# MMX™ Microarchitecture of Pentium® Processors With MMX Technology and Pentium® II Microprocessors

Michael Kagan, IDC, Intel Corp.
Simcha Gochman, IDC, Intel Corp.
Doron Orenstien, IDC, Intel Corp.
Derrick Lin, MD6, Intel Corp.

Index Words: MMX™ technology, multimedia applications, IA extensions, Pentium® processor

**Abstract**

*The MMX™ technology is an extension to the Intel Architecture (IA) aimed at boosting the performance of multimedia applications. This technology is the most significant IA extension since the introduction of the Intel386™ microprocessor. The challenge in implementing this technology came from retrofitting the new functionality into existing Pentium® and Pentium® Pro processor designs.*

*The main challenge was how to incorporate the new instructions while also keeping upcoming products on the Intel performance curve. Both projects had to deliver higher performance than their predecessors on legacy applications, using both frequency gain and CPI (Clocks Per Instruction) microarchitecture improvements.*

*On the other hand, new instructions had to be implemented in a cost-effective way, e.g., provide a breakthrough performance boost on multimedia applications while maintaining reasonably low die size cost. Moreover, the Pentium processor with MMX technology and Pentium® II processor, being the first microprocessors to implement the new Instruction Set Architecture (ISA), had to deliver superior multimedia performance to demonstrate that the benefit of the ISA extension would be compelling enough for Independent Software Vendors (ISVs) to develop software using these new instructions and fuel up the software spiral.*

*The new instructions operate on packed data types (single operand represents more than one datum) and use a flat register file that is architecturally aliased to an existing register file of the Floating-Point (FP) stack. This definition allows a variety of implementation alternatives.*

*Additional changes were introduced in the micro-architecture of the predecessor microprocessor in order to stay on the performance curve, improving the frequency and clock per instruction performance.*

## Introduction

During the ramp of the Pentium® processor in 1993, it became evident that the home market was becoming a major consumer of PCs, with a major boost coming from multimedia applications.

Traditionally, multimedia applications were supported by expansion hardware with dedicated software, thereby increasing the cost of the machine and lacking common standards. Engineers in the Intel Architecture group envisioned the need of executing operations for multimedia on the core CPU. This would establish a standard for the industry, reduce the cost of the system, and free up motherboard expansion slots.

A distinct characteristic of multimedia applications is the execution of the same operation on multiple small-size data items (e.g., 8 and 16 bits). The Single Instruction Multiple Data (SIMD) architecture provides a cost-effective solution for such applications, and therefore it was decided to extend the IA with 57 new MMX™ SIMD-type instructions.

At the time of this decision, two design projects were in their initial development stages: a high-end Pentium processor, and the Pentium® II processor, a Pentium® Pro processor compaction, both based on Intel's 0.35u CMOS process. In order to allow a fast ramp and a top-to-bottom penetration of the new extensions into the PC market, it was decided to incorporate new instructions in both projects and have them become the flagships of the new architecture extension.

At that time, the Pentium and Pentium Pro processors were both in advanced development stages with a much more mature database and silicon experience. In order to stay on the performance curve and catch up on frequency, we had to set a more aggressive frequency goal than our predecessors and also improve CPI performance. In the Pentium processor with MMX technology, this resulted in restructuring the entire machine by adding one more stage to the processor main pipeline. The Pentium II processor design team improved the performance of graphics applications and achieved a higher frequency through less aggressive architectural changes.

Both design teams delivered excellent results. The Pentium processor with MMX technology achieved both its CPI and frequency goals. It is 20% higher in frequency (running at 233MHz in production) and 15% faster on CPI than other Pentium processors. The Pentium II processor significantly improved the performance of graphics code and achieved a 300MHz frequency at introduction. The speedup goal for multimedia applications was achieved as well. Most applications using the new instructions improved by a factor of 1.6X, with some having improved up to 4X.

## Pentium Processor With MMX Technology Microarchitecture

In order to exceed the performance of its predecessor, the design team had to improve both the frequency and CPI performance of the microprocessor. Both of these goals could be achieved with microarchitecture changes implemented in the new processor.

### Frequency Speedup

Frequency is the most significant factor that determines the performance of a microprocessor and is a major (and sometimes only) performance indicator used by customers. Therefore, it was not possible to come up with a new product running at a lower frequency than its predecessor.

The frequency improvement of a product approaches asymptotically the architectural limit of the device by cleaning up escapes and by making slight design improvements in critical paths. Therefore, in order to match a predecessor's frequency, a product that comes to market later must have higher architectural frequency limits. Figure 1 illustrates frequency improvement trends for the Pentium processor and Pentium processor with MMX technology.

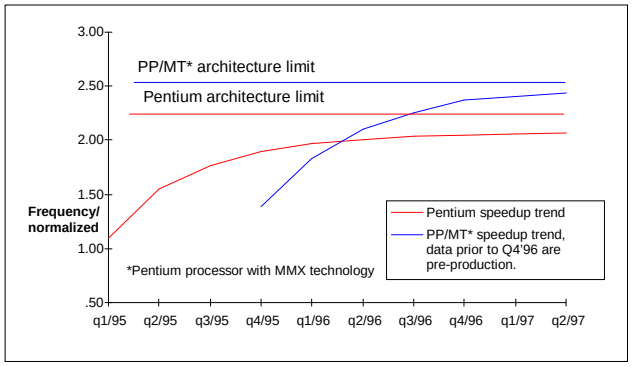


Figure 1. Frequency Improvement Trends

In order to improve the architectural limit of the device, we had to identify and resolve the major speed bottlenecks of the Pentium processor's architecture. After a thorough analysis, two major bottlenecks were identified: the decoder and the data cache access. The two bottlenecks were dependent. In other words, resolving one of them would help to speed up the other one. We decided to resolve the decoder bottleneck, since it was simpler and less risky, and it would also allow a smooth implementation of MMX instruction decoding. The Pentium processor execution pipeline originally consisted of five pipeline stages: Pre-fetch (PF), Decode1 (D1), Decode2 (D2), Execute (E), and Writeback (WB). We added an additional pipeline stage in the front end of the machine, rebalanced the entire pipeline to take advantage of the extra clock cycle, and added a queue between the F and D1 stages to decouple freezes, which are the most critical signals generated in every pipeline stage. Figure 2 illustrates the difference between the original Pentium processor pipeline and the MMX technology pipeline.

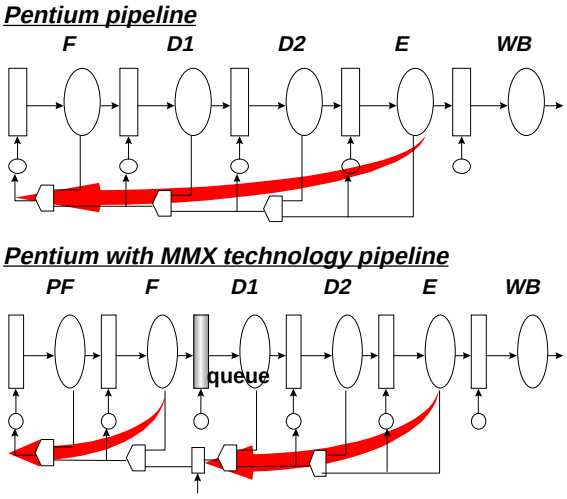


Figure 2. Pentium Processor and
Pentium Processor With MMX Technology Pipeline

An additional clock cycle in the front end of the pipeline resolved the decoder speed bottleneck and reduced fan-

out for the data cache freeze (generated in the E stage), which in turn relaxed a requirement for this freeze signal. This was the first step in the resolution of the data cache bottleneck.

The next step was to improve the timing of the data freeze signal generated by the data cache. The cache access path starts with address generation in the D2 stage, followed by a subsequent cache access in the E stage. The entire path was redesigned to self-time pipelined execution with time borrowing between the stages. The address generation logic was changed, incorporating simplified and faster adders, thereby allowing faster address generation.

The third step was the cache circuit architecture. It was performance-crucial to execute a single clock read and write operation in each cache port. As a result, the Pentium processor's cache access windows were designed to support two access windows per clock, as illustrated in Figure 3.

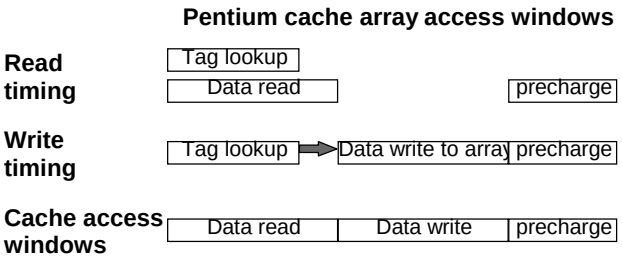


Figure 3. Pentium Processor's Cache Array Access Windows

Although read and write operations to the same port were never performed in the same cycle, cache timers had to support two access windows, thereby limiting the overall cache access time. On the other hand, since read and write operations never happen in the same clock to the same port, both access windows could never be active in the same clock cycle. In other words, during a read operation, no data access could be performed in a write access window and vice versa. Therefore, we decided to have just one data access window in the front end of the cycle (e.g., read window timing) and use it for both read and write accesses. The read access works as in other Pentium processors; it is a speculative operation and can be thrown away. Write access depends on the result of a tag lookup and cannot be executed if the same clock tags are looked up. Therefore, the Pentium processor with MMX technology implemented a cache store hit buffer. If a store hit is encountered at the cache lookup phase, the data is stored to this buffer. The actual store to the data array will be done at the data access window of the next write operation, while this window is idle. Meanwhile, before the next write, the data can be delivered from the store hit buffer to subsequent reads from this address. In other words, the Pentium processor with MMX technology pipelines write operations among each other. Each time a store is executed, the tag lookup is performed for the current store, while the data array is updated with data from the previous store. This way we could have only one data array access window, which allowed a significant speedup of cache access.

Figure 4 illustrates the Pentium processor with MMX technology's cache access windows architecture.

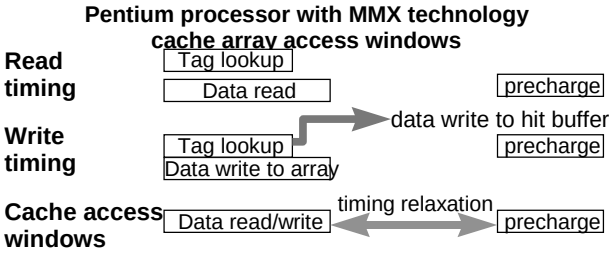


Figure 4. Pentium Processor With MMX Technology's Cache Array Access Windows Architecture

The solutions described above resolved major Pentium processor speed paths, allowing a frequency leap. Additional local changes were performed in every functional block to keep all the rest of the circuitry in line with this new goal.

In summary, the Pentium processor with MMX technology designers addressed two major bottlenecks at a global architecture level (adding a pipeline stage and re-balancing the entire machine), made few changes on the intermediate level (time borrowing between pipe stages for a specific operation), and implemented numerous local changes to keep the machine balanced. This top-down approach allowed us to achieve a 20% frequency boost over the original Pentium processor design.

## CPI Performance

Although adding a pipeline stage improves frequency, it decreases CPI performance, i.e., the longer the pipeline, the more work done speculatively by the machine and therefore more work is being thrown away in the case of branch miss prediction. The additional pipeline stage costs decreased the CPI performance of the processor by 5-6%.

In order to stay on the performance curve, we had to gain back this loss and, in addition, speed up the machine further.

The Pentium processor with MMX technology's CPI performance was increased in three major ways:

1. Improved branch prediction. We implemented a more advanced branch prediction algorithm that was developed by the Pentium Pro processor design team. This algorithm improved the prediction of branches, which resulted in fewer miss-predictions of branches

and caused less work to be thrown away. On top of the Branch Target Buffer (BTB), we also implemented a Return Stack Buffer (RSB)—a dedicated branch prediction logic for call/return instructions. The combination of the updated BTB algorithm and the RSB improved CPI performance by about 8%. This helped close the performance gap opened while adding the new pipeline stage and gave us some advantage over the Pentium processor.

2. Improving core/bus protocols. The original Pentium processor design was tuned to a 1:1 ratio between the core and bus clocks. As a result, some price/performance tradeoffs that were made for a 1:1 clock ratio were not optimal for use when the gap between the core and bus frequency increased. Several enhancements were made by the design team to tune the protocols. Write buffers were combined into a single pool, thereby allowing both pipes to share the same hardware, the clock crossover mechanism was changed, and the DP protocol was completely redesigned to decouple core and bus frequencies. These improvements gained about a 5% CPI performance improvement and simplified the design and testing (e.g., crossover, DP protocols).

3. Creating larger caches and fully-associative Translation Lookaside Buffers (TLB). In general, increasing cache size is the most cost-effective way to improve performance. The Pentium processor with MMX technology increased the size of both caches from 8Kbyte to 16Kbyte and made them four-way set-associative. Fully-associative TLBs improved CPI to some extent, making address translation faster than in the original TLB design. Larger caches and fully-associative TLBs bought us about a 7-10% CPI performance improvement.

In summary, by improving the BTB, redesigning the core/bus protocol, and making larger caches, the Pentium processor with MMX technology achieved about a 15% higher CPI performance than the Pentium processor despite the CPI loss due to the additional pipeline stage.

## MMX Technology Implementation

After setting the stage for frequency and CPI performance, we could incorporate the MMX instructions relatively straightforwardly.

The instruction decode logic had to be modified to decode, schedule, and issue the new instructions at a rate of up to two instructions per clock. The MMX opcodes are mapped to a 0F prefix, which is rarely used in previous IA native software. Therefore, decoding of these instructions in the original Pentium processor design was slow, with a throughput of two clock cycles per instruction. The Pentium processor with MMX technology decoder was redesigned to quadruple the throughput of 0F instructions, allowing two instructions per cycle throughput.

Additional modifications were made to the MMX technology pipeline to incorporate the MMX execute stage (MEX) and the MMX writeback stage. To improve the performance of MMX ARITH-MEM instructions, the integer-execute stage is used as an MMX "read-stage," where the source operands as well as the memory operands are read. As a result, an ARITH-MEM instruction is executed in a single clock cycle. Since the Pentium processor with MMX technology may pair an ARITH-MEM instruction with an ARITH instruction, it is equivalent to having three execution units (two ARITH, one LOAD) working in parallel, similar in concept to a Pentium II processor.

According to the MMX technology architecture definition, the MMX register file is aliased to the FP mantissa register file. It was decided to design dedicated hardware to execute the MMX instructions (the Munit). This unit has a dedicated MMX register file, capable of delivering four 64-bit operands and storing three 64-bit results in a single clock cycle. The Munit also incorporates the MMX execution units, which were defined and designed as a module, and which allowed the design to be shared with the Pentium II processor.

Clean partitioning of the MMX technology design and an additional pipeline stage in the decoder resulted in no speed issues associated with the new units. The area penalty for the Munit was small.

## Pentium Processor With MMX Technology Block Diagram

The block diagram of the Pentium processor with MMX technology is shown in Figure 5, outlining parts that were redesigned for speed, CPI, and MMX technology.

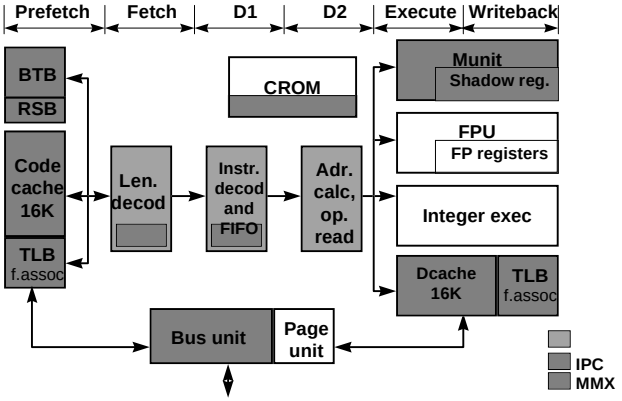

Figure 5. Block Diagram of the Pentium Processor With MMX Technology

## Results

The Pentium processor with MMX technology design achieved its goals. The processor taped out in late 1995, and samples were delivered to customers less than a week after the first silicon. With six months of extensive silicon debug, we closed the frequency gap with the Pentium processor and, half a year later, achieved 233MHz in production, which is one bin above the Pentium processor's production frequency.

Figure 6 shows the actual speed improvement of the Pentium processor and the Pentium processor with MMX technology versus the anticipated trend.

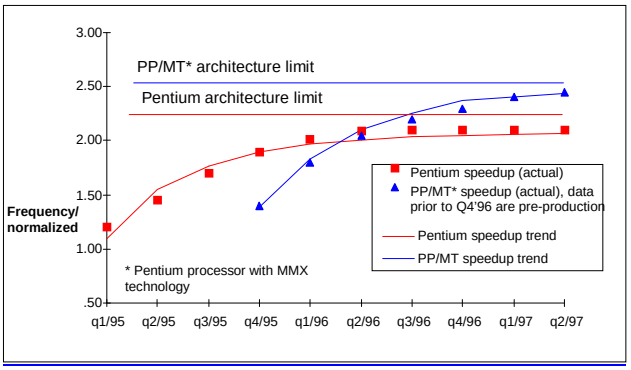


Figure 6. Actual Versus Anticipated Speed Improvement Trend

The Pentium processor with MMX technology also met its CPI goals. Figure 7 shows the CPI performance of the Pentium processor with MMX technology compared to the Pentium processor.

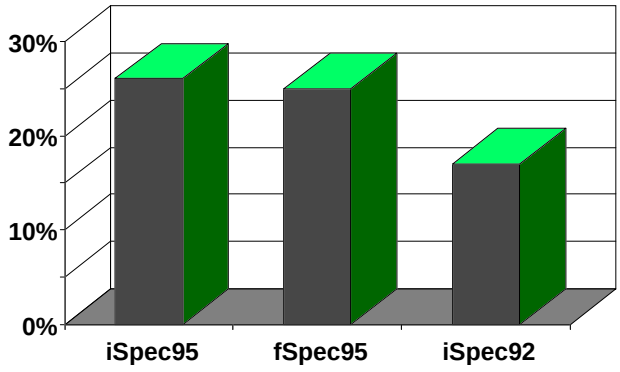


Figure 7. CPI Performance of the Pentium Processor with MMX Technology Compared to the Pentium Processor

And at last, multimedia applications gained significant performance using new instructions. Figure 8 illustrates the performance gain that can be achieved by several applications when using the new instructions.

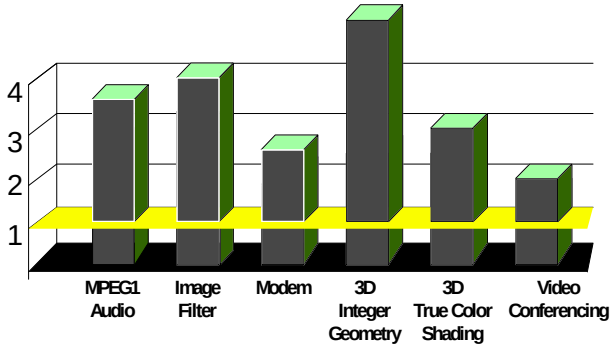


Figure 8. Performance Improvement Using New Instructions

## Pentium II Processor Microarchitecture

While the Pentium processor with MMX technology made microarchitecture changes to improve frequency and performance as well as implement the MMX technology, the Pentium II processor improved upon the Pentium Pro processor's microarchitecture and brought MMX technology to a new level of performance. The Pentium II processor is based on the dynamic execution microarchitecture of the Pentium Pro processor. Changes were made in the Pentium II processor's microarchitecture to improve graphics performance and to implement MMX technology. In addition, the entire back-side bus interface that connects the processor to an off-chip second-level cache was redesigned to allow low-cost commodity SRAMs to be used as second-level cache. Doing so significantly reduced the system cost compared to the Pentium Pro processor's Multi-Chip Module (MCM) that houses the processor as well as the second-level cache. A higher frequency was achieved through aggressive circuit techniques and other changes.

### Overview

The Pentium II processor is the second Intel microprocessor to implement MMX technology. The Pentium II processor's MMX technology implementation offers multimedia applications the benefits of an out-of-order execution, aggressive memory speculation, a superpipelined and superscalar microarchitecture, etc. These are the same features that the Pentium Pro microprocessor provides. The Pentium II processor supports two packed ALU operations, one packed shift, and one packed multiply operation. Pack and unpack operations are implemented by the packed shifter. The Pentium II processor allows packed shift and packed multiply to be executed concurrently.

## MMX Technology Implementation

The Pentium II processor's microarchitecture is similar to that of the Pentium Pro microprocessor. The Pentium Pro processor's high-level microarchitecture consists of the following pipelines: instruction pre-fetch, length decode, instruction decode, rename/resource allocation, uop scheduling/dispatch, execution, writeback, and retirement. The length decoder was modified to include decoding for the new instructions. Some of the Pentium Pro processor's microarchitecture was modified to add MMX technology. The Instruction Decoder logic was modified to convert the new MMX instructions to Pentium Pro processor-specific uops (new Single Instruction Multiple Data [SIMD] uops were added to implement the new functionality). The renamer (RAT) was modified to handle MMX technology management of the floating-point stack because MMX registers are aliased onto the floating-point stack to avoid the need to modify operating systems. The resource allocator (ALLOC) was changed to provide static scheduling binding for the new SIMD uops. A new execution unit, the SIMD unit (MMX instructions are primarily SIMD instructions), was added. The Reservation Station (RS) was changed to accommodate the new SIMD 64-bit datapath. In addition, minor changes were made to expand some buses to accommodate the 64-bit MMX technology requirement.

Performance data showed a need for a dual execution pipeline for MMX instructions. Of the five execution ports (port 0 - FP, integer; port 1 - integer; port 2 - load; port 3 - store address; and port 4 - store data), it was decided to put the SIMD unit in ports 0 and 1. Due to area constraints, the SIMD unit implemented only one shifter and one multiplier. The multiplier and shifter are on different ports so that the two operations can execute in parallel. Arithmetic and logical execution hardware was duplicated to provide concurrent execution. The three-clock latency, fully pipelined multiplier was placed in port 0 because the Reservation Station (RS) already had logic to handle a multiple clock latency in port 0.

Floating-point registers are stacked. Although multimedia registers are aliased onto the floating-point stack, they are not used as stack registers. Like integer registers, multimedia registers are general purpose, randomly-accessed registers. The RAT treats the stack-based floating-point registers and the randomly-accessed multimedia registers the same way in terms of register renaming. This simplifies the floating-point rename logic. However, this approach only works if the floating-point Top-of-Stack (TOS) is zero. All floating-point registers are converted from stack-based logical registers to real logical registers by adding the stack register number with the floating-point TOS. If the floating-point TOS is zero, the stack adjustment is transparent to multimedia registers. For all floating-point and multimedia registers, the RAT converts the stack-based register number into a logical register number by factoring in the TOS reference. The logical register number is used to read-write the floating-point rename tables. In addition, each uop that presents stack-referenced registers as source/destination can modify the TOS. The RAT needs to perform the TOS reference and manipulation logic on the fly, as uops enter the rename pipeline stage.

Since the existing floating-point register logic works transparently for multimedia registers only if the floating-point TOS is zero prior to an MMX instruction, the rename logic will produce errors if the floating-point TOS is not zero. A microcode assist was created to correct the problem and redo the operation. An assist is a customer-invisible event that flushes out the machine and allows microcode to handle rare but difficult-to-handle problems. Since all MMX instructions zero the TOS, the assist needs to write the TOS to zero and restart the operation. Then, the operation sees zero TOS and the rename can occur correctly.

The MMX technology specification requires that all MMX instructions, except EMMS, set all floating-point stack registers to full. This logic is performed by the RAT as well. The setting of the stack valid bits to full is accomplished by setting all stack valid bits at once. This logic can get complicated when an MMX instruction is issued or retired at the same time as a floating-point instruction that performs a stack pop or otherwise clears one or more stack valid bits. To avoid the complexity, MMX instructions and floating-point instructions that check for stack valid or perform a pop are not allowed to issue or retire in the same clock. Due to timing constraints, the function of disallowing MMX instructions and floating-point instructions that check for stack valid or perform a pop to retire in the same clock is performed by another microcode assist. This assist can be avoided by following the MMX technology programming guidelines, i.e., placing an EMMS instruction between the last MMX instruction and the first floating-point instruction.

Challenges were encountered in the Instruction Decoder (ID) when adding functionality to decode the MMX instructions. All MMX instructions are placed on the 0F opcode map. However, these instructions are not arranged in blocks or in regular patterns. Opcode holes are sprinkled among the new instructions. The ID has three decoders: decoder 0, 1, and 2. Decoder 0 is a full-function decoder capable of decoding all instructions, with a maximum output of four uops. Decoders 1 and 2 are capable of decoding a subset of instructions that require only one uop. If the ID encounters an illegal instruction, a pair of uops are inserted into the uop stream by decoder 0,

so the Re-order Buffer (ROB) knows to signal an illegal opcode fault. Thus, decoders 1 and 2 were not capable of handling illegal opcodes. There is a PLA (Programmable Logic Array) that determines whether an instruction can be decoded by any decoder. If not, the instruction must be decoded by decoder 0. This was a very speed-critical PLA and a new way of thinking was needed.

Since decoders 1 and 2 can decode one uop instruction, all register-register forms of MMX instructions, as well as the MOVD and MOVQ load instructions, can be decoded by these decoders. To reduce the minterm count of the critical PLA, we considered reducing the number of MMX instructions decodable by all decoders. This led to a noticeable performance loss. In order to avoid this performance loss while still meeting frequency goals, a new assist was created so that decoders 1 and 2 could handle illegal opcodes. This way, opcode holes and single uop instructions look the same to the two decoders. They all cause one uop to be generated. Illegal opcodes that are instruction holes in the MMX instruction opcode map are defined to generate a one uop assist call. This assist call instructs the ROB to flush the machine and causes an assist microcode flow to cause the processor to handle illegal opcode faults. All legal MMX instructions that decoders 1 and 2 can handle generate signal uops that implement the instructions. This optimization results in a minterm count increase of only 3 (out of 30). This enabled the PLA to meet the Pentium II processor's frequency target.

All MMX operations except load and store are executed by the SIMD unit. Each SIMD adder is capable of performing add, subtract, and compare of 8-byte, 4-word, and 2-doubleword data types. The adders are optimized to perform these operations with roughly the same speed as a normal 32-bit adder. The SIMD shifter can perform left and right parallel shifts of word, doubleword, and quadword. The SIMD multiplier performs the parallel multiply and multiply-add operations.

### Summary

The challenges of implementing MMX technology in the Pentium II processor were in adding a major functionality while creating minimal disturbance to the base microarchitecture, which the processor inherited from the Pentium Pro processor. Furthermore, the changes made to the microarchitecture were such that the processor could run at the same frequency whether or not the functionality was present. To meet these objectives, novel microarchitecture, logic, and circuit techniques had to be used. The result of the Pentium II processor MMX technology implementation was a processor that can run at 300MHz at introduction, surpassing Pentium Pro processor performance.

## Conclusion

The Pentium processor with MMX technology and Pentium II microprocessors are now available in the marketplace, introducing the next step in desktop, workstation, and server computing. The Pentium processor with MMX technology and Pentium II processors' acceptance into the marketplace is beyond expectations, and their ramp-up is the fastest in Intel history.

Both design projects successfully implemented the IA extension with a small cost in silicon area and with no architecture or logic bugs.

## Acknowledgment

This paper presents the hard work of many talented people from the Pentium processor with MMX technology and Pentium II processor architecture teams. The persistence and dedication of these design and architecture teams made it possible to drive the definition and execution of these two outstanding design projects. Special thanks to design engineers from the Arizona design center who provided tremendous help to the Pentium processor with MMX technology design by sending about 40 engineers to work with us in Israel and taking some Pentium processor with MMX technology design tasks to execute in the Arizona design center.

## Authors

Michael Kagan joined Intel in 1983 after his IDF military service. Michael graduated from the Technion in 1981 with a BS in Computer and Electrical Engineering. Michael worked in various areas of VLSI design, on all Intel microprocessor architectures, i.e., the 80890 (design engineer), 80387 (design engineer), 80860 (architecture and logic design manager), and Pentium processor with MMX technology (design manager). Michael's main professional focus is on processors and platform architecture and validation. His email address is michaelk@iil.intel.com.

Simcha Gochman received his BS and MS degrees in Electrical Engineering from the Technion in Haifa, Israel. From 1980 to 1983, he was an adjunct faculty member in the Department of Electrical Engineering at the Technion. From 1983 to 1984, he was with Refa'el, the Israeli Weapons Development Authority. Since 1984, he has been with the Intel Design Center in Haifa working on VLSI design and processor microarchitectures. His email address is simcha@iil.intel.com.

Derrick Lin is currently one of the senior logic designers on the Pentium II processor project. He received his BS

and MS degrees in Electrical Engineering from Stanford University and has been with Intel since 1991. His technical interests include all aspects of microprocessor and computer design, for example, microarchitecture, compiler, and high-speed circuit techniques. His email address is dlin@mipos2.intel.com.