
Multimedia Instruction Set Extensions for a Sixth-Generation x86 Processor

Cyrix Corporation
<http://www.cyrix.com>

Multimedia Architectural Goals

- ❑ Develop a host-based multimedia capability at minimal cost
 - ◆ Provide a substantial increase in performance for multimedia applications
 - ◆ Native to the processor
- ❑ Programmable solution
 - ◆ Provide software flexibility to meet the needs of today's & tomorrow's multimedia requirements
- ❑ MMX compatible == > Avoid x86 software fragmentation
 - ◆ Complete compatibility with MMX instruction set

M2 Processor Specification

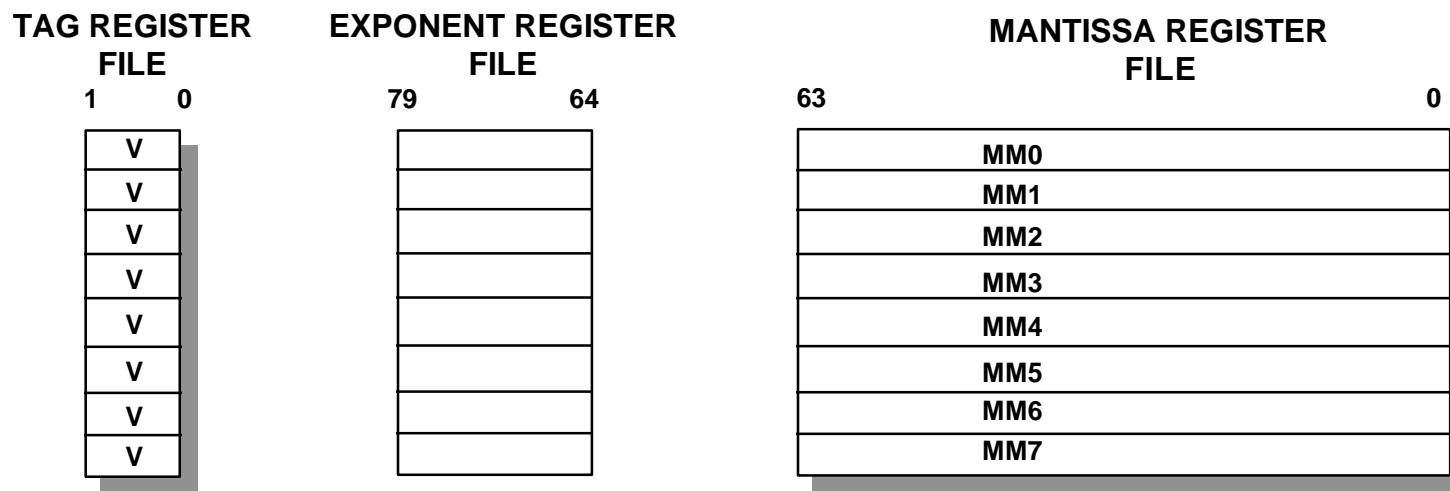
- ❑ Multimedia: MMX compatible
- ❑ Transistor count: 6 million
- ❑ 64 KB L1 cache
- ❑ 1600 MOPS
- ❑ 2.5V core; 3.3V bus interface
- ❑ 6x86 socket compatible

Multimedia Extensions: Overview

- ❑ New data types: packed Byte/Word/Doubleword, Quadword
- ❑ Eight media registers
 - ◆ Aliased with floating point (FP) register state
 - ◆ 64-bit general purpose registers
- ❑ Instruction set
 - ◆ Single Instruction Multiple Data (SIMD)
 - ◆ Saturation or modulo arithmetic
- ❑ Instructions
 - ◆ Arithmetic, Comparison, Conversion, Logical, Shift, & Data Transfer
 - ◆ Source operand can reside in memory or in media register
 - ◆ Destination operand can only reside in media register

Software Compatibility

- ❑ No new processor state is required to support MMX
- ❑ MMX instructions use existing FP context switch mechanism
- ❑ Compatible with existing OS & application software

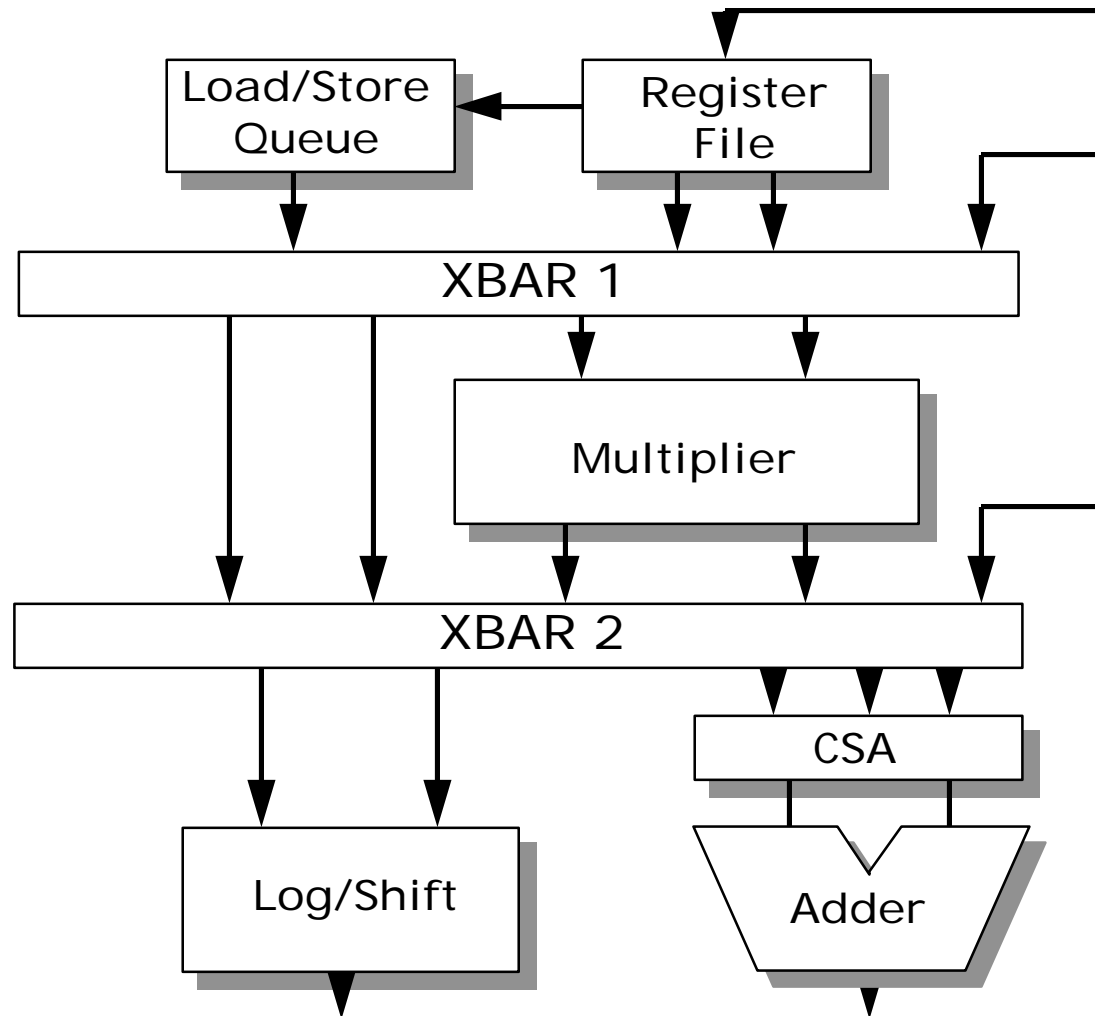


MMX Registers shared with FPU Mantissa & Tag Registers

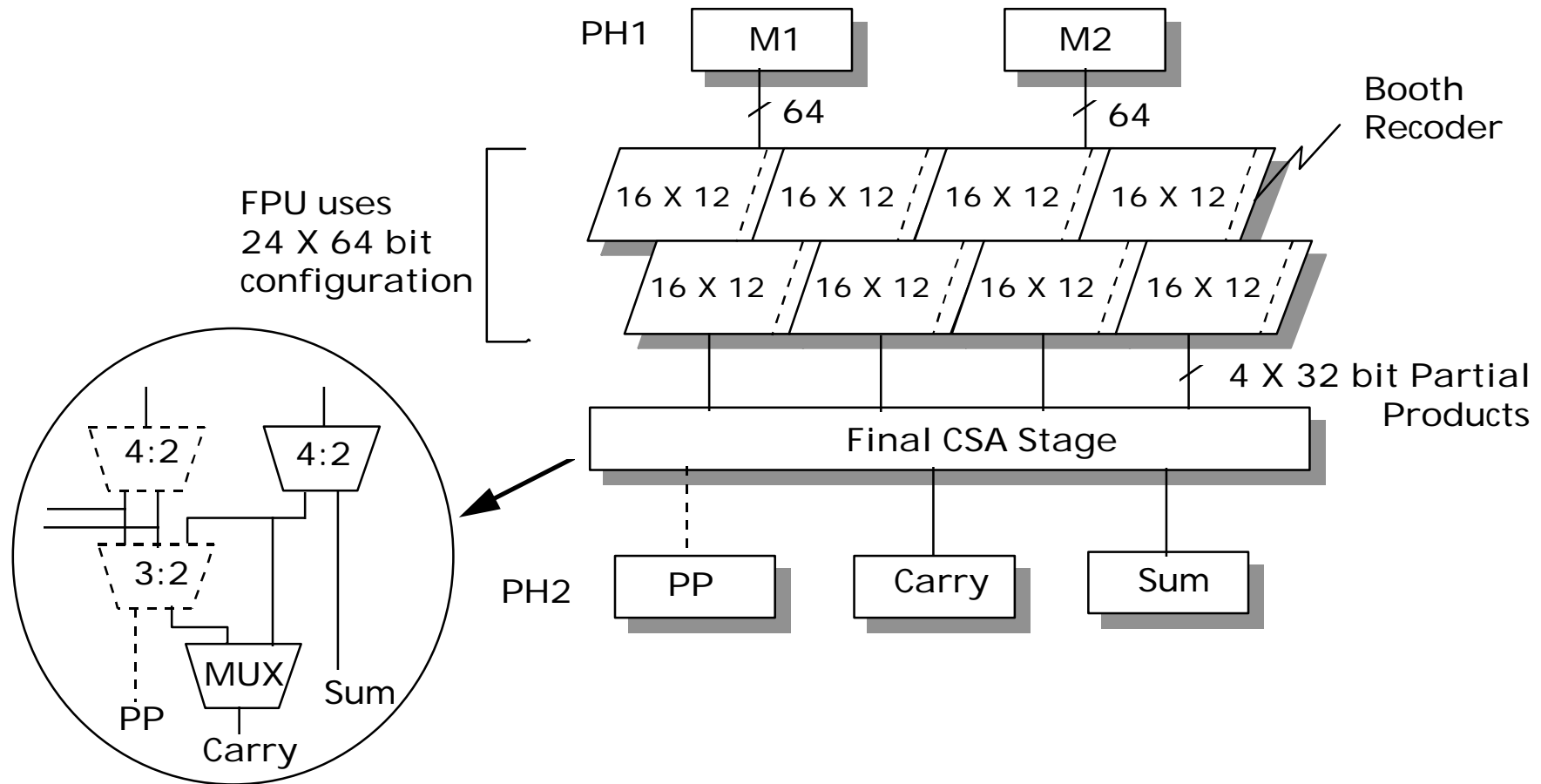
Integrated Multimedia & FPU

- ❑ Existing wide adders & multiplier in FP module can be efficiently subdivided to support SIMD processing
- ❑ FPU pipeline design can accommodate single-cycle multiply & fused multiply-add operations
- ❑ MMX compatibility prohibits simultaneous use of FPU hardware by both x86 FP & MMX operations
 - ◆ Why add dedicated functional unit/hardware?

M2 Multimedia/FPU Block Diagram

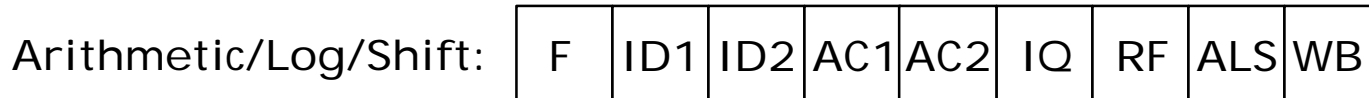


Multimedia Architecture: Multiplier Design



Multimedia Architecture: Pipeline Diagram

Multimedia instructions execute in a variable length pipeline
Instructions sent to instruction shelf in IQ stage



AC1 ==> Address Calculation
F ==> Fetch
ID1 ==> Instruction Decode
ID2 ==> Instruction Decode
AC2 ==> Address Calculation/Cache Access
WB ==> Write Back
IQ ==> Transfer to MMX Instruction Shelf
RF ==> FP Register File Access
ALS ==> Arithmetic/Logical/Shift
M1 ==> Multiply Stage 1
M2 ==> Multiply Stage 2



M2 Multimedia Performance

☐ Cache line locking == > Scratch Pad Memory

- ◆ Locked memory lines guarantee locality of reference
- ◆ Used by driver software code & data
- ◆ Predictable access speed yields real-time capability

☐ Pipeline accesses to L1 cache

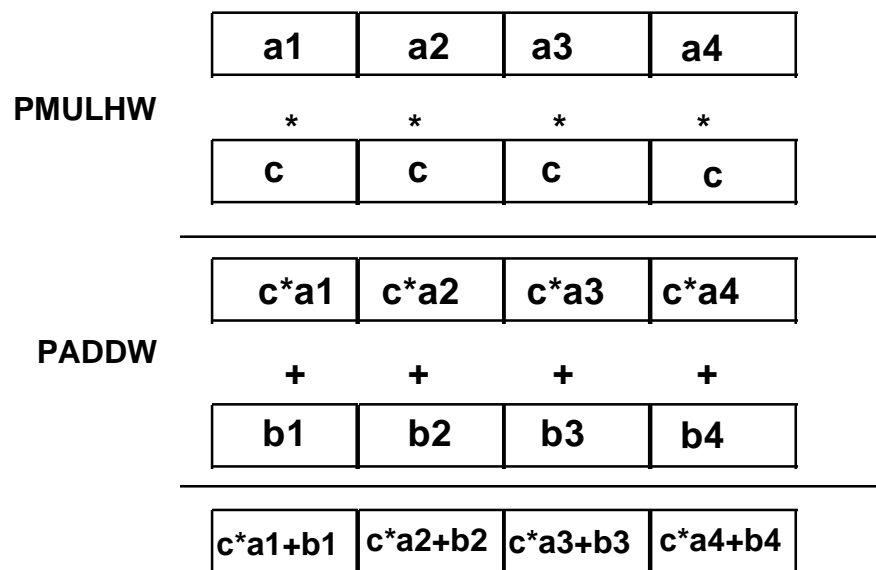
- ◆ Memory operand access at same speed as register access
- ◆ Combined with lockdown capability, permits giant “register files”

☐ Single-cycle execution

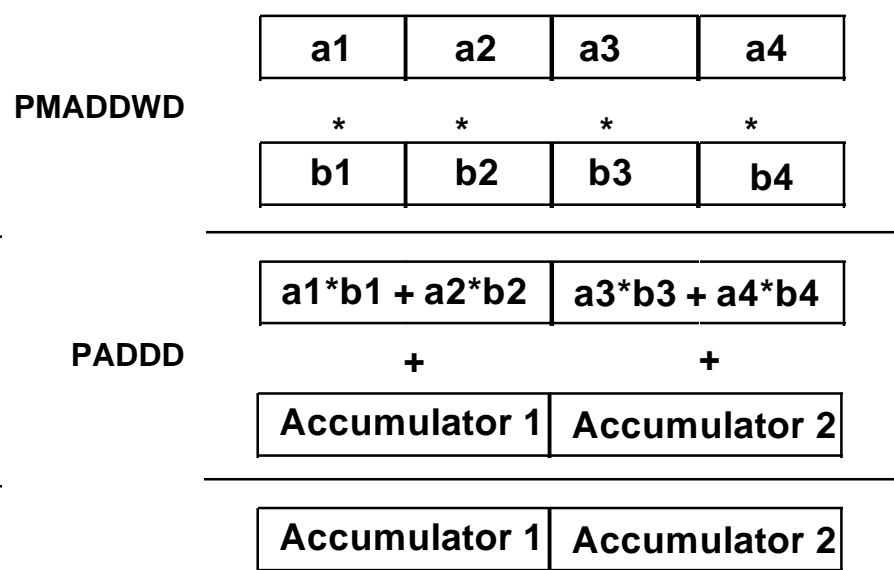
- ◆ Add, Subtract, Logical, & Shift operations execute in a single cycle
- ◆ Multiply & MAC execute with single-cycle throughput & latency of 2 clocks

Fast MAC with Cyrix MMX Technology

Multiply Accumulate is Basic Signal Processing Operation



2 clock, 16-bit precision MAC



3 clock, 32-bit precision MAC

4 Multiply Accumulates done in 2 to 3 cycles

IDCT with Cyrix MMX Technology

```

x86
-----
lea esp, input      mov ax, cx
mov ax, [esp]       imul c2
imul k0             add dx, MEM4
mov MEM1, dx        mov MEM4, dx
mov ax, [esp+ 2]    mov ax, cx
imul k1             imul -c6
mov bx, dx          mov bx, dx
mov ax, [esp+ 4]    add dx, MEM3
imul k2             mov MEM3, dx
mov cx, dx          mov cx, dx
mov ax, [esp+ 6]    mov dx, bx
imul k3             add bx, MEM1
mov ax, MEM1        sar bx, 01h

mov MEM1, ax        sub dx, MEM1
add ax, dx          imul c4
sub dx, MEM1
mov MEM1, bx
add bx, cx
sub cx, MEM1

mov MEM1, dx
mov MEM2, ax
imul -c2
mov MEM3, dx
mov ax, MEM2
imul -c6
mov MEM4, dx
    
```

X 4

```

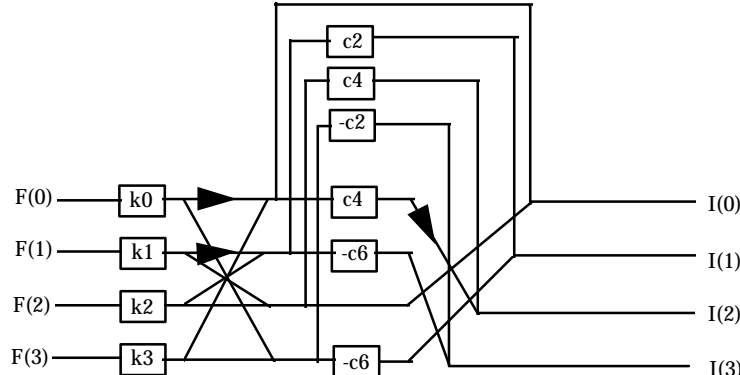
Cyrix MMX
-----
pmulhw mm0, k0
pmulhw mm1, k1
pmulhw mm2, k2
pmulhw mm3, k3

movq MEM, mm0
paddsw mm0, mm3
psubsw mm3, MEM
movq MEM, mm1
paddsw mm1, mm2
psubsw mm2, MEM
movq mm4, -c6
pmulhw mm4, mm0
pmulhw mm0, -c2

movq mm5, -c6
pmulhw mm5, mm2
paddsw mm5, mm0
pmulhw mm2, c2
paddsw mm2, mm4

movq mm7, c4
pmulhw mm7, mm3
paddsw mm3, mm1
psarw mm3, 1

pmulhw mm1, c4
psubsw mm1, mm7
    
```



Part of an IDCT algorithm

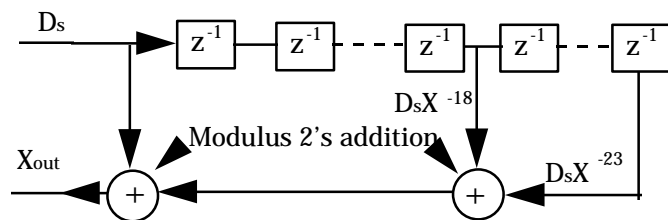
➤ Fused Multiply/Add

➤ Fused Multiply/Add

Cyrix MMX cycle count 31
Non-MMX cycle count 220

➤ Fused Multiply/Add

V.34 Modem Scrambler Using MMX



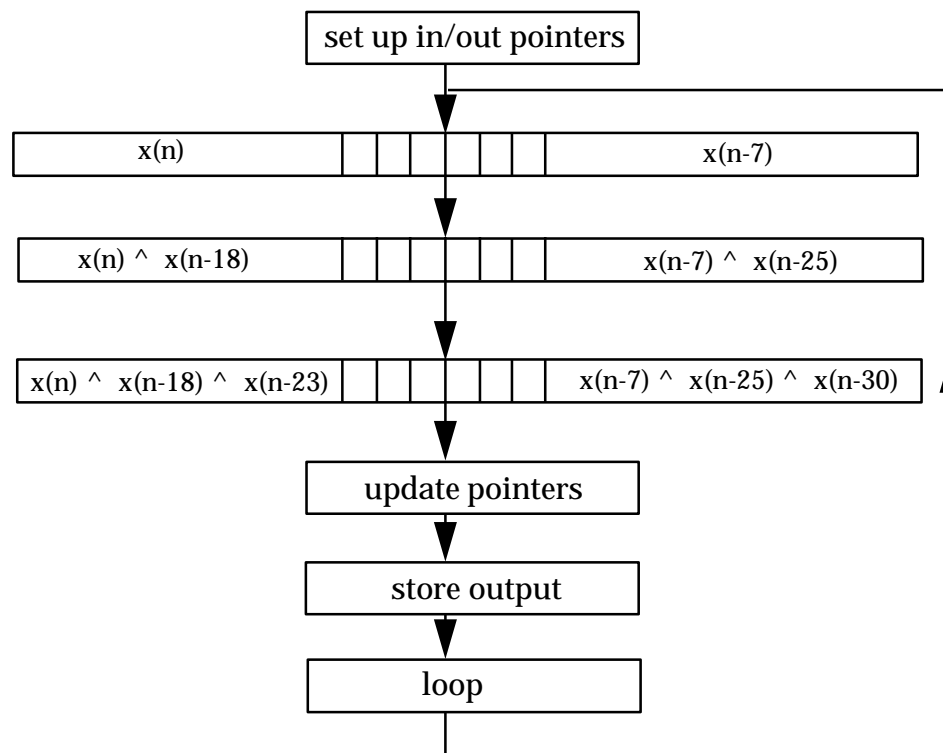
V.34 Modem Answer Mode Scrambler

```

lea esp, input
lea ebp, output
mov ecx, 010h

loop:
    movq mm0, [esp]
    add ebp, 08h
    pxor mm0, [esp-18]
    pxor mm0, [esp-23]
    add esp, 08h
    movq [ebp-8], mm0
    dec cx
    jnz loop
    
```

Pairable operations
for optimal 2
instructions per clock



Feature Set Comparison

Features	Cyrix MMX	Intel MMX	Sun VIS	HP 7100
Native Datapath Width	64 bits	64 bits	64 bits	32 bits
Total Partitions / Partition Size	8, 8 bit 4, 16 bit 2, 32 bit 1, 64 bit	8, 8 bit 4, 16 bit 2, 32 bit 1, 64 bit	4, 16 bit 2, 32 bit	8, 8 bit 4, 16 bit
Saturation Arithmetic	YES	YES	NO	YES
Multiplier Design	4x16x16	4x16x16	4x16x8	NA
Multiply/MAC Performance (Throughput/Latency)	1/ 2	1/3	1/3	-----
Integer Units Available During Multimedia Instruction Execution	1	1	2	0
Incremental Die Size/Cost	1.0 %	?	3.0 %	0.2 %
Cache Line Locking / Scratch Pad Memory	YES	NO	NO	NO

Results

- ❑ High performance multimedia capability, native to the processor
 - ◆ Efficiently integrated into existing hardware with minimal complexity & die area
- ❑ Programmable solution with general purpose x86 'style' instructions
 - ◆ Ease of programming
- ❑ MMX software compatible
- ❑ High level of performance/functionality with less than 20,000 transistors (less than a 1% die size increase)
- ❑ Development time of 9 mos. from specification to silicon
- ❑ Sampling December 1996; Production Q1 1997