calls and interrupts efficient provided that there are available windows. Variance in execution time occurs when the number of register sets needed exceeds the number available. When this occurs register sets must be written to memory to free them for the most recent scope. A controlled variance windowing scheme for real-time applications may be possible provided that software has the ability to explicitly manage the window sets (e.g., reserve sets specifically for system use like fast interrupt response).

The large number of registers (128) makes context switching expensive (most other processors have 64 registers).

Addressing. A 64 entry TLB is present. The architecture specification allows 4KB, 256KB, 16MB, and 4GB pages to be mapped. This range is large, but the granularity is much too coarse. The TLB is writeable but indirectly via an LRU hardwired replacement policy. It can, however, be flushed quickly. Using the lock bit, selective flushing is also possible.

Caches and buffers. The instruction cache is 20KB and the data cache is 16KB. A 1MB physically addressed second level cache is supported.

A writebuffer of 3 entries exists.

Monitoring support. Insufficient information.

Integer unit. The SuperSPARC uses register windows. A window consists of 8 output registers, 8 local registers, and 8 input registers. There are 8 windows total. Since a window's output registers overlap with the input registers of the next window, the 8 windows require only 8 sets of 16 registers for a total of 128 registers. The window organization makes for very efficient procedure calls that have a small set of parameters.

In [1], Hennessy and Patterson state that tregister windows have proven useful for LISP code and object oriented code where interprocedural register allocation optimizations cannot be performed by the compiler. This optimization works best when all procedures are compiled together, so it is an expensive technique to apply to LISP code which is often used in a rapid debug-edit-compile cycle. In object-oriented languages such as Smalltalk with their dynamic equivalent of a procedure call, the compiler does not know which procedure will be invoked. In these environments register windows may have an advantage that cannot be compensated for by an intelligent compiler.

Floating point unit. The floating point unit has been optimized for double-precision arithmetic. Single- and double-precision adds and multiplies can be retired one per cycle. Single-precision division takes 4 cycles and double-precision takes 5. Single-precision square root takes 6 cycles and double-precision takes 8.

The FPU handles all non-exception generating numeric cases in hardware, requiring no software assistance (i.e., no traps) for special cases such as subnormal results.

External Data paths. The 64-bit system bus has an electrical limit of 400MB/sec. at 50MHz.

Additional pipeline capabilities. The pipeline is superscalar and is capable of issuing up to 3 instructions per cycle. There are restrictions on the combination of instructions that can be issued togther, but in general any combination of different types of instructions (integer, floating point, branch, shift, memory reference) may be issued simultaneously. The exception is that two 32-bit integer results can be created during any cycle.

Graphics support. None.

Communication support. No message passing support for distributed memory systems.

UltraSparc[25]

Noteworthy instructions. The UltraSparc has a full set of pixel operations and high speed block memory moves. The references do not have enough information to determine the branch penalty.

Coprocessor Support. Insufficient information.

Interrupts and context switches. The register windowing scheme (see integer unit entry below) makes procedure calls and interrupts efficient provided that there are available windows. Variance in execution time occurs when the number of register sets needed exceeds the number available. When this occurs register sets must be written to memory to free them for the most recent scope. A controlled variance windowing scheme for real-time applications may be possible provided that software has the ability to explicitly manage the window sets (e.g., reserve sets specifically for system use like fast interrupt response).

The large number of registers (128) makes context switching expensive (most other processors have 64 registers).

Addressing. Insufficient information.

Caches and buffers. The instruction cache is 16KB two-way set-associative and the data cache is 16KB direct mapped.

Monitoring support. Insufficient information.

Integer unit. The UltraSPARC uses register windows similar to the SuperSparc. A window consists of 8 output registers, 8 local registers, and 8 input registers. There are 8 windows total. Since a window's output registers overlap with the input registers of the next window, the 8 windows require only 8 sets of 16 registers for a total of 128 registers. The window organization makes for very efficient procedure calls that have a small set of parameters. A new feature has been added so that on an exception a new window is selected. This can harm predictability when there is a register overflow and a whole window must be written out to memory.

Integer multiply executes 2 bits of the multiplicand per cycle while division executes 1 bit of the divisor per cycle. SPECint92 is estimated to be between 250 and 300.

Floating point unit. The floating point unit has been optimized for double-precision arithmetic. Single- and double-precision adds and multiplies can be retired one per cycle. Division takes longer.

External Data paths. The 128-bit system bus has an electrical limit of 1600MB/sec. at 100MHz.

Additional pipeline capabilities. The pipeline is superscalar and is capable of issuing up to 4 instructions per cycle. There are restrictions on the combination of instructions that can be issued togther, but in general any combination of different types of instructions (integer, floating point, branch, shift, memory reference) may be issued simultaneously.

Graphics support. A complete set of pixel operations is defined to handle 64 bits of pixel data per cycle. A high speed block memory move that can sustain up to 600 MB/sec is available.

Communication support. No message passing support for distributed memory systems.

Texas Instruments TMS320C40[26]

Noteworthy instructions. Branches require zero cycles if it is a delayed branch because no prediction is made, but there is a 3 cycle delay. Special instructions RPTB, RPTBD, and RPTS are for repeating blocks of code and use special hardware for a zero delay and 1 or 0 cycle overhead branching.

There is no penalty on conditional branches because the C40 does not do branch prediction. A branch requires 4 cycles. A delayed branch is available that requires the next 3 instruction slots to be filled with work that will occur before the branch.

The repeat single-instruction, RPTS, is not interruptible. This instruction sets the repeat count register to the given value then repeatedly executes the next instruction, decrementing the counter on each cycle. The repeat count (RC) register is the loop control register and is 32-bits. Since this instruction cannot be interrupted, the potential latency that an interrupt might experience is theoretically on the order of 2^{31} cycles.

Coprocessor Support. The DMA ports allow a loose coupling with relatively high latency.

Interrupts and context switches. During an interrupt only the program counter is saved. Other state must be explicitly saved by the software. This gives the software complete control to streamline individual interrupts. The manual lists code to save the complete context; the routine requires 39 cycles. At 40ns per cycle the total time is 1.6μs.

Addressing. The C40 does not support virtual addressing, so it does not have a TLB. Thus, individual address spaces for tasks that require protection are not supported.

Caches and buffers. There is a 512-byte instruction cache, but there is no data cache. Data is fetched either from an external memory or from a fast internal 8KB memory that can be considered a cache explicitly managed by the software. There is no support for a second level cache.

A writebuffer exists in the path to main memory.

Monitoring support. A timer register is on-chip with a frequency of half the processor clock. It is quickly accessed through simple control register read and write operations.

Integer unit. The C40 is a 32-bit processor and has eight 32-bit general purpose registers mostly used for addressing. Thirty-two bit adds and multiplies take 1 cycle. Division is performed through a small subroutine of other instructions so it is interruptible. Its execution time is data dependent.

Floating point unit. The floating point unit has 12 40-bit registers. Floating point addition and multiplication require 1 cycle. Division is approximated through a reciprocal operation (accurate to 8 bits) to give an approximate answer in 2 cycles.

External Data paths. There are two external interfaces to memory, one for global memory and one for local memory. Each bus is 32-bits with an electrical limit of 100MB/sec. For distributed memory models there are 6 ports each with a 20MB/sec. electrical limit and 16MB/sec. maximum sustainable rate for DMA transfers.

Additional pipeline capabilities. The C40 is a dual issue processor with zero overhead branching capability in looping constructs; very efficient pipelining sequences can be programmed. Instructions explicitly encode the two particular operations to be executed in parallel. There are four basic stages in the pipe with the DMA unit as a fifth functional unit and instructions are executed in order.

Graphics support. None.

Communication support. Interrupts, shared-memory multiprocessing, and ports for messaging passing in distributed memory systems. Shared memory multiprocessing support is through explicit instructions for *interlocked* access to external memory, hence coherency is managed through software with the minimum of hardware support.

For distributed memory models there are 6 ports for DMA transfers. Each port has 9 control registers to load which takes about 380ns per port. The data transfer itself has an overhead of 500ns for the first 32-bit word and 250ns for subsequent 32-bit words.

Other information. Texas Instruments is noncommittal about future support for the C40.

Analog Devices SHARC[27]

Noteworthy instructions. The reference has insufficient information on the instruction set.

Coprocessor Support. It provides support to run an array of SHARC processors in lock-step (SIMD computing). The flexibility of this synchronization mechanism is unclear.

Interrupts and context switches. Insufficient information.

Addressing. There is no virtual addressing support or process protection.

Caches and buffers. A 512KB on-chip memory is software partitionable into separate I- and D-caches. There is no support for a second level cache.

Monitoring support. Insufficient information.

Integer unit. The processor is a 32-bit machine. All 32-bit operations are single-cycle and non-pipelined.

Floating point unit. All 32-bit operations are single-cycle and non-pipelined.

External Data paths. The external parallel port has an electrical limit of 240MB/sec (48-bits run at 40MHz). There are 6 DMA IO channels of 40MB/sec each. Two full-duplex serial ports add 10MB/sec each.

Additional pipeline capabilities. The core of the SHARC is the Analog Devices ADSP-21020. The SHARC chip adds a lot of on-chip memory and 6 highspeed communication ports. The architecture supports parallel ALU and multiplier operations while fetching two words from memory in a single cycle. Two Data Address Generators (DAGs) allow both data fetching and conditional branching to occur in parallel with computation.

Graphics support. None.

Communication support. The processor has 6 ports of 40MB/sec. each to implement a distributed memory system. The information on the overhead in setting up a channel is not available in the reference.

Other information. The SHARC has not yet been fabricated.

Adaptive Solutions CNAPS[28, 29]

The Adaptive Solutions CNAPS architecture is radically different from the processors already discussed and as such it will be presented in a less constrained format. The CNAPS chip is designed to be a coprocessor to a host and has 64 processors (PNs) per chip arranged in a 1D array. They operate in a SIMD fashion using a single control sequencer. The target applications are in AI and for computational routines that perform tasks such as neural net simulation, spectral analysis, JPEG, wavelet processing, and Gabor filtering.

The current design runs at 25 MHz with 4KB on-chip memory per PN and 25 MB/sec peak IO bandwidth, 20MB/sec. sustained. The execution unit is implements fixed point calculations not floating point. An 8-bit multiply can be performed each cycle by each PN, however, a 16-bit multiply requires two cycles. The design is optimized for large data parallel operations requiring modest precision such as in neural net computations and low-level image processing. The next generation will enhance every aspect of the chip (faster pipeline, more memory, greater IO bandwidth, etc.) including adding decoupling to make the CNAPS chip a loosely coupled coprocessor.

The computational shortcomings of the processor are the following. It is strictly a coprocessor optimized for low precision arithmetic, not for scientific floating point computations and graphics. The design favors data parallelism, not control parallelism. And, it currently has low IO bandwidth.

The reference has insufficient information to comment on the processor's real-time support. However, since the predictability of an application's execution time using CNAPS processors will depend largely on the array control sequencer and the host, it is necessary for both to be real-time processors if any such system is to efficiently support real-time computing. In addition, the future move of Adaptive Solutions to a more loosely coupled interface between the host and the CNAPS system may cause difficulties in a real-time environment.

	Alpha 21064-AA	Pentium	HP PA7100
Branch Penalty	2 cycles	>= 3 cycles	1 cycle
Variable execution time instructions	• Division requires from 16 to 144 cycles. • Arithmetic operations can raise exceptions.	****	• Integer multiply and divide are data dependent. • Arithmetic exeptions
Coprocessor support	Only through shared memory multiprocessing and interprocessor interrupts.	****	The architecture definition defines special functional units (SFUs) and coprocessors. It is unclear if the PA7100 implements these or if they are dedicated.
Interrupt and context switch handling	Done via PALcode, so slow interrupt response.	****	Fast interrupt response due to shadow registers.
Addressing	12 entry instr. TLB and 32 entry data TLB mapping 8KB to 4MB pages with coarse increments. Slow PALcode allows selective flush.	The Pentium uses a TLB	120 entry combined TLB for 4KB pages with and additional 12 entries for pages up to 64MB. TLB is managed entry by entry, no quick flush.
Caches and buffers	8KB I-cache and 8KB D-cache, logically addressed and has coherency support for multiprocessing. Physically addressed 2nd level cache supported.	8KB I-cache and 8KB D-cache, logically addressed.	*External* caches allow a 1MB I-cache and a 2MB D-cache, both logically addressed with nstructions to purge and flush the caches. No 2nd level cache support.
Monitoring support	A cycle counter is accessible via slow PALcode as well as counters for other events.	****	****
Integer unit	32 64-bit regs	8 registers	32 32-bit regs
Floating point unit	32 64-bit regs	8 registers	32 64-bit regs
Additional pipeline notes	Superscalar (dual issue) allowing restricted use of both the int and float units for some level of vector processing. No multiply-accumulate ability.	The pipe is dual issue and has two integer units. Vector processing can occur using the floating point unit for calculation and the integer unit for operand fetch.	Superscalar (dual issue) for floating point vector processing. Multiply-accumulate operations are supported.
Performance (64-bit add/mult/div in cycles): Integer Floating point	 a=1, m=16, d=16-144 a=1, m=1, ****	 ****, ****, **** a=1, m=1, d=32	 a=1, ****, **** a=1, m= 1, d= 15
SPEC ratings	int89: 116.2 fp89: 275.8 int92: 74.3 fp92: 126.0	int89: **** fp89: **** int92: 64.5 fp92: 56.9	int89: 88.1 fp89: 206.2 int92: 109.1 fp92: 167.9
External datapaths	1200 MB/sec.	528 MB/sec.	400 MB/sec.
Graphics support	None	None	Block move, z-buffering support, color interpolation, clip test instructions. The PA7100 alone can render 19,400 100-pixel objects per second.
Computational models directly supported	Shared memory	Shared memory	Shared memory
Clock rate	150 MHz	66 MHz	100 MHz

	MIPS R4400	i860	PowerPC 601
Branch Penalty	2 cycles	1 cycle	****
Variable execution time instructions	Arithmetic exceptions.	Arithmetic exceptions.	Arithmetic exeptions handled in hardware; overhead is unknown, but it must be zero for real-time concerns.
Coprocessor support	The architecture defines 3 coprocessors, but their use is reserved.	None	****
Interrupt and context switch handling	****	****	****
Addressing	48 pairs of TLB entries. Each pair maps two consecutive virtual pages. Pages may be from 4KB to 16MB with an increment factor of 4.	64 TLB entries map fixed 4KB pages. Has a quick flush but not selectively.	256 entries for 4KB pages and allows up to 4 entries to map blocks of 128KB to 8MB. Instruction unit uses the TSA as a 4 entry TLB cache. Unclear if software has direct control of TSA and/or TLB.
Caches	16KB I-cache and 16KB D-cache, logically addressed and has coherency support for multiprocessing. Physically addressed 2nd level cache.	16KB I-cache and 16KB D-cache, logically addressed.	32KB unified cache is logically accessed. Cache control operations allow explicit cache management to the software.
Monitoring support	A cycle counter can be quickly accessed as a special register. Increments every other cycle.	****	An on-chip real-time clock exists, but details on its precision and access time not available.
Integer unit	32 64-bit regs	32 32-bit regs	****
Floating point unit	32 64-bit regs	32 64-bit regs	32 64-bit regs
Additional pipeline notes	Superpipelined so can only issue 1 instruction per cycle. No multiply-accumulate ability.	The pipe is dual issue and supports cascaded floating mult and add units (mult-add, add-mult). Non-interlocked pipe requires compiler support for correct operation.	Triple issue superscalar design for efficient floating point vector processing and includes mult-accum. All arithmetic exceptions can be handled directly in hardware. The overhead is not clear.
Performance (64-bit add/mult/div in cycles): Integer Floating point	 a=1, m=20, d=133 a=1, m=4, d=35	 a=1, ****, **** a=1, m=2, ****	 ****, ****, **** a=1, m= 2, d= 31
SPEC ratings	int89: 82.5‡ fp89: 103.5‡ int92: 94.2 fp92: 105.2	int89: **** fp89: **** int92: **** fp92: ****	int89: **** fp89: **** int92: 62 fp92: 80
External datapaths	600 MB/sec.	320 MB/sec. (160 sustained)	528 MB/sec. (422 sustained)
Pixel graphics support	None	Z-buffer checking along with pixel and z interpolation support. The i860 can render 40,000 100 pixel objects per second.	None
Computational models directly supported	Shared memory	Shared memory	Shared memory
Clock rate	150 MHz	50 MHz	66 MHz

‡estimated values

	Motorola 88110	SuperSPARC	TI TMS320C40
Branch Penalty	2 cycles	1 cycle	0 cycles (No penalty)
Variable execution time instructions	Interrupts require the pipeline to be flushed, but the different latencies in the various pipes (int vs. float) cause the stall time to be dependent on preceding instr.	• Arithmetic exceptions • Traps when register window stack exceeded.	RPTS, repeat single-instruction is uninterruptible and can be given a loop count up to 2^{32} iterations.
Coprocessor support	Can run in lockstep with another 88K once synced. Unclear if mechanism is general enough.	****	Six DMA controllers allow loosely coupled multiprocessing.
Interrupt and context switch handling	On external interrupt CPU halts and aborts unfinished instr. for fast response.	Has register windowing, but no control given to software.	Only PC saved by hardware, software saves only what is necessary. Complete state save is 39 cycles.
Addressing	32 entries for 4KB and 8 entries for pages up to 64MB	64 TLB entries map up to 4GB pages. Very coarse granularity. Quick selective flush. Hardware controlled using an LRU algorithm.	Only physical addressing supported. There is no TLB.
Caches	8KB I-cache and 8KB D-cache, logically addressed. Physically addressed 2nd level cache. Touch-load allows prefetching. "Store-thru" restricts cache updates.	20KB I-cache and 16KB D-cache, logically addressed. A 1MB 2nd level cache is supported.	512 byte instruction cache and an 8KB on-chip memory that can be explicitly managed by software as a cache.
Monitoring support	****	****	A half frequency timer easily accessible
Integer unit	32 32-bit regs	128 general regs (8 windows)	8 32-bit regs
Floating point unit	32 80-bit regs	Uses general regs	12 40-bit regs
Additional pipeline notes	Dual issue capability, but cannot cascade units for mult-accumulate function. Can handle all arithemtic exceptions in hardware for real-time applications.	The pipe is triple issue and two 32-bit integer results can be generated per cycle.	Dual issue pipe, but the instruction explicitly encodes the parallelism. All operations are non-pipelined in the execution unit. In order instruction completion.
Performance (64-bit add/mult/div in cycles): Integer Floating point	 a=1, m=1, **** a=1, m=1, ****	 **** **** , **** a=1, m=1, d=5	32-bit ops only a=1, m=1, **** a=1, m=1, d=2
SPEC ratings	int89: 51.0 fp89: 73.9 int92: **** fp92: ****	int89: **** fp89: **** int92: 65.0 fp92: 80.0	int89: **** fp89: **** int92: **** fp92: ****
External datapaths	400 MB/sec.	400 MB/sec.	100 MB/sec. (global or local) 120 MB/sec. (6 DMA ports)
Pixel graphics support	Hardware for floating point infinities, modulo and saturation arithmetic, general packing/unpacking instr., line flushing to video buffer.	None	None
Computational models directly supported	Shared memory	Shared memory	6 DMA channels for distributed memory system interconnect. Minimum shared memory support.
Clock rate	50 MHz	50 MHz	25 MHz

	SHARC	CNAPS
Branch Penalty	****	1 cycle
Variable execution time instructions	****	****
Coprocessor support	Arrays of SHARC processors can be synchronized to run in lockstep SIMD mode.	CNAPS is *not* a general processor, it is a coprocessor itself.
Interrupt and context switch handling	****	****
Addressing	Physically addressed, no TLB	****
Caches	512KB on-chip memory can be partitioned by software into I- and D-caches. No 2nd level cache support.	4KB memory on-chip.
Monitoring support	****	****
Integer unit	****	****
Floating point unit	****	****
Additional pipeline notes	Core of SHARC is ADSP-21020 DSP. Parallel ALU and multiplier operations supported. The 32-bit processor performs all 32-bit operations in a single cycle.	Pipe is optimized for low precision integer operations.
Performance (64-bit add/mult/div in cycles): Integer Floating point	32 bit ops only a=1, m=1, d=1 a=1, m=1, d=1	Low precision architecture **** **** , **** **** , **** , ****
SPEC ratings	int89: **** fp89: **** int92: **** fp92: ****	int89: **** fp89: **** int92: **** fp92: ****
External datapaths	240 MB/sec. (system bus) 240 MB/sec. (6 DMA ports) 20 MB/sec. (2 serial ports)	25 MB/sec.
Pixel graphics support	None	None
Computational models directly supported	Distributed memory multiprocssing	****
Clock rate	40 MHz	25 MHz

**** Insufficient information

	Alpha 21164	UltraSPARC	MIPS R10K
Branch Penalty	****	****	Variable, 1 to 4 cycles
Variable execution time instructions	****.	Traps when register window stack exceeded.	****
Coprocessor support	****	****	Architecture defines 3 coprocessors but their use is reserved.
Interrupt and context switch handling	****	Has register windowing, but no control given to software.	****
Addressing	****	****	64 entry TLB with 44-bit virtual addressing.
Caches	8KB I-cache and 8KB D-cache, logically addressed. On-chip 96 KB secondary cache. Has support for a third level of cache.	16KB I-cache and 16KB D-cache, logically addressed, two-way set-associative.	32 KB I and D caches, two-way set-associative. The D cache is also two-way interleaved.
Monitoring support	****	****	****
Integer unit	32 64-bit regs	128 general regs (8 windows)	64 64-bit regs
Floating point unit	32 64-bit regs	Uses general regs	64 64-bit regs
Additional pipeline notes	Quad issue capability with in-order execution.	The pipe is quad issue.	Quad issue pipe with out-of-order execution. Up to four branches may be outstanding.
Performance (64-bit add/mult/div in cycles): Integer Floating point	****, ****, **** ****, ****, ****	a=1, m=32, d=64 a=1, m=1, d=****	a=1, m=10, d=67 a=1, m=1, d=18
SPEC ratings	int89: **** fp89: **** int92: 330 fp92: 500	int89: **** fp89: **** int92: 250-300 fp92: ****	int89: **** fp89: **** int92: 250 fp92: 350
External datapaths	3200 MB/sec. peak	1600 MB/sec. peak	1600 MB/sec. peak
Pixel graphics support	****	Full support	None
Computational models directly supported	Shared memory	Shared memory	Shared memory
Clock rate	300 MHz	200 MHz	200 MHz

	604	620
Branch Penalty	1 cycle	1 cycle
Variable execution time instructions	****	****
Coprocessor support	None	None
Interrupt and context switch handling	****	****
Addressing	****	****
Caches	16 KB I and D caches, both 4-way set-associative, physically addressed.	32 KB I and D caches.
Monitoring support	instr. execution rate, branch prediction rate cache hits, miss latency, single stepping, breakpoints.	instr. execution rate, branch prediction rate cache hits, miss latency, single stepping, breakpoints.
Integer unit	****	****
Floating point unit	****	****
Additional pipeline notes	Quad issue superscalar.	Quad issue superscalar
Performance (64-bit add/mult/div in cycles): Integer Floating point	 a=1, m=2, d=19 (32-bit) a=1, m=1, d=31	 **** **** **** **** **** ****
SPEC ratings	int89: **** fp89: **** int92: 160 fp92: 165	int89: **** fp89: **** int92: 225 fp92: 300
External datapaths	422 MB/sec.	1064 MB/sec.
Pixel graphics support	None	None
Computational models directly supported	Shared memory	Shared memory
Clock rate	100 MHz	133 MHz

**** Insufficient information

References

[1] Hennessy, J.L. and D.A. Patterson, *Computer Architecture A Quantitative Approach*. Second ed. 1990, Palo Alto, CA: Morgan Kaufmann Publishers Inc.

[2] Knieser, M.J. and C.A. Papachristou. *Y-Pipe: A Conditional Branching Scheme Without Pipeline Delays*. in *International Symposium on Microarchitecture*. 1992.

[3] Stankovic, J. *The Spring Architecture*. in *EuroMicro Workshop on Real-Time*. 1990.

[4] *Alpha Architecture Reference Manual*. ed. R.L. Sites. 1992, Maynard: Digital Equipment Corporation.

[5] *DECchip 21064-AA Microprocessor Hardware Reference Manual*. first ed. 1992, Maynard: Digital Equipment Corporation.

[6] McLellan, E., *The Alpha AXP Architecture and 21064 Processor*. IEEE Micro, 1993. **13**(June): p. 36-47.

[7] Ryan, B., *Alpha Rides High*. Byte Magazine, 1994. (October): p. 197-198.

[8] Alpert, D. and D. Avnon, *Architecture of the Pentium Microprocessor*. IEEE Micro, 1993. **13**(June): p. 11-21.

[9] Lee, R.B., *Precision Architecture*. IEEE Computer, 1989. **22**(January): p. 78-91.

[10] Asprey, T., *et al.*, *Performance Features of the PA7100 Microprocessor*. IEEE Micro, 1993. **13**(June): p. 22-35.

[11] Dowdell, C. and L. Thayer, *Scalable Graphics Enhancements for PA-RISC Workstations*. IEEE COMPCON, 1992. (Spring): p. 122-128.

[12] Simha, S., *R4400 Microprocessor Product Information*. Technical Report MIPS Technologies, Inc., September 27, 1993 1993.

[13] Mirapuri, S., M. Woodacre, and N. Vasseghi, *The Mips R4000 Processor*. IEEE Micro, 1992. (April): p. 10-22.

[14] Niehaus, D., *Program Representation and Execution in Real-Time Multiprocessor Systems*1994, University of Massachusetts:

[15] Halfhill, T.R., *T5: Brute Force*. Byte Magazine, 1994. (November 1994): p. 123-128.

[16] Atkins, M., *Performance and the i860 Microprocessor*. IEEE Micro, 1991. (October): p. 24-.

[17] *i860 64-bit Microprocessor Hardware Design Guide*. 1989, Intel Corporation.

[18] Grimes, J., L. Kohn, and R. Bharadhwaj, *The Intel i860 64-Bit Processor: A General-Purpose CPU with 3D Graphics Capabilities*. IEEE Computer Graphics and Applications, 1989. (July): p. 85-94.

[19] Becker, M.C., *et al.*, *The PowerPC 601 Microprocessor*. IEEE Micro, 1993. (October): p. 54-68.

[20] Song, S.P., M. Denman, and J. Chang, *The PowerPC 604 RISC Microprocessor*. IEEE Micro, 1994. (October): p. 8-17.

[21] Thompson, T. and B. Ryan, *PowerPC 620 Soars*. Byte Magazine, 1994. (November): p. 113-120.

[22] Diefendorff, K. and M. Allen, *Organization of the Motorola 88110 Superscalar RISC Microprocessor*. IEEE Micro, 1992. (April): p. 40-63.

[23] Catanzaro, B., *Multiprocessor System Architectures*. 1994, Prentice Hall.

[24] Blanck, G. and S. Krueger, *The SuperSPARC Microprocessor*. IEEE COMPCON, 1992. **Spring**: p. 136-141.

[25] Wayner, P., *SPARC Strikes Back*. Byte Magazine, 1994. (November): p. 105-112.

[26] *TMS320C4x User's Guide*. 1991, Texas Instruments.

[27] Brewer, J.E., *et al. A Single-Chip Digital Signal Processing Subsystem*. in *International Conference on Wafer Scale Integration*. 1994.

[28] Hammerstrom, D. *CNAPS: A VLSI Image Processing Pattern Recognition Engine*. in *Advanced Vision Systems Workshop Proceedings*. 1994.

[29] Baker, T., *Artificial Neural Network and Image Processing using the Adaptive Solutions' Architecture*1991, Adaptive Solutions: Beaverton, OR. p. 1-12.