Pentium® III Processor Implementation Tradeoffs

Jagannath Keshava and Vladimir Pentkovski: Microprocessor Products Group, Intel Corp.

ABSTRACT

This paper discusses the implementation tradeoffs of the Pentium® III processor. The Pentium III processor implements a new extension of the IA-32 instruction set called the Internet Streaming Single-Instruction, Multiple-Data (SIMD) Extensions (Internet SSE). The processor is based on the Pentium® Pro processor microarchitecture.

The initial development goals for the Pentium III processor were to balance performance, cost, and frequency. Descriptions of some of the key aspects of the SIMD Floating Point (FP) architecture and of the memory streaming architecture are given. The utilization and effectiveness of these architectures in decoupling memory accesses from computation, in the context of balancing the 3D pipe, are discussed. Implementation choices in some of the key areas are described. We also give some details of the implementation of Internet SSE execution units, including the development of new FP units, and discuss how we have implemented the 128-bit instructions on the existing 64-bit datapath. We then discuss the details of the memory streaming implementation.

The Pentium III processor is now in production on frequencies of up to 550 MHz. The new instructions in the Internet SSE were added at about a 10% die cost and have enabled the Pentium III processor to offer a richer, more compelling visual experience.

INTRODUCTION

The goal of the Internet SSE development was to enable a better visual experience and to enable new applications such as real-time video encoding and speech recognition [7]. The Pentium® III processor is the first implementation of ISSE. It is based on the P6 microarchitecture, which allows an efficient implementation in terms of die size and effort. The key development goals were the implementation of the Internet SSE while keeping about a 10% larger die size than the Pentium® II processor and achieving a higher frequency by at least one bin.

Two features of these new applications challenge designers of computer systems. First, the algorithms that the applications are based on are inherently parallel in the sense that the same sequence of operations can be applied concurrently to multiple data elements. The Internet SSE allows us to express this parallelism explicitly, but the hardware needs to be able to translate the parallelism into higher performance. The P6 superscalar out-of-order microarchitecture is capable of utilizing explicit as well as extracted implicit parallelism. However, hardware that supports higher computation throughput improves the performance of these algorithms. The development of such hardware and increasing its utilization were key tasks in the development of the Pentium III processor. Second, in order to feed the parallel computations with data, the system needs to supply high memory bandwidth and hide memory latency.

The implementation section of this paper contains details of some of the techniques we used to provide enhanced throughput of computations and memory while meeting aggressive die-size and frequency goals. The primary purpose of this paper, however, is to describe key implementation techniques used in the processor and the rationale for their development.

ARCHITECTURE

The Pentium® III processor is the first implementation of the Internet SSE. The Internet SSE contains 70 new instructions and a new architectural state. It is the second significant extension of the instruction set since the 80386 and the first to add a new architectural state. MMX™ was the first significant instruction extension, but it did not add any new architectural state. The new instructions fall into different categories:

- SIMD FP instructions that operate on four single precision numbers
- scalar FP instructions
- cacheability instructions including prefetches into different levels of the cache hierarchy
- control instructions
- data conversion instructions

- new media extensions that are instructions such as the PSAD and the PAVG that accelerate encoding and decoding respectively

Adding the new state reduced implementation complexity, eased programming model issues, and allowed SIMD-FP and MMX technology or X87 instructions to be used concurrently. It also addressed ISV and OSV requests. All of the SSE State was separated from the X87-FP State; there is a dedicated interrupt vector to handle the numeric exceptions. There is also a new control/status register, MXCSR, which is used to mask/unmask numerical exception handling, to set rounding mode, to set flush to zero mode, and to view status flags. Applications often require both scalar and packed mode of operations. To address this issue, explicit scalar instructions (in the new SIMD-FP mode) were defined, which for the Pentium III processor execute only a single micro-instruction. Support is provided for two modes of FP arithmetic: IEEE compliant mode for applications that need exact single precision computation and portability and a Flush-to-Zero (FTZ) mode for high-performance real-time applications.

(Details of the instruction set are given in other papers in this issue of the *Intel Technology Journal*.)

IMPLEMENTATION

In this section we discuss some of the key features that we developed to increase FP and memory throughput on the Pentium® III processor. We then discuss a couple of techniques we developed to help provide an area-efficient solution. Figure 1 shows the block diagram of the P6 pipeline.

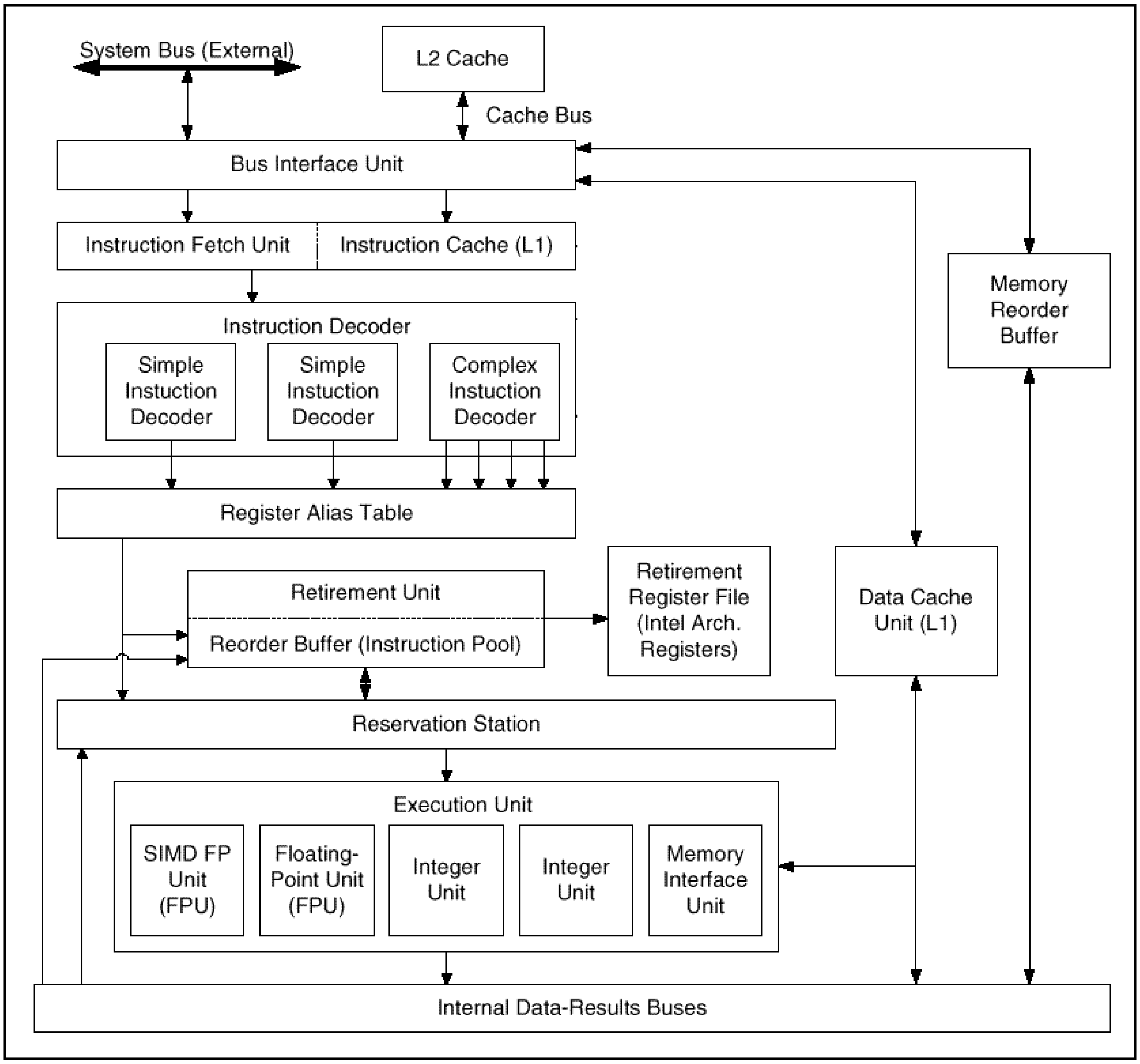

Figure 1. Functional Block Diagram of the P6 Family Processor Microarchitecture

Figure 2 shows the Dispatch/Execute units in the Pentium® II processor. An overview of the P6 architecture and the microarchitecture is given in [5] and [6] where you will also find a description of the blocks shown in Figures 1 and 2.

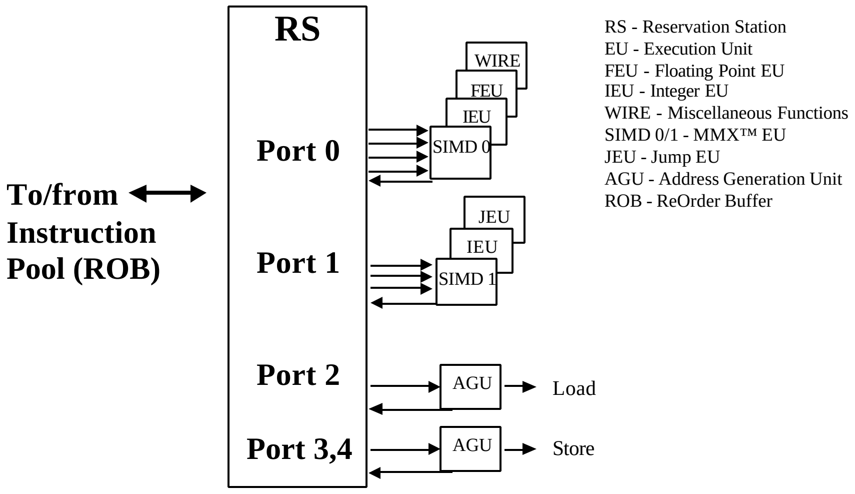

Figure 2: Pentium II Dispatch/Execute units

Implementation of Internet SSE Execution Units

The Internet SSE was implemented in the following way. The instruction decoder translates 4-wide (128-bit) Internet SSE instructions into a pair of 2-wide (64-bit) internal uops. The execution of the 2-wide uops is supported by 2-wide execution units. Some of the FP execution units were developed by extending the functionality of existing P6 FP units. The 2-wide units boost the performance to that of twice the Pentium II processor. Further, implementing the 128-bit instruction set on the 64-bit datapath limits the changes to the decoder and the utilization of existing and new execution units. We also implemented a few other features to improve the utilization of the FP hardware:

1. The adder and multiplier were placed on different ports. This allows for simultaneous dispatch of 2-wide addition and 2-wide multiplication operations. This boosts the peak performance two more times when compared to the Pentium II, and hence, it allows 2.2 GFLOP/sec peak at 550 MHz. The new units developed on the Pentium III and modified P6 units are shown in color in Figure 3.

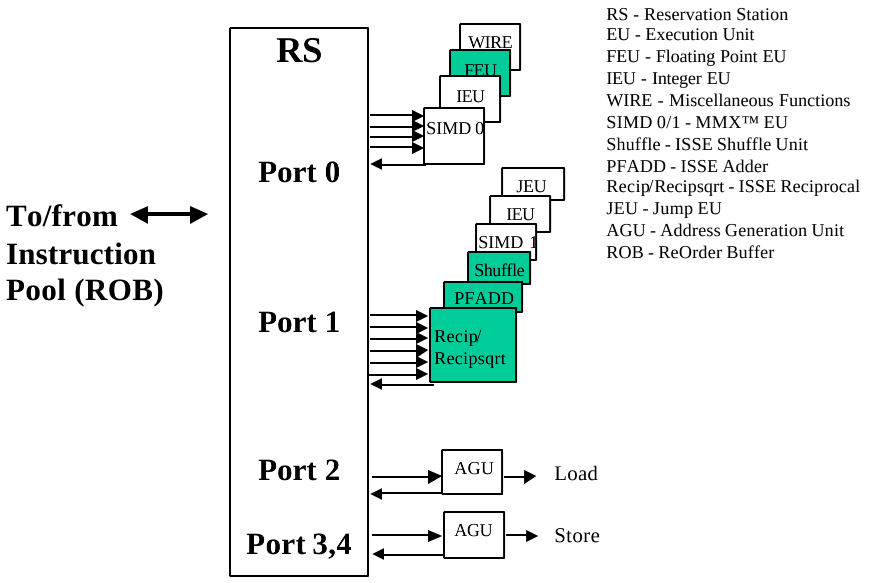

Figure 3: Pentium III Dispatch/Execute units

All the new units have been added on Port 1. The new operations executed on Port 0 have been accomplished by modifying existing units. The multiplier resides on Port 0 and is a modification of the existing FP multiplier. The new packed SP multiplication is performed with a throughput of two SP numbers each cycle with a latency of four cycles. (The X87 SP processor, on the other hand, had a throughput of a single SP number every two cycles and a latency of five cycles.) A new packed SP adder is provided on Port 1. The adder operates on two SP numbers with a throughput of one cycle and a latency of three cycles. The adder also executes all compare, subtract, min/max, and convert instructions. Essentially, we have assigned different units (and therefore different instructions) to Ports 0 and 1 to ensure that a full 4-wide peak execution can be achieved.

2. Hardware support is in place for data reorganization. Effective SIMD computations require adequate SIMD organization of data. For instance, the conventional representation of 3D data has the format of "(x, y, z, w)", where x, y, and z are the three coordinates of a vertex, and w is the perspective correction factor. In some cases, SIMD computations are more effective if the data are represented as vectors of equivalently named coordinates "(x1, x2, …), (y1, y2,…), (z1, z2,…), (w1, w2,…"). In order to support transformations between these type of data representations, the Internet SSE includes the set of data manipulation instructions. We considered the effective hardware support of these instructions to be an important method to improve the utilization of FP units, since it allows for less time to be spent in data reorganization. The new shuffle/logical unit serves this purpose. It shares Port 1 and executes the unpack high and unpack low, move, and logical uops. The 128-bit shuffle operation is performed through three uops: (1) copy temporary, (2) shuffle low, and (3) shuffle high. The shuffle unit also executes packed integer shuffle, PINSRW and PEXTRW, through sharing of the FP shuffle unit.

3. Data is copied faster. IA-32 instructions overwrite one of the operands of the instruction. We knew that this feature would add more MOVE instructions to the code. For instance, consider the following fragment:

Load memory operand A to register XMM0

Multiply XMM0 by memory operand B

The second instruction overwrites the content of the register XMM0. Hence, if the subsequent code uses the same memory operand A, then we need to either load it again from the memory (thus putting additional pressure on the load port, which is frequently used in multimedia algorithms), or we need to copy XMM0 to another register for later re-use. In order to facilitate the latter method, we implemented two move ports in the Pentium III processor.

Exceptions Handling

The implementation of a 128-bit processor via two 64-bit micro-ops raises the possibility of an exception occurring on either of the two independent micro-ops (uops). For instance, if the first uop retires while the second uop causes an exception, the architecturally visible 128-bit register is updated only partially, and it would cause an inconsistency in the architectural state of the machine. Retirement is the act of committing the results of an operation to architectural state. In order to provide "precise exceptions handling" we implemented the Check Next Micro-Operation (CNU) mechanism that prevents the retirement of the first uop if the second one causes an exception. The mechanism acts as follows. The first uop in a pair of two uops, which need to be treated as an atomic operation and/or data type, is marked with the CNU flow marker. The instruction decoder decodes the CNU marker and sends the CNU bit to the allocator. The allocator informs the ROB to set the CNU bit in the ROB entry allocated for this uop. The ROB is the reorder buffer and stores the micro-ops in the original program order. The ROB will delay retirement of the first uop until the second uop is also guaranteed to retire. Since this mechanism throttles the retirement, we implemented the following optimization. In the case where all exceptions are masked, each uop may be retired individually. Since multimedia software usually masks the exceptions, for all practical purposes, there is no loss of computational throughput.

Moreover, to maintain high computational throughput, we implemented in hardware the fast processing of masked exceptions, which happen routinely during execution of multimedia algorithms, such as overflow, divide by zero, and flush-to-zero underflow. These exceptions are handled by hardware through modifications to the rounder/writeback multiplexers.

CACHEABILITY IMPLEMENTATION

We now discuss the key changes in the memory implementation. These include support for the cacheability control features introduced by the Internet SSE instruction set. Support for byte masked writes, streaming stores, data prefetching, multiple WC buffers, and store fencing operations have been incorporated.

These are some of the aspects of the prefetch implementation on the Pentium III processor. Figure 4 shows the compulsory effect of two stalls that happen in the Pentium II if the load instruction misses the cache.

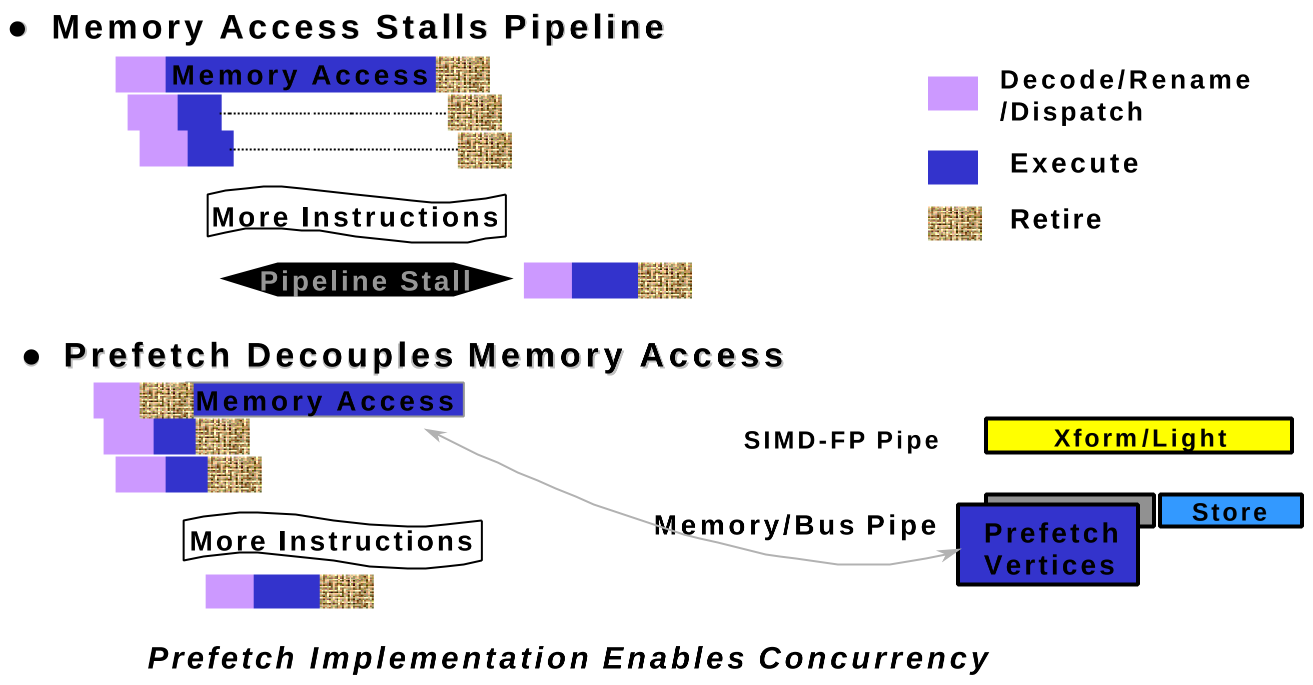

Figure 4: Prefetch implementation

The pipeline stall shown in the "memory access stalls pipeline" portion of Figure 4 is caused by the fact that the load instruction retires much later than the previous instruction. The instructions following the load are executed, but they cannot retire from the machine until the load returns the data. This is illustrated in the "memory access stalls pipeline" portion of the figure: the instructions subsequent to the memory access execute and then wait for the memory access to finish executing before they retire. Therefore, these instructions accumulate inside the machine. Eventually some of the key resources of the processor, such as the ROB that registers non-retired instructions, get saturated. This immediately stalls the front end of the processor since no new instructions can get the resources needed for execution. In the Pentium III processor, we removed this bottleneck for the prefetch instruction. We moved the retirement of the prefetch instruction much earlier in the pipe. This is illustrated in the "prefetch decouples memory access" portion of Figure 4. Here we observe that instructions after the memory access (in this case, Prefetch) are allowed to retire even though the memory access itself has not completed its execution. The prefetch is implemented such that even in the case of a cache miss, it retires almost immediately, and no retirement and resources saturation stalls occur due to memory access. As a result, we get much better concurrency of computations and memory access. This concept is called senior load and is shown in Figure 5.

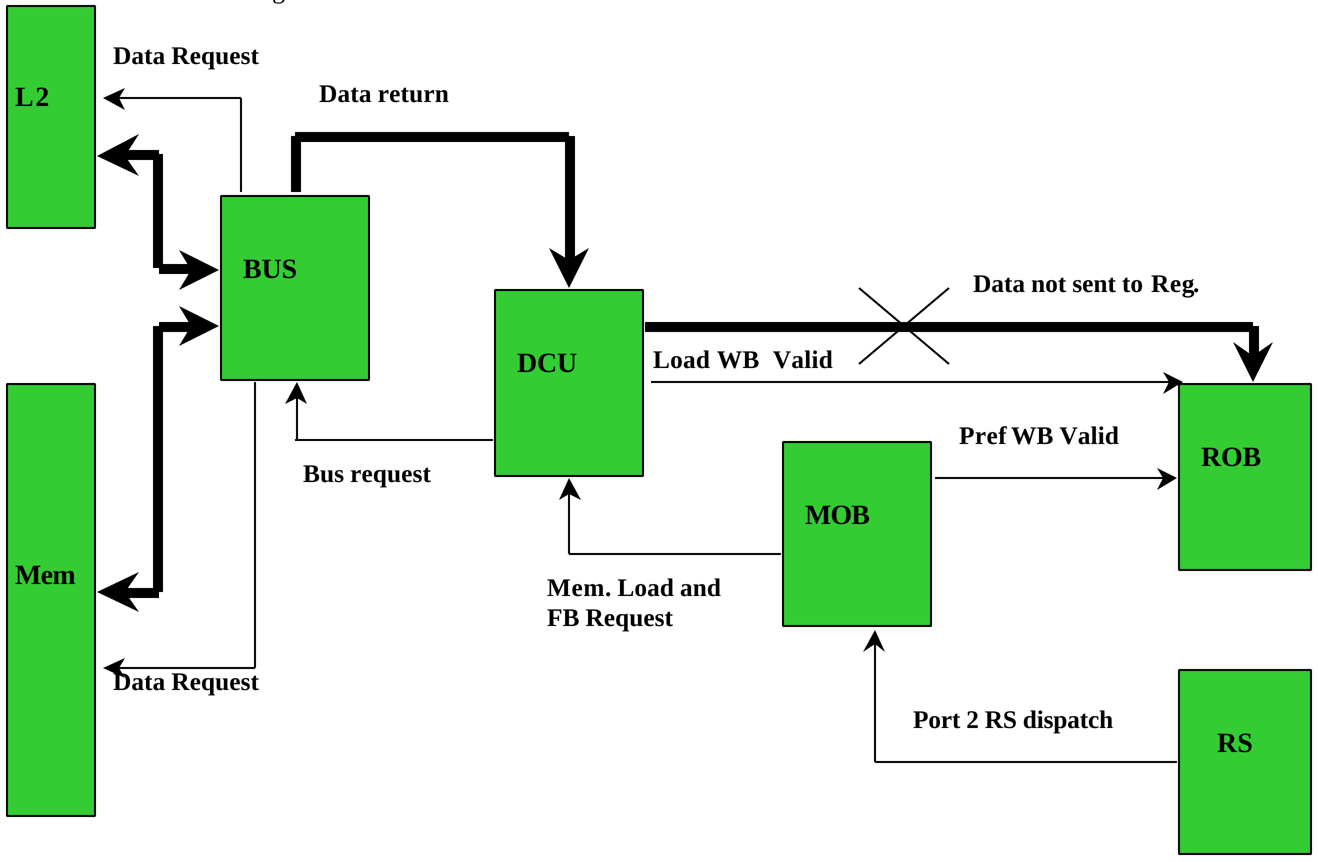

Figure 5: Senior load implementation

Figure 5 shows the differences in how readiness for retirement is signaled in load instructions and prefetch instructions. In the case of a load, the instruction is dispatched into the memory pipe after dispatch from the RS. If it misses the DCU, it is dispatched by the BUS unit. After the data returns from the L2 or the bus, the load is signaled as complete (Load WB valid), and the load and subsequent completed instructions are eligible to retire. In the case of the Prefetch, completion is signaled (by the Pref WB Valid) almost immediately after allocation into the MOB. The completion is not delayed until the data is actually fetched from the memory subsystem. The signaling of early completion permits the retirement of the load, and subsequent instructions occur earlier than in the case of the load, thus removing the resource stalls associated with memory access latency. The prefetch instructions can fetch data into different levels of the cache hierarchy. Also, streaming store instructions are supported. In many instances, applications stream data from memory and use them once without modifications. Using regular cache models will result in eviction of useful data from the cache. In the Discussion section of this paper, you will find details on cache management to increase performance.

We now discuss writes. The main issue with writes is that many applications such as video and 3D have a large working set. This working set doesn't fit into the cache. In such a situation, additional performance may actually be gained if the application bypasses the cache. In other words, the application should keep data in the memory. Hence it should write directly to the memory. This is what streaming stores are for.

The implementation of high write throughput is as important as high bandwidth memory reads. What we have done in the Pentium III processor is mainly two things: we have improved the write bandwidth and the write combining allocation and eviction. The bus write bandwidth was improved by 20%. The Pentium III processor now can saturate a 100 MHz bus with 800 MBs of writes. This was done by removing the dead cycle between back to back write combining writes.

We also improved the write buffers allocation mechanism in order to support this large write bandwidth. Since we re-use Pentium II fill buffers to do this, some further clarifications on the difference in buffers in the Pentium III and the Pentium II processors are in order. The differences are based on the very nature of SSE. Before the Pentium III processor, the architecture was mainly oriented to scalar applications. The purpose of fill buffers was to provide high instantaneous throughput caused by bursts of misses in scalar applications. The average bandwidth requirements were comparatively small, about 100 MB per second, but instantaneous requirements were high. SSE applications are streaming applications such as vector algorithms. Hence, the purpose of SSE buffers is to sustain high average throughput. In terms of overall (read+write) throughput requirements, the capacity of the Pentium II processor's fill buffers is enough for the Pentium III processor timeframe. But the allocation policy had to be improved in order to increase the efficiency with which this capacity is used. We therefore allowed a few write buffers at a time, and we provided fast draining of the buffers to reduce the occupancy time. The faster draining of the buffers and the efficient utilization techniques for multiple buffers are described below.

The Pentium III processor's write combining has been implemented in such a way that its memory cluster allows all four fill buffers to be utilized as write-combining fill buffers at the same time, as opposed to the Pentium II processor which allows just one. To support this enhancement, the following WC eviction conditions, as well as all Pentium™ Pro WC eviction policies, are implemented in the Pentium III processor:

- A buffer is evicted when all bytes are written (all dirty) to the fill buffer. Previously the buffer eviction policy was "resource Demand" drivem, i.e. a buffer gets evicted when DCU requests the allocation of new buffer.
- When all fill buffers are busy a DCU fill buffer allocation request, such as regular loads, stores, or prefetches requiring a fill buffer can evict a WC buffer even if it is not full yet.

Die-Frequency Efficient Implementation

In the Pentium III processor, a number of tradeoffs were made to remain within tight die-size constraints and to reach the frequency goals. Two of these tradeoffs are mentioned below:

- We merged the x87 multiplier with the packed FP multiplier. This helped significantly with die size and kept the loading on the ports down. Loading on the writeback busses was an important factor for us. The writeback busses have been significant speed paths in past implementations, and the addition of new units and logic for implementation of the Internet SSE would have made the situation worse. This was an area of focus from the very inception of the project. We also considered merging the x87 adder with the packed FP adder, but we did not follow through with this because of schedule tradeoffs.
- We reused the multiplier's Wallace tree to do the PSAD. The PSAD, computing the absolute difference of packed MMX values, was implemented with three uops: computation of the difference, computation of the absolute value, and the sum of the absolutes. The sum of the absolutes was computed in the multiplier's Wallace tree. The bytes that needed to be added were fed into the tree that normally sums the multiplication partial products. The reuse of this logic enabled us to implement the instruction with a very small die and frequency impact. Alternatives to execute the instruction with reasonable performance were significantly more expensive.

RESULTS

We would like to outline two main results: the method of implementation of 4-wide ISA via concurrent dispatch of two 2-wide streams of computations, and decoupled execution of the streams of computations and memory.

Implementation of a 4-wide ISA based on a 2-wide data path provides a good tradeoff between die size and performance. From the performance standpoint, this approach may raise the question of how is 2-wide implementation of a 4-wide ISA different from a 2-wide implementation of a 2-wide ISA The difference comes from the fact that 4-wide ISA has twice as many registers. Consider a loop of instructions that uses eight registers. The loop coded in 4-wide ISA can be viewed as the loop coded in 2-wide ISA, which is then twice unrolled and software-pipelined. In general, the explicit loop unrolling improves performance. In particular, it delivers additional improvement even in the out-of-order architecture, since it exposes more parallelism to the machine. The same loop in a 2-wide ISA cannot be unrolled since the loop uses all the available registers. Hence, in the case of a 4-wide ISA, the performance benefits come from two sources: internal out-of-order scheduling plus the explicit loop unrolling. In the case of 2-wide ISA, the benefits come from the internal out-of-order execution only.

The main reason behind the streaming architecture is to meet the requirements of multimedia performance by providing concurrent processing of data streams. From the implementation standpoint, it means that the processor should provide concurrent execution of the computational stream and stream of memory accesses.

The P6 microarchitecture extracts concurrency of memory accesses and computations. However, the explicit prefetch instructions allowed us to completely decouple the data fetch and retirement of subsequent instructions. Hence, the throughput of each of these streams can reach almost the theoretical maximum possible for the given task. As a result, the maximum throughput that can be reached with the Pentium III processor for the given tasks is equal to the lowest of maximum memory throughput and maximum computational throughput of this task.

Since we increased the effective memory throughput, we had to balance the throughput of the processor buffering subsystem and bus throughput. We did not implement new buffers but rather we implemented a few methods to improve the utilization of the existing buffers and improve the write throughput of the external bus. This allowed us to pay a negligible die-size price for performance balancing the memory datapath.

DISCUSSION

In parallel with the development of the Pentium III processor, we developed programming models that allow us to utilize the potential gain of this implementation in real-world applications. In order to outline the details of these models, we discuss three types of multimedia applications:

1. *Compute bound applications such as AC3 Audio*. These applications exhibit fairly small memory bandwidth requirements, but need large computational throughput. In the Pentium III processor these applications are supported by high throughput FP units. In order to utilize the computational power of these units, programmers are supposed to use SSE optimization tools described in [2].

2. *Memory bound applications such as 3D imaging*. The distinct feature of these applications is a fairly large working set. Because of this, the data of these applications usually are in the memory, and the cache doesn't work as well as it does for compute-bound applications. Moreover, in some cases, it is even better to bypass the cache. In these cases, the software can keep data in the memory and utilize the high memory throughput and concurrency described above. In order to utilize these features, it is recommended that a software developer identify incoming and outgoing streams, program these streams using prefetch and streaming store instructions in order to ensure that these streams are fetched/stored directly from/to memory without excessive internal caching. The paper in [3] describes the details of some of these techniques by describing an on-line driver approach for 3D geometry.

   Additional techniques for prefetching include optimizing the length of the data streams to reduce the degree of memory access de-pipelining. This may happen in the beginning of a data stream due to unutilized prefetches. Reference [3] describes a DrawMultiPrimitive technique that demonstrates the details of this programming model.

   The details of these methods are described in [3]. 3D processing is an example. Software implementation of this model allowed us to achieve twice the speedup at the application level.

3. *Mixed class such as video encoding*. These applications usually have few working sets; some of them fit into cache, some of them do not. The strategy of implementation support and programming model for these applications is based on the combination of the above methods. For these types of applications, it is important to separate frequently reused working sets from ones that are used less frequently, and to build a caching strategy based on the frequency of reuse.

   For instance, the MPEG2 Encoder [4] processes the data shown in Figure.6: color and brightness data of Intrinsics frames (I-frame), color and brightness data of Bi-directional frames (B-frames), and downsampled data.

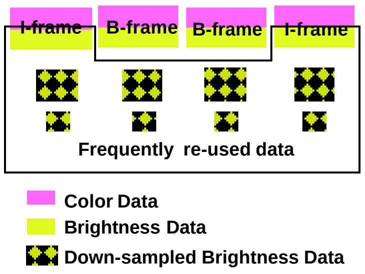

Figure 6: Data reuse in MPEG-2 encoder

According to the encoding algorithm, the I-frame brightness data and downsampled data are processed a few times more frequently than I-frame color data and B-frame data. Hence, the caching strategy for this application is to keep the former data in the cache, and the latter in memory. Figure. 7 shows the difference between two methods of data placement in the cache/memory hierarchy: non-controlled caching in the case of regular IA-32 caching, and software controlled caching that can be achieved in the Pentium III using Internet SSE streaming store and prefetch instructions. Though the I-frame color data and B-frame data are in the memory, the high throughput memory prefetch/store instructions allow us to hide the latency of the data fetch.

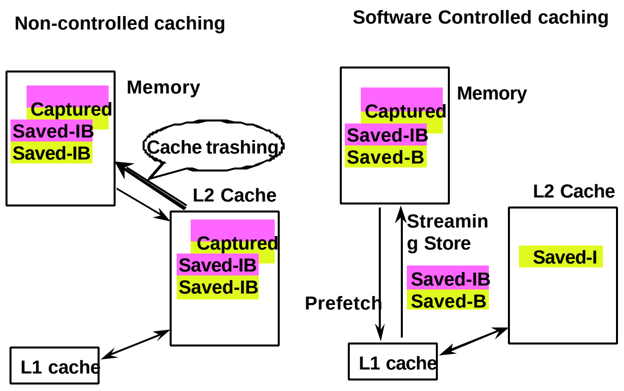

Figure 7: Software controlled caching vs. non-controlled caching in the MPEG-2 encoder

The combination of this model with the application of the PSAD instruction (in the motion estimation algorithm) allowed us to reach MPEG2 real-time software encoding in the Pentium III processor.

CONCLUSION

The Pentium III processor is now in production on frequencies of up to 550 MHz. The 70 new instructions that were added were done at a cost of an additional ~10% in die size. The features that we have described have enabled the Pentium III processor to achieve superior multimedia performance. One more important feature is that our fairly straightforward implementation of the 4-wide Internet SSE and concurrency of the computational and memory streams allows for further performance scalability of SSE applications moving toward higher frequencies.

ACKNOWLEDGMENTS

The development of the Pentium III processor is the result of work done by a number of people in MPG-FM architecture, design, and validation. A lot of the work described in this paper was done by the MPG-FM architecture group. We acknowledge the guidance of Srinivas Raman during the definition and execution of the product. We also thank Dave Papworth, Glenn Hinton, Pete Mcwilliams, Bob Colwell, Ticky Thakkar, and Tom Huff for their input during the process.

REFERENCES

[1] Narayanan Iyer, Mohammad Abdallah, S.Maiyuran, Vivek Garg, S.Spangler, "Katmai Features POR," Intel internal document.

[2] Joe Wolf, "Programming Methods for the Pentium® III Processor's Streaming SIMD Extensions Using the Vtune^{TM} Performance Enhancement Environment," *Intel Technology Journal*, Q2, 1999.

[3] Paul Zagacki, Deep Buch, Emile Hsieh, Hsien-Hsin Lee, Daniel Melaku, Vladimir Pentkovski, "Architecture of a 3D Software Stack for Peak Pentium® III Processor Performance," *Intel Technology Journal*, Q2, 1999.

[4] James Abel et al, "Applications Tuning for Streaming SIMD Extensions," *Intel Technology Journal*, Q2, 1999.

[5] David B. Papworth, "Tuning the Pentium Pro Microarchitecture," IEEE Micro 1996.

[6] Intel, *Pentium Pro Family Developer's Manual*, Intel literature.

[7] Ticky Thakkar et al, "The Internet Streaming SIMD Extensions," *Intel Technology Journal*, Q2, 1999.

AUTHORS' BIOGRAPHIES

Jagannath Keshava has been with Intel since 1990. He is currently working in the MPG-Folsom Architecture Group

on the definition of integrated microprocessors for the Value PC segment. He led the Pentium III processor definition and microarchitecture validation teams. In the past, he has held lead positions in design, microarchitecture, and validation on the Pentium II and i960® microprocessor groups. Jagannath has an M.S. degree in computer engineering from the University of Texas, Austin.

His e-mail is jagannath.keshava@intel.com

Vladimir Pentkovski is a Principal Engineer in the Microprocessor Product Group in Folsom. He was one of the architects in the core team, which defined the Internet Streaming SIMD Extensions of IA-32 architecture. Vladimir led the development of Pentium III processor architecture and performance analysis. Previously he led the development of compilers and software and hardware support for programming languages for Elbrus multi-processor computers in Russia. Vladimir holds a Doctor of Science degree and Ph.D. degree in computer science and engineering from Russia. His e-mail is vladimir.m.pentkovski@intel.com.