A 0.18-μm CMOS IA-32 Processor With a 4-GHz Integer Execution Unit

Glenn Hinton, Michael Upton, David J. Sager, *Member, IEEE*, Darrell Boggs, Douglas M. Carmean, Patrice Roussel, Terry I. Chappell, *Senior Member, IEEE*, Thomas D. Fletcher, Mark S. Milshtein, *Member, IEEE*, Milo Sprague, Samie Samaan, and Robert Murray

***Abstract*—This paper describes the main features and functions of the Pentium® 4 processor microarchitecture. We present the front-end of the machine, including its new form of instruction cache called the trace cache, and describe the out-of-order execution engine, including a low latency double-pumped arithmetic logic unit (ALU) that runs at 4 GHz. We also discuss the memory subsystem, including the low-latency Level 1 data cache that is accessed in two clock cycles. We then describe some of the key features that contribute to the Pentium® 4 processor's floating-point and multimedia performance. We provide some key performance numbers for this processor, comparing it to the Pentium® III processor.**

***Index Terms*—Adders, clocks, CMOS digital integrated circuits, computer architecture, high-speed integrated circuits, memory architecture, pipelines.**

I. Introduction

THE Pentium® 4 processor is Intel's newest microprocessor, introduced at 1500 MHz in November of 2000. It implements a completely new microarchitecture that features significantly higher clock rates and performance compared to the previous processor generation. This paper provides an in-depth examination of the features and functions of the Intel Pentium® 4 Processor.

The Pentium® 4 processor has 42 million transistors implemented on Intel's 0.18-μm CMOS process [1]. Table I summarizes the properties of the six levels of aluminum interconnect. The processor has a die size of 217 mm^2, and it consumes 55 W of power at 1500 MHz and 1.75 V. Its 3.2-GB/s system bus helps provide the high data bandwidths needed to supply data to today's and tomorrow's demanding applications. It adds 144 new 128-b Single Instruction Multiple Data (SIMD) instructions called SSE2 (Streaming SIMD Extension 2) that improve performance for multimedia, content creation, and scientific and engineering applications.

II. Clock Rates

Processor microarchitectures can be pipelined to different degrees. The degree of pipelining is a primary microarchitectural decision. The final frequency of a specific processor pipeline on a given silicon process technology depends heavily on how deeply the processor is pipelined. When designing a new processor, a key design decision is the target design frequency of operation. The frequency target determines how many gates of logic can be included per pipeline stage in the design. This then helps determine how many pipeline stages there are in the machine.

TABLE I
Metal Properties of 0.18-μm Process

	Pitch	Thickness	Aspect ratio
metal 1	500 nm	480 nm	1.9
metal 2,3	640 nm	700 nm	2.2
metal 4	1080 nm	1080 nm	2.0
metal 5	1600 nm	1600 nm	2.0
metal 6	1720 nm	1720 nm	2.0

There are tradeoffs when designing for higher clock rates. Higher clock rates need deeper pipelines, so the efficiency at the same clock rate goes down. Deeper pipelines make many operations take more clock cycles, such as mispredicted branches and cache misses, but usually more than make up for the lower per-clock efficiency by allowing the design to run at a much higher clock rate. For example, a 50% increase in frequency might buy only a 30% increase in net performance, but this frequency increase still provides a significant overall performance increase. High-frequency design also depends heavily on circuit design techniques, design methodology, design tools, silicon process technology, power and thermal constraints, etc. At higher frequencies, clock skew and jitter and latch delay become a much bigger percentage of the clock cycle, reducing the percentage of the clock cycle usable by actual logic. The deeper pipelines make the machine more complicated and require it to have deeper buffering to cover the longer pipelines.

A. Historical Trend of Processor Frequencies

Fig. 1 shows the relative clock frequency of Intel's last six processor cores. The vertical axis shows the relative clock frequency, and the horizontal axis shows the various processors relative to each other. As shown in Fig. 1, the 286, 386, 486, and Pentium® processors had similar pipeline depths, therefore, they would run at similar clock rates if they were all implemented on the same silicon process technology. They all had a similar number of gates of logic per clock cycle. The Pentium® III microarchitecture lengthened the processor pipelines, allowing fewer gates of logic per pipeline stage, which delivered significantly higher frequency and performance. The Pentium® III microarchitecture approximately doubled the number of pipeline stages compared to the earlier processors and was

Manuscript received March 26, 2001; revised June 15, 2001.

The authors are with Intel Corporation, Hillsboro, OR 97124 USA (e-mail: mike.upton@intel.com).

Publisher Item Identifier S 0018-9200(01)08218-X.

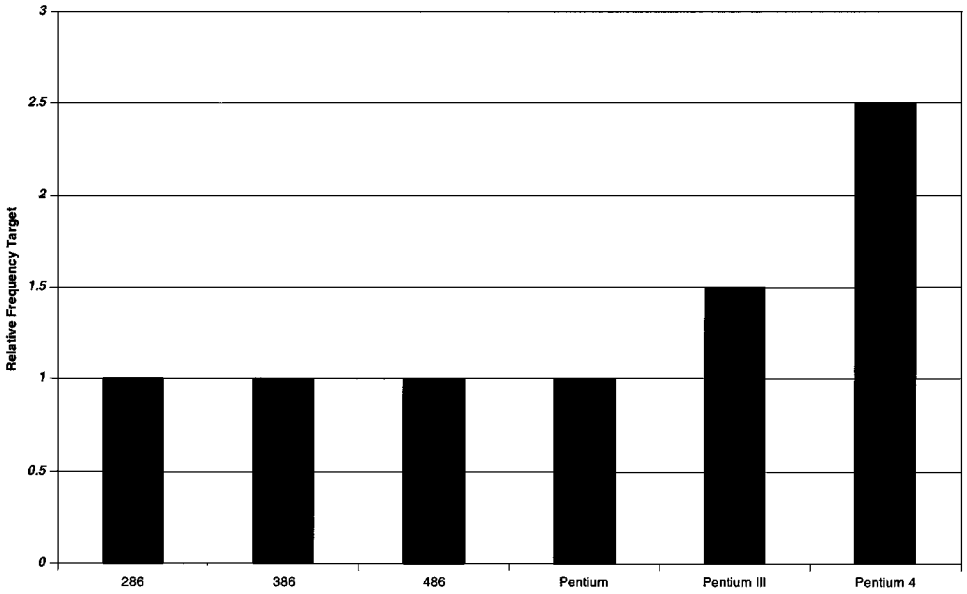

Fig. 1. Relative frequencies of Intel processors if fabricated on same process.

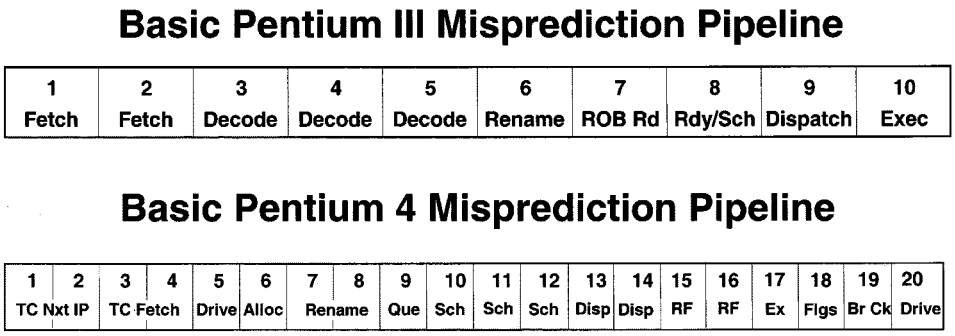

Fig. 2. Branch misprediction pipeline.

able to achieve about a 1.5 times higher frequency on the same process technology.

The Pentium® 4 microarchitecture was designed to have an even deeper pipeline (about two times the Pentium® III microarchitecture) with even fewer gates of logic per clock cycle to allow an industry-leading microprocessor clock rate in 0.18-μm CMOS. Compared to the Pentium® III family of processors, the Pentium® 4 processor was designed with a greater than 1.6 times higher frequency target for its main clock rate, on the same process technology. This allows it to operate at a much higher frequency than the Pentium® III family of processors on the same silicon process technology. At its introduction in November 2000, the Pentium® 4 processor was at 1.5 times the frequency of the Pentium® III processor. Over time, this frequency delta will increase as the Pentium® 4 processor manufacturing matures, and critical paths are eliminated by design revisions.

Different parts of the Pentium® 4 processor run at different clock frequencies. The frequency of each section of logic is set to be appropriate for the performance it needs to achieve. The highest frequency section (fast clock) was set equal to the speed of the critical ALU-bypass execution loop that is used for most instructions in integer programs. Most other parts of the chip run at half of the 4-GHz fast clock, since this makes these parts much easier to design. A few sections of the chip run at a quarter of this fast-clock frequency, making them also easier to design. The 2-GHz clock is distributed across the die, and the other needed frequencies are generated locally [2]. The bus logic runs at 100 MHz, to match the system bus needs.

As an example of the pipelining differences, Fig. 2 shows a key pipeline in both the Pentium® III and the Pentium® 4 processors: the mispredicted branch pipeline.

This pipeline covers the cycles it takes a processor to recover from a branch that went a different direction than the early fetch hardware predicted at the beginning of the machine pipeline. As shown, the Pentium® 4 processor has a 20-stage misprediction pipeline, while the Pentium® III microarchitecture has a 10-stage misprediction pipeline. By dividing the pipeline into smaller pieces, doing less work during each pipeline stage (fewer gates of logic), the clock rate can be much higher.

III. Pentium® 4 Microarchitecture

Fig. 3 shows a detailed block diagram of the organization of the Pentium® 4 processor. The top-left portion of the diagram shows the front-end part of the machine. The middle of the diagram illustrates the out-of-order buffering logic, and the bottom of the diagram shows the integer and floating-point execution

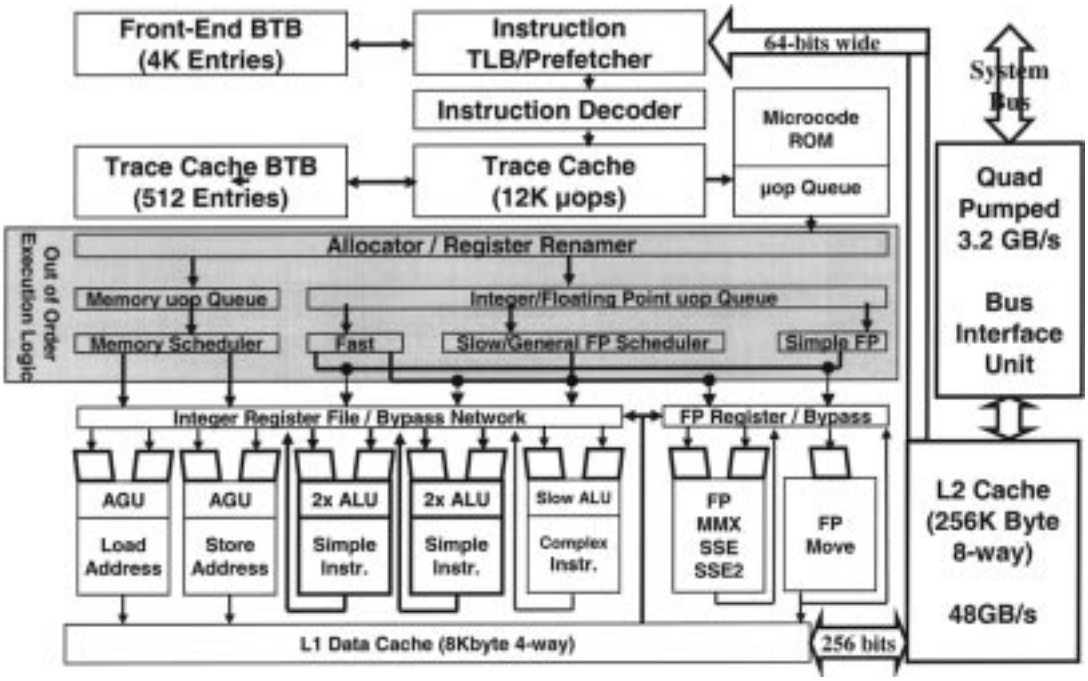

Fig. 3. Pentium® 4 processor microarchitecture organization.

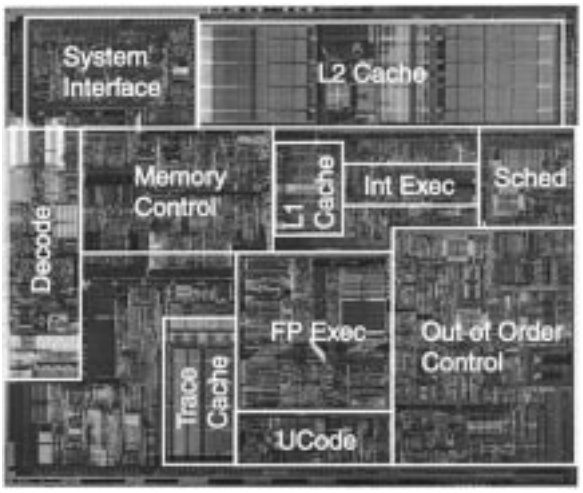

Fig. 4. Processor die photograph.

units and the L1 data cache. On the right of the diagram is the memory subsystem. Fig. 4 is a die photograph showing the physical organization of the processor.

A. Front-End

The front-end of the Pentium® 4 processor consists of several units, as shown in the upper part of Fig. 3. It has the instruction translation lookaside buffer (ITLB), the front-end branch predictor (labeled here Front-End BTB), the IA-32 instruction decoder, the trace cache, and the microcode ROM.

1) Trace Cache: The trace cache is the primary or Level 1 (L1) instruction cache of the Pentium® 4 processor and delivers up to three micro-operations (UOPs) per clock to the out-of-order execution logic. Most instructions in a program are fetched and executed from the trace cache. Only when there is a trace cache miss does the machine fetch and decode instructions from the Level 2 (L2) cache. This occurs about as often as previous processors miss their L1 instruction cache. The trace cache has a capacity to hold up to 12K UOPs. It has a similar hit rate to an 8K–16K-byte conventional instruction cache.

IA-32 instructions are cumbersome to decode. The instructions have a variable number of bytes and have many different options. The instruction decoding logic needs to sort this all out and convert these complex instructions into simple UOPs that the machine knows how to execute. This decoding is especially difficult when trying to decode several IA-32 instructions each clock cycle when running at the high clock frequency of the Pentium® 4 processor. A high-bandwidth IA-32 decoder, capable of decoding several instructions per clock cycle, takes several pipeline stages to do its work. When a branch is mispredicted, the recovery time is much shorter if the machine does not have to redecode the IA-32 instructions needed to resume execution at the corrected branch target location. By caching the UOPs of the previously decoded instructions in the trace cache, instructions bypass the instruction decoder most of the time, thereby reducing misprediction latency and allowing a simpler decoder that processes only one IA-32 instruction per clock cycle.

The trace cache takes the already decoded UOPs from the IA-32 Instruction Decoder and assembles or builds them into program-ordered sequences of UOPs called traces. It packs the UOPs into groups of six UOPs per trace line. There can be many trace lines in a single trace. These traces consist of UOPs running sequentially down the predicted path of the IA-32 program execution. This allows the target of a branch to be included in the same trace cache line as the branch itself even if the branch and its target instructions are thousands of bytes apart in the program.

Conventional instruction caches typically provide instructions up to and including a taken branch instruction, but

none after it during that clock cycle. If the branch is the first instruction in a cache line, only the single branch instruction is delivered that clock cycle. Also, conventional instruction caches often add a clock delay getting to the target of the taken branch, due to delays getting through the branch predictor and then accessing the new location in the instruction cache. The trace cache avoids both aspects of this instruction delivery delay for programs that fit well in the trace cache.

The trace cache has its own branch predictor that directs where instruction fetching needs to go next in the trace cache. This trace cache predictor (labeled Trace BTB in Fig. 3) is smaller than the front-end predictor, since its main purpose is to predict the branches in the subset of the program that is currently in the trace cache. The branch prediction logic includes a 16-entry return address stack to efficiently predict return addresses, because often the same procedure is called from several different call sites. The trace cache BTB, together with the front-end BTB, use an improved branch prediction algorithm that reduces the branch misprediction rate by about 1/3 compared to the predictor in the Pentium® III microarchitecture.

2) Microcode ROM: Near the trace cache is the microcode ROM. This ROM is used for complex IA-32 instructions, such as string move, and for fault and interrupt handling. When a complex instruction is encountered, the trace cache jumps into the microcode ROM which then issues the UOPs needed to complete the operation. After the microcode ROM finishes sequencing UOPs for the current IA-32 instruction, the front end of the machine resumes fetching UOPs from the trace cache.

The UOPs that come from the trace cache and the microcode ROM are buffered in a simple in-order UOP queue that helps smooth the flow of UOPs going to the out-of-order execution engine.

3) ITLB and Front-End BTB: The IA-32 Instruction TLB and front-end BTB, shown at the top of Fig. 3, steer the front-end when the machine misses the trace cache. The ITLB translates the linear instruction pointer addresses given to it into physical addresses needed to access the L2 cache. The ITLB also performs page-level protection checking.

Hardware instruction prefetching logic associated with the front-end BTB fetches IA-32 instruction bytes from the L2 cache that are predicted to be executed next. The fetch logic attempts to keep the instruction decoder fed with the next IA-32 instructions the program needs to execute. This instruction prefetcher is guided by the branch prediction logic (branch history table and branch target buffer listed here as the front-end BTB) to know what to fetch next. Branch prediction allows the processor to begin fetching and executing instructions long before the previous branch outcomes are certain. The front-end branch predictor is quite large—4K branch target entries—to capture most of the branch history information for the program. If a branch is not found in the BTB, the branch prediction hardware statically predicts the outcome of the branch based on the direction of the branch displacement (forward or backward). Backward branches are assumed to be taken and forward branches are assumed to not be taken.

4) IA-32 Instruction Decoder: The instruction decoder receives IA-32 instruction bytes from the L2 cache 64 bits at a time and decodes them into primitives that the machine knows how to execute, called UOPs. This single instruction decoder can decode at a maximum rate of one IA-32 instruction per clock cycle. Many IA-32 instructions are converted into a single UOP, and others need several UOPs to complete the full operation. If more than four UOPs are needed to complete an IA-32 instruction, the decoder sends the machine into the microcode ROM to do the instruction. Most instructions do not need to jump to the microcode ROM to complete. An example of a many-UOP instruction is string move, which could have thousands of UOPs.

B. Out-of-Order Execution Logic

The out-of-order execution engine consists of the allocation, renaming, and scheduling functions. This part of the machine reorders UOPs to allow them to execute as quickly as their input operands are ready.

The processor attempts to find as many instructions as possible to execute each clock cycle. The out-of-order execution engine will execute as many ready instructions as possible each clock cycle, even if they are not in the original program order. By looking at a larger number of instructions from the program at once, the out-of-order execution engine can usually find more ready-to-execute independent instructions to begin. The Pentium® 4 microarchitecture has much deeper buffering than the Pentium® III microarchitecture to allow this. It can have up to 126 instructions in flight at a time and have up to 48 loads and 24 stores allocated in the machine at a time.

1) The Allocator: The out-of-order execution engine has several buffers to perform its reordering, tracking, and sequencing operations. The allocator logic allocates many of the key machine buffers needed to execute each UOP. If a needed resource, such as a register file entry, is unavailable for one of the three UOPs coming to the allocator this clock cycle, the allocator will stall this part of the machine. When the resources become available, the allocator assigns them to the requesting UOPs and allows these satisfied UOPs to flow down the pipeline to be executed. The allocator allocates a reorder buffer (ROB) entry, which tracks the completion status of one of the 126 UOPs that could be in flight simultaneously in the machine. The allocator also allocates one of the 128 integer or 128 floating-point register entries for the result data value of the UOP, and possibly a load or store buffer used to track one of the 48 loads or 24 stores in the machine pipeline. In addition, the allocator allocates an entry in one of the two UOP queues in front of the instruction schedulers.

2) Register Renaming: The register renaming logic renames the logical IA-32 registers, such as the register named EAX, onto the processor's 128-entry physical register file. This allows the small eight-entry architecturally defined IA-32 register file to be dynamically expanded to use the 128 physical registers in the Pentium® 4 processor. This renaming process removes false conflicts caused by multiple instructions creating their simultaneous but unique versions of a register such as EAX. There could be dozens of unique instances of EAX in the machine pipeline at one time. The renaming logic remembers the most current version of each register, such as EAX, in the Register Alias Table (RAT), so that a new instruction coming down the

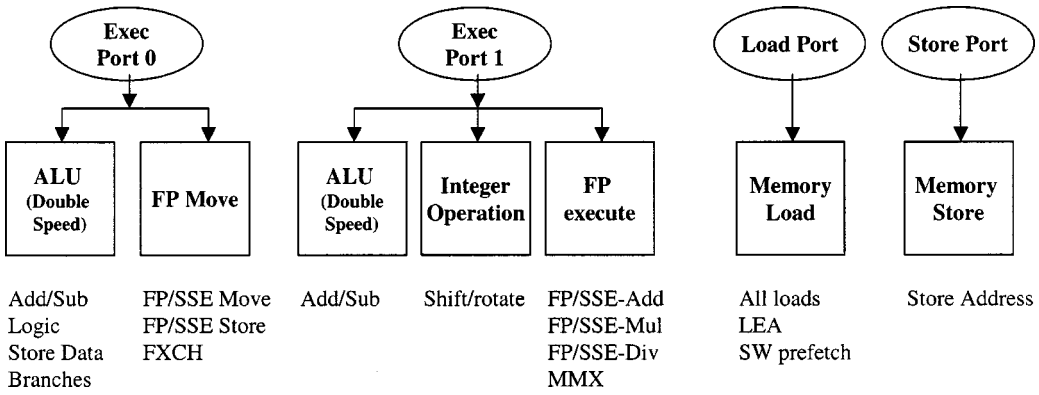

Fig. 5. Pentium® 4 processor dispatch ports.

pipeline can know where to get the correct current instance of each of its input operand registers.

3) UOP Scheduling: The UOP schedulers determine when a UOP is ready to execute by tracking its input register operands. This is the heart of the out-of-order execution engine. The UOP schedulers are what allow the instructions to be reordered to execute as soon as they are ready, while still maintaining the correct dependencies from the original program. The Pentium® 4 microarchitecture has two sets of structures to aid in UOP scheduling: the UOP queues and the actual UOP schedulers.

There are two UOP queues, one for memory operations (loads and stores) and one for nonmemory operations. Each of these queues stores the UOPs in strict first-in-first-out (FIFO) order with respect to the UOPs in its own queue, but each queue is allowed to be read out-of-order with respect to the other queue. This allows the dynamic out-of-order scheduling window to be larger than just having the UOP schedulers do all the reordering work.

There are several individual UOP schedulers that are used to schedule different types of UOPs for the various execution units on the Pentium® 4 processor, as shown in Fig. 5. These schedulers determine when UOPs are ready to execute based on the readiness of their dependent input register operand sources and the availability of the execution resources the UOPs need to complete their operation.

These schedulers are tied to four different dispatch ports. There are two execution unit dispatch ports labeled port 0 and port 1 in Fig. 5. These ports can dispatch up to two operations each main processor clock cycle. Multiple schedulers share each of these two dispatch ports. The fast ALU schedulers can schedule on each half of the main clock cycle while the other schedulers can only schedule once per main processor clock cycle. They arbitrate for the dispatch port when multiple schedulers have ready operations at once. There is also a load and a store dispatch port that can dispatch a ready load and store each clock cycle. Collectively, these UOP dispatch ports can dispatch up to six UOPs each main clock cycle. This dispatch bandwidth exceeds the front-end and retirement bandwidth of three UOPs per clock to allow for peak bursts of greater than three UOPs per clock and to allow higher flexibility in issuing UOPs to different dispatch ports. Fig. 5 also shows the types of operations that can be dispatched to each port during each clock cycle.

C. Integer and Floating-Point Execution Units

The execution units are where the instructions are actually executed. The execution units are designed to optimize overall performance by handling the most common cases as fast as possible. There are several different execution units in the Pentium® 4 microarchitecture. The units used to execute integer operations include the low-latency integer ALUs, the complex integer instruction unit, the load and store address generation units, and the L1 data cache.

Floating-point (x87), MMX™ technology, SSE (Streaming SIMD Extension), and SSE2 (Streaming SIMD Extension 2) operations are executed by the two floating-point execution blocks. MMX instructions are 64-b packed integer SIMD operations that operate on 8-, 16-, or 32-b operands. The SSE instructions are 128-b packed IEEE single-precision floating-point operations. The Pentium® 4 processor adds new forms of 128-b SIMD instructions called SSE2. The SSE2 instructions support 128-b packed IEEE double-precision SIMD floating-point operations and 128-b packed integer SIMD operations. The packed integer operations support 8-, 16-, 32-, and 64-b operands [4].

The integer and floating-point register files sit between the schedulers and the execution units. There is a separate 128-entry register file for each of the integer and the floating-point/SSE operations. The register files also each have a multiclock bypass network that bypasses or forwards just-completed results, which have not yet been written into the register file, to the new dependent UOPs. This multiclock bypass network is needed because of the very high frequency of the design.

1) Low-Latency Integer ALU: The Pentium® 4 processor execution units are designed to optimize overall performance by handling the most common cases as fast as possible. The Pentium® 4 processor can do fully dependent ALU operations at twice the main clock rate. Fig. 6 shows many of the key loops that determine the performance of the Pentium® 4 Processor. The ALU-bypass loop is a key closed loop in the processor pipeline. Approximately 60%–70% of all UOPs in typical integer programs use this key integer ALU loop. Executing these operations at half the latency of the main clock helps speed up program execution for most programs. Doing the ALU operations in one half a clock cycle does not buy a 2× performance increase, but it does improve the performance for most integer applications.

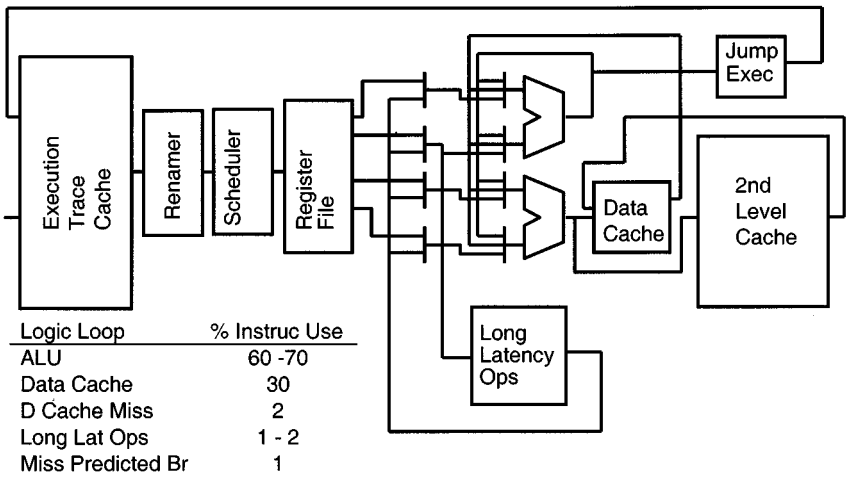

Fig. 6. Important logic loops.

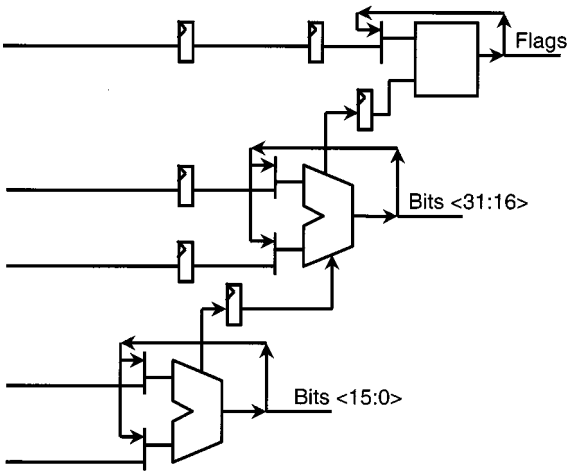

Fig. 7. Staggered ALU design in 16-b slices.

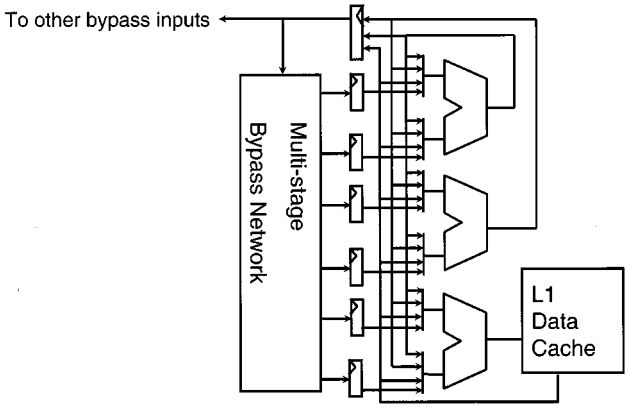

Fig. 8. ALU inputs, loading, and bypass loop.

This high-speed ALU core is kept as small as possible to minimize the metal length and loading. Only the essential hardware necessary to perform the frequent ALU operations is included in this high-speed ALU execution loop. Functions that are not used very frequently in most integer programs are not put in this low-latency ALU loop, but are put elsewhere. Some examples of integer execution hardware put elsewhere are the multiplier, shifts, flag logic, and branch processing.

Determining if the data obtained was correct, or if it was a cache miss, TLB miss, or protection violation, or if it had some form of interaction with other things in the memory system, is more complex and takes longer than getting the data. Hence, the first-level data cache data path is in the high-speed core, but the more extensive logic for determining if the data obtained is correct is placed elsewhere.

The processor does ALU operations with an effective latency of one-half of a clock cycle. It does this operation in a sequence of three fast clock cycles (the fast clock runs at $2\times$ the main clock rate), as shown in Fig. 7. In the first fast clock cycle, the low-order 16 bits are computed and are immediately available to feed the low 16 bits of a dependent operation the very next fast clock cycle. The high-order 16 bits are processed in the next fast cycle, using the carryout just generated by the low 16-b operation. This upper 16-b result will be available to the next dependent operation exactly when needed. This is called a staggered add. The ALU flags are processed in the third fast cycle. This staggered add means that only a 16-b adder and its input muxes need to be completed in a fast clock cycle. The low-order 16 bits are needed at one time in order to begin the access of the L1 data cache when used as an address input.

To maximize operating frequency, the first-level bypass multiplexers are restricted to four inputs (Fig. 8). These four inputs include the two ALU outputs, the output of the first level data cache, and the output of a multistage bypass network that was resolved during the previous clock. Similarly, the output loading of the execution units is strictly controlled. The execution units drive seven output loads: the six first-level bypass multiplexers (two per adder) and a latch that buffers all other uses of the execution unit output.

The ALU includes selectable inversion of each source of one operand and selectable carry blocking at byte boundaries, as well as sign extension and a full set of logic functions. The ALU adder is a propagate–generate–kill addition algorithm (PGK) true-and-complement design. A jam latch at the final domino stage provides the required hold time to close the loop and redrive the inputs of the ALUs themselves and the address generator for the load. The add loop, including the input multiplexer with built-in selectable inverter, the selectable

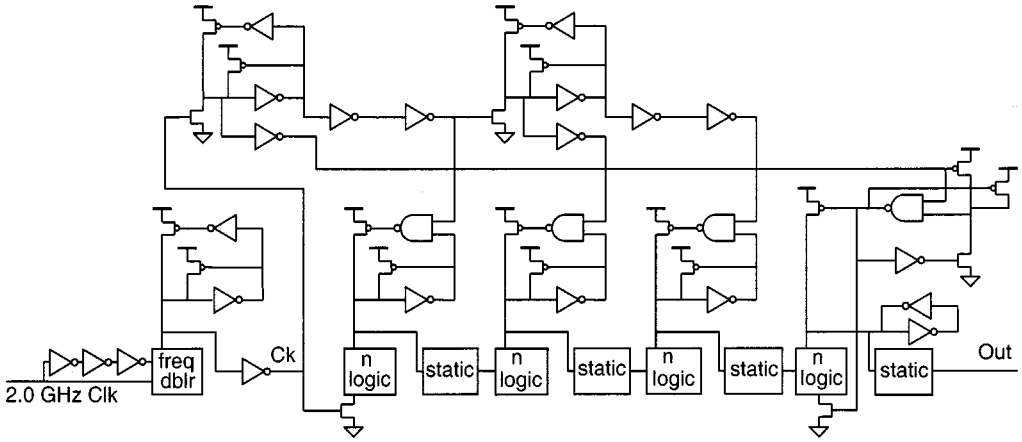

Fig. 9. ALU design in global STP: a typical bit.

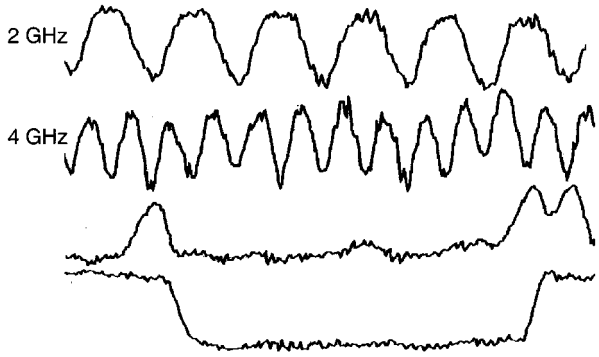

Fig. 10. Laser voltage probe of 4-GHz ALU operation.

carry blocker, and the PGK logic, requires eight logic inversion levels (Fig. 9).

The fundamental logic family used in the ALUs is a type of self-resetting domino [6] internally called Globally Reset Domino with Self-Terminating Precharge (Global STP). The 2-GHz clock that is globally distributed is doubled locally to generate a 4-GHz clock with a self-timed phase boundary. Each gate cycles at 4 GHz, i.e., evaluation, precharge, and termination of the precharge signal are all completed within 250 ps, including all appropriate timing margins, and the gate is ready to receive new data. The state is held in the second-to-last stage. In the beginning of a 4-GHz cycle, the last two stages are being reset while the first six logic stages of the ALUs (three domino and three static) go through an evaluation. After the precharge of the second-to-last stage has been turned off, the evaluate signal propagates to the output, where it is held until the next cycle. The first six stages, however, are reset in the same cycle in which they have evaluated. The precharge of these stages is globally initiated by a suitably delayed clock and is locally terminated, allowing a faster cycle time than if the precharge signal was terminated by the global reset clock. A representative bit with the global reset circuitry is shown in Fig. 9.

Fig. 10 shows laser voltage probe [7] results for two clocks and two internal ALU signals. The clocks shown are the 2-GHz main clock and the generated 4-GHz local clock. The two signals show a one-clock output pulse and a multiclock latched result.

2) Complex Integer Operations: The simple, very frequent ALU operations go to the high-speed integer ALU execution units described above. Integer operations that are more complex go to separate hardware for completion. Most integer shift or rotate operations go to the complex integer dispatch port. These shift operations have a latency of four clocks. Integer multiply and divide operations also have a long latency. Typical forms of multiply and divide have a latency of about 14 and 60 clocks, respectively.

3) Low-Latency L1 Data Cache: The L1 data cache is an 8K-byte cache that is used for both integer and floating-point/SSE loads and stores. It is organized as a four-way set-associative cache that has 64 B per cache line. It is a write-through cache, which means that writes to it are always copied into the L2 cache. It can do one load and one store per clock cycle, each of which can be up to 16 B long.

The latency of load operations is a key aspect of processor performance. This is especially true for IA-32 programs that have many loads and stores because of the limited number of registers in the instruction set. The Pentium® 4 microarchitecture optimizes for the lowest overall load-access latency with a small very low-latency 8K-byte cache backed up by a large high-bandwidth second-level cache with medium latency. For most IA-32 programs, this configuration of a small but very low-latency L1 data cache followed by a large medium-latency L2 cache gives lower net load-access latency and, therefore, higher performance than a larger, slower L1 cache. The L1 data cache operates with a two-clock load-use latency for integer loads and a six-clock load-use latency for floating-point/SSE loads.

This two-clock load latency is difficult to achieve with the high clock rates of the Pentium® 4 processor. The cache access algorithms leverage the fact that almost all accesses hit the first-level data cache and the data TLB (DTLB) to enable the two-cycle load-access latency.

At this high frequency and with this deep machine pipeline, the distance in clocks, from the load scheduler to execution, is longer than the load execution latency itself. The UOP schedulers dispatch dependent operations before the parent load has finished executing. In most cases, the scheduler assumes that the load will hit the L1 data cache. If the load misses the L1

data cache, there will be dependent operations in flight in the pipeline. These dependent operations that have left the scheduler will get temporarily incorrect data. This is a form of data speculation. Using a mechanism known as *replay*, logic tracks and re-executes instructions that use incorrect data. Only the dependent operations are replayed; the independent ones are allowed to complete.

There can be up to four outstanding load misses from the L1 data cache pending at any one time in the memory subsystem.

4) Store-to-Load Forwarding: In an out-of-order-execution processor, stores are not allowed to be committed to permanent machine state (the L1 data cache, etc.) until after the store has retired. Waiting until retirement means that all other preceding operations have completely finished. All faults, interrupts, mispredicted branches, etc., must have been signaled beforehand to make sure this store is safe to perform. With the very deep pipeline of the Pentium® 4 processor, it can take hundreds of clock cycles for a store to make it to retirement. Also, stores that are at retirement often have to wait for previous stores to complete their update of the data cache. This machine can have up to 24 stores in the pipeline at a time. Sometimes many of them have retired but have not yet committed their state into the L1 data cache. Other stores may have completed, but have not yet retired, so their results are also not yet in the L1 data cache. Often loads must use the result of one of these pending stores, especially for IA-32 programs, due to the limited number of registers available. To enable this use of pending stores, modern out-of-order-execution processors have a pending store buffer that allows loads to use the pending store results before the stores have been written into the L1 data cache. This process is called store-to-load forwarding.

To make this store-to-load-forwarding process efficient, this pending store buffer is optimized to allow efficient and quick forwarding of data to dependent loads from the pending stores [5]. The Pentium® 4 processor has a 24-entry store-forwarding buffer to match the number of stores that can be in flight at once. This forwarding is allowed if a load hits the same address as a proceeding completed pending store that is still in the store-forwarding buffer. The load must also be the same size or smaller than the pending store and have the same beginning physical address as the store for the forwarding to take place. This is by far the most common forwarding case. If the bytes requested by a load only partially overlap a pending store or need to have some bytes come simultaneously from more than one pending store, this store-to-load forwarding is not allowed. The load must get its data from the cache and cannot complete until the store has committed its state to the cache.

This disallowed store-to-load forwarding case can be quite costly, in terms of performance loss, if it happens very often. When it occurs, it tends to happen on older Pentium® III core optimized applications that have not been optimized for modern out-of-order-execution microarchitectures. The newer versions of the IA-32 compilers remove most or all of these bad store-to-load forwarding cases, but they are still found in many old legacy Pentium® III optimized applications and benchmarks. This bad store-forwarding case is a big performance issue for Pentium® III-based processors and other modern processors, but due to the even deeper pipeline of the Pentium® 4 processor, these cases are even more costly in performance to the Pentium® 4 processor.

5) FP/SSE Execution Units: The floating-point (FP) execution cluster of the Pentium® 4 processor is where the floating-point, MMX, SSE, and SSE2 instructions are executed. These instructions typically have operands from 64 to 128 bits in width. The FP/SSE register file has 128 entries and each register is 128 bits wide. This execution cluster has two 128-b execution ports that can each begin a new operation every clock cycle. One execution port is for 128-b general execution and one is for 128-b register-to-register moves and memory stores. The FP/SSE engine can also complete a full 128-b load each clock cycle.

Many FP/multimedia applications have a fairly balanced set of multiplies and adds. The machine can usually keep busy interleaving a multiply and an add every two clock cycles at much less cost than fully pipelining all the FP/SSE execution hardware. In the Pentium® 4 processor, the FP adder can execute one extended-precision (EP) addition, one double-precision (DP) addition, or two single-precision (SP) additions every clock cycle. This allows it to complete a 128-b SSE/SSE2 packed SP or DP add UOP every two clock cycles. The FP multiplier can execute either one EP multiply every two clocks, or it can execute one DP multiply or two SP multiplies every clock. This allows it to complete a 128-b IEEE SSE/SSE2 packed SP or DP multiply UOP every two clock cycles giving a peak 6 GFLOPS for single-precision or 3 GFLOPS for double-precision floating point at 1500 MHz.

Many multimedia applications interleave adds, multiplies, and pack/unpack/shuffle operations. For integer SIMD operations, which are the 64-b wide MMX or 128-b wide SSE2 instructions, there are three execution units that can run in parallel. The SIMD integer ALU execution hardware can process 64 SIMD integer bits per clock cycle. This allows the unit to do a new 128-b SSE2 packed integer add UOP every two clock cycles. A separate shuffle/unpack execution unit can also process 64 SIMD integer bits per clock cycle, allowing it to do a full 128-b shuffle/unpack UOP operation each two clock cycles. MMX/SSE2 SIMD integer multiply instructions use the FP multiply hardware mentioned above to also do a 128-b packed integer multiply UOP every two clock cycles.

The FP divider executes all divide, square root, and remainder UOPs. It is based on a double-pumped SRT radix-2 algorithm, producing two bits of quotient (or square root) every clock cycle.

The deep buffering of the Pentium® 4 processor (126 UOPs and 48 loads in flight) allows the machine to examine a large section of the program at once. The out-of-order-execution hardware often unrolls the inner execution loop of these programs numerous times in its execution window. This dynamic unrolling allows the Pentium® 4 processor to overlap the long-latency FP/SSE and memory instructions by finding many independent instructions to work on simultaneously. This deep window produces more performance for most FP/multi-media applications than additional execution units would.

FP/multimedia applications usually need a very high-bandwidth memory subsystem. Floating-point and multimedia applications frequently have large working sets and fit poorly in the L1 data cache. To optimize these applications, the Pen-

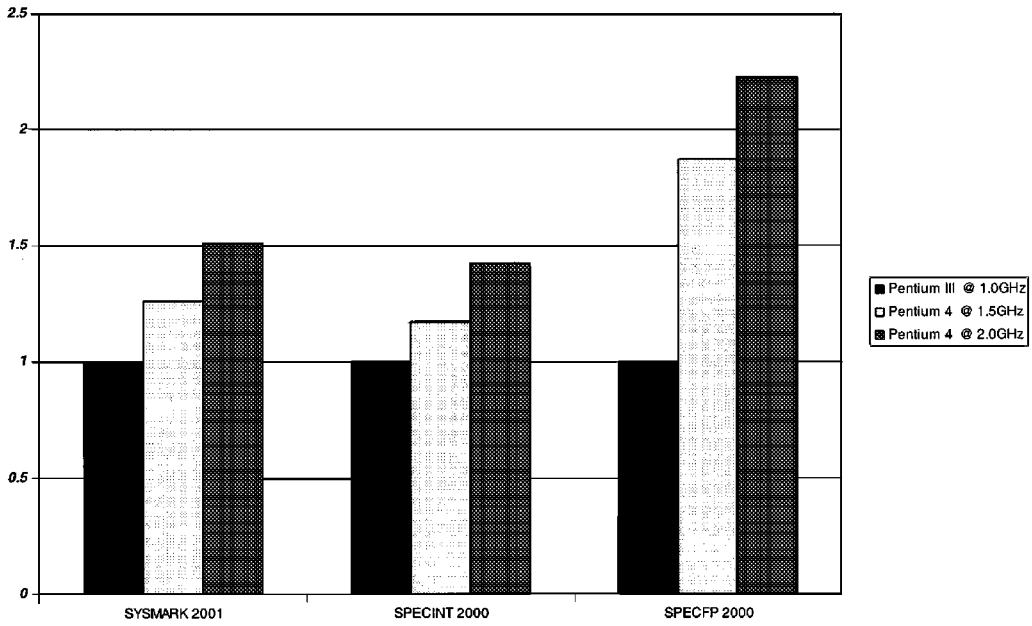

Fig. 11. Relative performance.

tium® 4 processor has a high-bandwidth path from the L2 data cache to the L1 data cache. Many FP/multimedia applications stream data from memory—no practical cache size will hold the data. They need a high bandwidth path to main memory to perform well. The long 128-byte L2 cache lines together with the hardware prefetcher described below help to prefetch the data that the application will soon need, effectively hiding the long memory latency. The high-bandwidth system bus of the Pentium® 4 processor allows this prefetching to help keep the execution engine well fed with streaming data.

D. Memory Subsystem

The Pentium® 4 processor has a highly capable memory subsystem to enable the new emerging high-bandwidth stream-oriented applications such as 3-D, video, and content creation. The memory subsystem includes the L2 cache and the system bus. The L2 cache stores data that cannot fit in the L1 caches. The external system bus is used to access main memory when the L2 cache has a cache miss and also to access the system I/O devices.

1) Level 2 Instruction and Data Cache: The L2 cache is a 256K-byte cache that holds both instructions that miss the trace cache and data that miss the L1 data cache. The L2 cache is organized as an eight-way set-associative cache with 128 B per cache line. These 128-byte cache lines consist of two 64-byte sectors. A miss in the L2 cache typically initiates two 64-byte access requests to the system bus to fill both halves of the cache line. The L2 cache is a writeback cache that allocates new cache lines on load or store misses. It has a net load-use access latency of seven clock cycles. A new cache operation can begin every two processor clock cycles for a peak bandwidth of 48 GB/s, when running at 1500 MHz.

Associated with the L2 cache is a hardware prefetcher that monitors data access patterns and brings data automatically into the L2 cache. It attempts to stay 256 B ahead of the current data access locations. This prefetcher remembers the history of cache misses to detect concurrent independent streams of data that it tries to bring on-chip ahead of program use. The prefetcher also tries to minimize fetching unwanted data that can cause overutilization of the memory system and delay demand accesses the program needs.

2) 400 MHz System Bus: The Pentium® 4 processor has a system bus with 3.2 GB/s of bandwidth. This high bandwidth is a key enabler for applications that stream data from memory. This bandwidth is achieved with a 64-b wide bus capable of transferring data at a rate of 400 MHz. It uses a source-synchronous protocol that quadpumps the 100-MHz bus to give 400 million data transfers per second. It has a split-transaction deeply pipelined protocol to allow the memory subsystem to overlap many simultaneous requests to actually deliver high memory bandwidths in a real system. The bus protocol has a 64-byte access length.

IV. Performance

At the time of introduction (November 2000), the Pentium® 4 processor delivered the highest SPECint_base performance of any processor in the world with a score of 524 at 1500 MHz. It also delivers good SPECfp2000 performance, with a score of 549 at 1500 MHz. Fig. 11 shows the relative performance of the Pentium® 4 processor at 1500 and 2000 MHz, compared to a 1000-MHz Pentium® III processor. Three different industry standard benchmarks are shown, Sysmark 2001, SPECint2000, and SPECfp2000. The higher frequency of the Pentium® 4 processor gives higher performance on integer workloads like SPECint2000, and the 3.2-GB/s system bus and hardware prefetcher give significant performance improvement on SPECfp2000.

V. Conclusion

The Pentium® 4 processor is a new processor microarchitecture and design of the IA-32 processor architecture. It is the be-

ginning of a new family of processors that utilize the new Intel Pentium® 4 microarchitecture. Its deeply pipelined design delivers high frequencies and performance. It uses many novel microarchitectural ideas including a trace cache, double-clocked ALU, new low-latency L1 data cache algorithms, and a new high-bandwidth system bus.

ACKNOWLEDGMENT

The authors thank all the architects, designers, and validators who contributed to making this processor into a real design.

REFERENCES

[1] S. Yang *et al.*, "A high-performance 180-nm generation logic technology," presented at the IEDM, 1998.
[2] N. Kurd *et al.*, "Multi-GHz clocking scheme for Intel® Pentium® 4 microprocessor," in *Int. Solid State Circuits Conf.*, Feb. 2001, pp. 404–405.
[3] D. Sager *et al.*, "A 0.18-μm CMOS IA-32 microprocessor with a 4-GHz integer execution unit," in *Int. Solid State Circuits Conf.*, Feb. 2001, pp. 324–325.
[4] *IA-32 Intel Architecture Software Developer's Manual Vol. 1: Basic Architecture*.
[5] *Intel® Pentium® 4 Processor Optimization Intel Ref. Manual*.
[6] R. Haring *et al.*, "Self-resetting logic register and incrementer," in *Symp. VLSI Circuits*, June 1996, pp. 18–19.
[7] S. Rusu *et al.*, "Backside infrared probing for static voltage drop and dynamic timing measurements," in *Int. Solid State Circuits Conf.*, Feb. 2001, pp. 276–277.

Glenn Hinton received the M.S. degree in electrical engineering from Brigham Young University, Provo, UT, in 1983.

He is an Intel Fellow and Director of IA-32 Microarchitecture Develoment in the Intel Architecture Group. He joined Intel Corporation in 1983. He was one of the three Senior Architects in 1990 responsible for the P6 processor microarchitecture, which became the Pentium® Pro, Pentium® II, Pentium® III, and Celeron™ processors. He was responsible for the microarchitecture development of the Pentium® 4 processor.

Michael Upton received the B.S. and M.S. degrees in electrical engineering from the University of Washington, Seattle, in 1985 and 1990, respectively, and the Ph.D. degree from the University of Michigan, Ann Arbor, in 1994.

He is a Principal Engineering/Architect in Intel's Desktop Platforms Group, and is one of the architects of the Intel® Pentium® 4 processor. After a number of years in IC design and CAD tool development, he entered the University of Michigan to study computer architecture. Upon completion of the Ph.D. degree, he joined Intel Corporation to work on the Pentium® Pro and Pentium® 4 processors.

David J. Sager (M'74) received the Ph.D. degree in physics from Princeton University, Princeton, NJ, in 1973.

He is a Principal Engineer/Architect in Intel's Desktop Plateforms Group, and is one of the overall architects of the Intel® Pentium® 4 processor. He joined Intel Corporation, Portland, OR, in 1995. He was previously with Digital Equipment Corporation for 17 years in their processor research labs.

Darrell Boggs received the M.S. degree in electrical engineering from Brigham Young University, Provo, UT.

He is a Principal Engineer/Architect with Intel Corporation, and has been working as a microarchitect for nearly ten years. He played a key role on the Pentium® Pro Processor design, and was one of the key architects of the Pentium® 4 Processor. Darrell holds many patents in the areas of register renaming, instruction decoding, events and state recovery mechanisms.

Douglas M. Carmean is a Principal Engineer/Architect with Intel's Desktop Products Group, Hillsboro, OR. He was one of the key architects, responsible for definition of the Intel Pentium® 4 processor. He has been with Intel for twelve years, working on IA-32 processors from the 80486 to the Intel Pentium® 4 processor and beyond. Prior to joining Intel, he worked at ROSS Technology, Sun Microsystems, Cypress Semiconductor, and Lattice Semiconductor.

Patrice Roussel received the M.S. degree in signal processing and VLSI design from the University of Rennes in 1980 and L'Ecole Superieure d'Electricite in 1982, respectively.

Upon graduation, he worked at Cimatel, an Intel/Matra Harris joint design center. He joined Intel Corporation in Arizona in 1988, where he worked on the 960CA chip. In late 1991, he moved to Intel, Hillsboro, OR, to work on the P6 processors. Since 1995, he has been the floating-point architect of the Pentium® 4 processor.

Terry I. Chappell (A'91–SM'95) received the B.S. degree in physics from Harvey Mudd College and the M.S. and Ph.D. degrees in electrical engineering from the University of California, Berkeley.

From 1978 to 1995, he worked at the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, as a Research Staff Member involved in the development of high-speed CMOS circuits. In 1995, he joined Intel Corporation, Portland, OR, to work on the design of Pentium® 4 processor circuits and tools. He holds 30 patents and is the author of many technical papers and presentations.

Thomas D. Fletcher received the B.S. degree in physics and the M.S. degree in electrical engineering from Brigham Young University, Provo, UT.

He has worked at Intel Corporation, Hillsboro, OR, for ten years and is currently a Principal Engineer. He was the clock unit owner for the Pentium® Pro processor and was responsible for the Pentium®4 processor circuit methodology and clocking during the early development. He is currently doing research in advanced circuit technology for future microprocessors. Before joining Intel, he worked at Signetics and Philips for twelve years, where he designed bipolar, CMOS, and BiCMOS digital logic and CMOS analog circuits.

Mark S. Milshtein (M'98) received the B.A. degree in physics from Rutgers University, New Brunswick, NJ, in 1986.

He was with Bellcore from 1986 to 1989, working on GaAs heterostructures. He was with IBM Thomas J. Watson Research Center, Yorktown Heights, NY, from 1989 to 1995 working on GaAs devices and materials, copper metallization, and high-speed microprocessor circuit design. In 1995, he joined Intel Corporation, Portland, OR, to work on the design of high-speed circuits for the Pentium® 4 processor. He holds eight patents and is a co-author of numerous technical papers.

Milo Sprague received the B.S.E.E. degree in 1990 and the M.S.E.E. degree in 1993, both from Virginia Polytechnic Institute and State University, Blacksburg.

He joined Intel Corporation, Hillsboro, OR, in 1993 to help design the Pentium® Pro microprocessor. While at Intel, he has researched high-speed pulsed circuit techniques and incorporated them in elements of the Pentium® 4 microprocessor.

Samie Samaan received the Diploma of Advanced Studies (D.E.A.) in electrical engineering from the University of Nancy 1, Nancy, France, in 1982, and the M.S.E.E. degree in electromagnetics and microwaves from the University of Mississippi, Oxford, in 1984.

From 1984 to 1992, he was a Design Engineer at Tektronix, in the CRT division, and later at Tektronix Research Labs designing gigahertz analog linear ICs, in silicon and GaAs. He joined Intel Corporation, Hillsboro, OR, in 1994, where he worked on the Pentium® II front-side bus electrical specification, and timing test issues. Since 1996, he has been a VLSI Senior Staff Design Engineer. He designed the pulsed-domino 4-GHz ALUs in the Pentium® 4 processor, in addition to working on intradie variation and circuit race modeling. He is currently in the Circuit Design Group at Intel, developing future products.

Robert Murray was Integer Execution Unit owner on the Pentium® 4 processor. He is currently working on future IA-32 processors.