

# The Evolution of DSP Processors

Presented to:  
CS152, University of California at Berkeley  
November 14, 1997

Jeff Bier  
Berkeley Design Technology, Inc.  
[www.bdti.com](http://www.bdti.com)  
[bier@bdti.com](mailto:bier@bdti.com)  
(510) 665-1600

Copyright © 1997 Berkeley Design Technology, Inc.



## Outline

- **DSP applications**
- **Digital filtering as a motivating problem**
- **The first generation of DSPs, with an example**
- **Comparison of DSP processors to general-purpose processors**
- **DSP evolution continues... later-generation DSPs and alternatives**
- **Conclusions**

## Who Cares?

- **DSP is a key enabling technology for many types of electronic products**
- **DSP-intensive tasks are the performance bottleneck in many computer applications today**
- **Computational demands of DSP-intensive tasks are increasing very rapidly**
- **In many embedded applications, general-purpose microprocessors are not competitive with DSP-oriented processors today**
- **1997 market for DSP processors: \$3 billion**

## Example DSP Applications

- Digital cellular phones
- Automated inspection
- Vehicle collision avoidance
- Voice over Internet
- Motor control
- Consumer audio
- Voice mail
- Navigation equipment
- Audio production
- Videoconferencing
- Pagers
- Music synthesis, effects
- Satellite communications
- Seismic analysis
- Secure communications
- Tapeless answering machines
- Sonar
- Cordless phones
- Digital cameras
- Modems (POTS, ISDN, cable,...)
- Noise cancellation
- Medical ultrasound
- Patient monitoring
- Radar

## Today's DSP "Killer Apps"

In terms of dollar volume, the biggest markets for DSP processors today include:

- Digital cellular telephony
- Pagers and other wireless systems
- Modems
- Disk drive servo control
  - Most demand good performance
  - All demand low cost
  -

Trends are towards better support for these (and similar) major applications.

## DSP Tasks for Microprocessors

- **Speech and audio compression**
- **Filtering**
- **Modulation and demodulation**
- **Error correction coding and decoding**
- **Servo control**
- **Audio processing (e.g., surround sound, noise reduction, equalization, sample rate conversion, echo cancellation)**
- **Signaling (e.g., DTMF detection)**
- **Speech recognition**
- **Signal synthesis (e.g., music, speech synthesis)**

## What Do DSP Processors Need to Do Well?

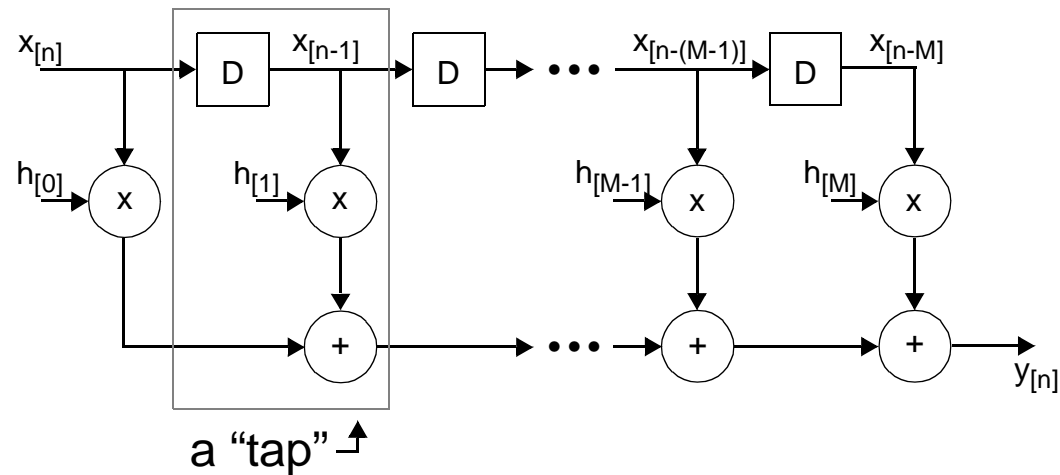
Most DSP tasks require:

- Repetitive numeric computations
- Attention to numeric fidelity
- High memory bandwidth, mostly via array accesses
- Real-time processing

Processors must perform these tasks efficiently while minimizing:

- Cost
- Power
- Memory use
- Development time

## FIR Filtering: A Motivating Problem



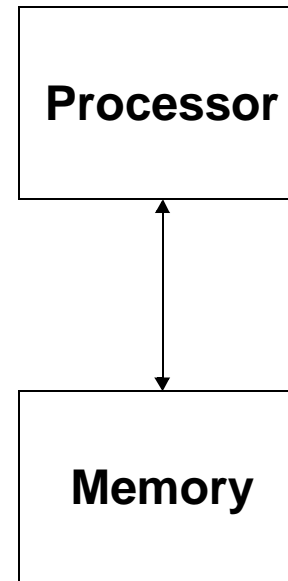
Each tap ( $M+1$  taps total) nominally requires:

- Two data fetches
- Multiply
- Accumulate
- Memory write-back to update delay line

## FIR Filter on Von Neumann Architecture

loop:

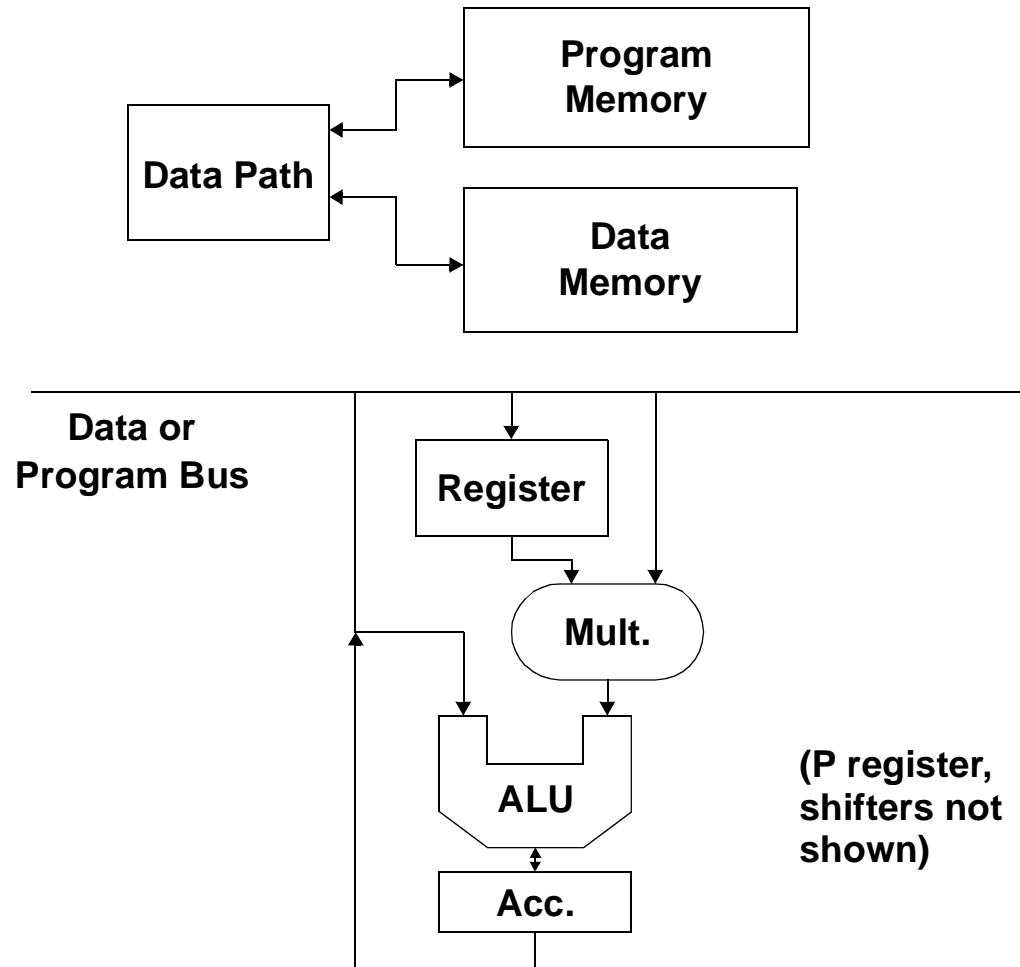
```
mov    *r0,x0
mov    *r1,y0
mpy    x0,y0,a
add    a,b
mov    y0,*r2
inc    r0
inc    r1
inc    r2
dec    ctr
tst    ctr
jnz    loop
```



**Problems: Bus / memory bandwidth bottleneck, control code overhead**

# First Generation DSP (1982): Texas Instruments TMS32010

- 16-bit fixed-point
- Harvard architecture
- Accumulator
- Specialized instruction set
- 390 ns MAC time (228 ns today)



(P register,  
shifters not  
shown)



## TMS32010 FIR Filter Code

Here X4, H4, etc. are direct (absolute) memory addresses:

```
LT      X4      ; Load T with x(n-4)
MPY     H4      ; P = H4*X4

LTD     X3      ; Load T with x(n-3); x(n-4) = x(n-3);
          ; Acc = Acc + P
MPY     H3      ; P = H3*X3

LTD     X2
MPY     H2
```

*etc.*

- Two instructions per tap, but requires unrolling

## Features Common to Most DSP Processors

- Data path configured for DSP
- Specialized instruction set
- Multiple memory banks and buses
- Specialized addressing modes
- Specialized execution control
-

## Data Path

- **Specialized hardware performs all key arithmetic operations in 1 cycle.**
- **Hardware support for managing numeric fidelity:**
  - **Shifters**
  - **Guard bits**
  - **Saturation**
- **Multiplies often take >1 cycle**
- **Shifts often take >1 cycle**
-

## Instruction Set

### DSP Processor

- Specialized, complex instructions
- Multiple operations per instruction

```
mac x0,y0,a  x:(r0)+,x0  y:(r4)+,y0
```

### General-Purpose Processor

- General-purpose instructions
- Typically only one operation per instruction

```
mov *r0,x0  
mov *r1,y0  
mpy x0,y0,a  
add a,b  
mov y0,*r2  
inc r0  
inc r1
```

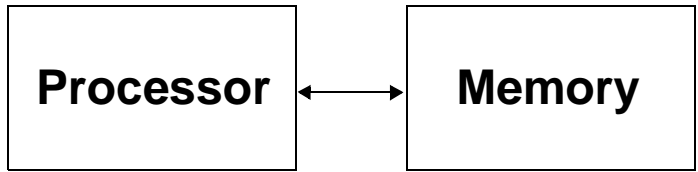
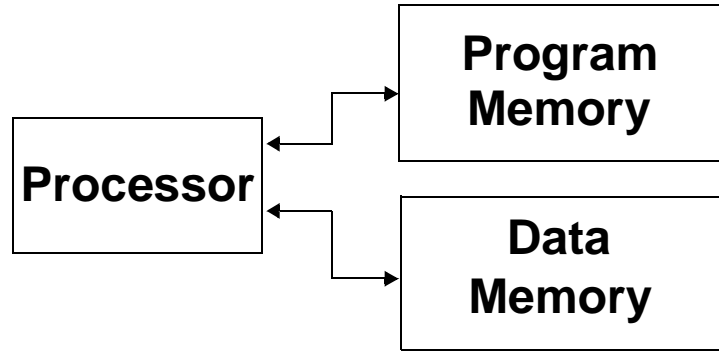
# Memory Architecture

## DSP Processor

- Harvard architecture
- 2-4 memory accesses/ cycle
- No caches—on-chip SRAM

## General-Purpose Processor

- Von Neumann architecture
- Typically 1 access/ cycle
- May use caches



## Addressing

### DSP Processor

- Dedicated address generation units
- Specialized addressing modes; e.g.:
  - Autoincrement
  - Modulo (circular)
  - Bit-reversed (for FFT)
- Good immediate data support

### General-Purpose Processor

- Often, no separate address generation unit
- General-purpose addressing modes

## Execution Control

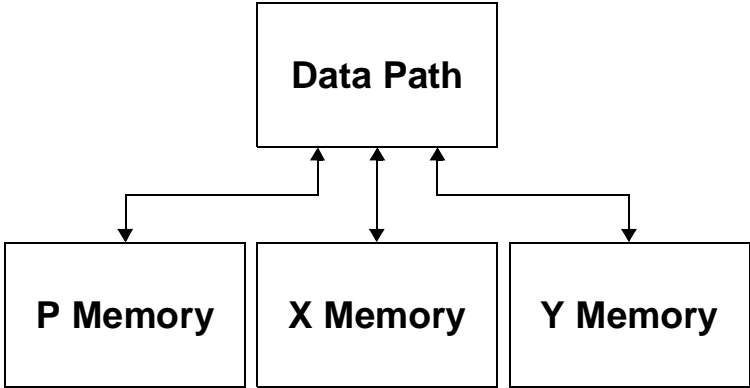
- **Hardware support for fast looping**
- **“Fast interrupts” for I/O handling**
- **Real-time debugging support**

## Specialized Peripherals for DSPs

- Synchronous serial ports
  - Parallel ports
  - Timers
  - On-chip A/D, D/A converters
  - Host ports
  - Bit I/O ports
  - On-chip DMA controller
  - Clock generators
- On-chip peripherals often designed for “background” operation, even when core is powered down.

# Second Generation DSPs (1987) Example: Motorola DSP56001

- 24-bit data, instructions
- 3 memory spaces (X, Y, P)
- Parallel moves
- Single- and multi-instruction hardware loops
- Modulo addressing
- 75 ns MAC (21 ns today)



```

move  #Xaddr, r0
move  #Haddr, r4
rep   #Ntaps
mac   x0, y0, a    x: (r0)+, x0    y: (r4)+, y0

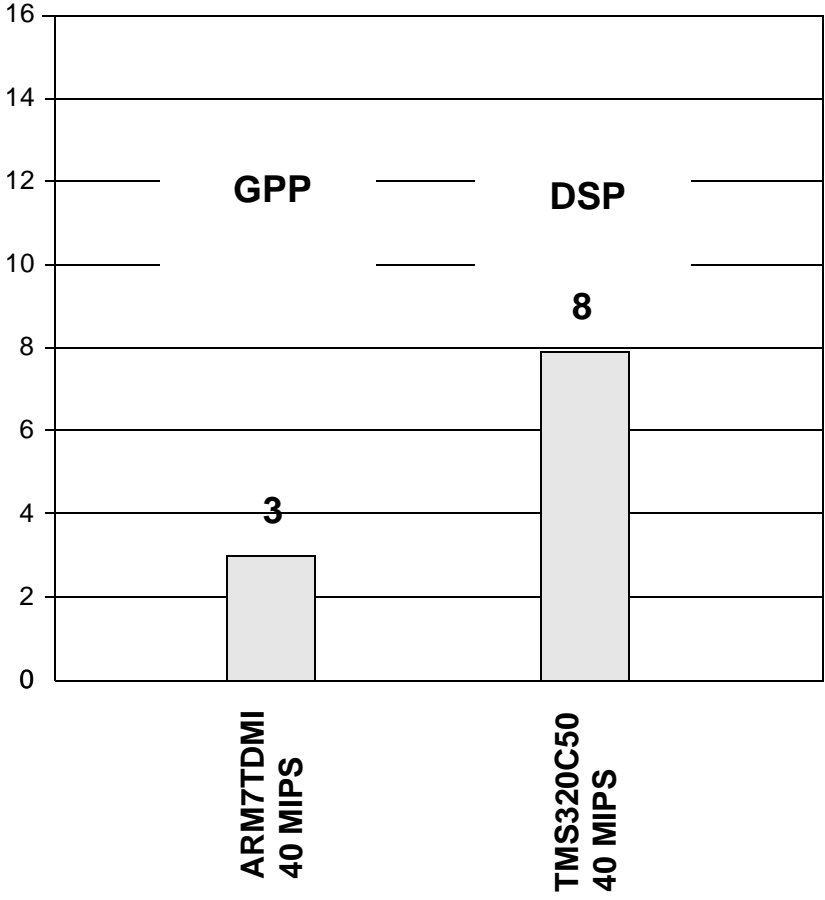
```

- Other second-generation processors: AT&T DSP16A, Analog Devices ADSP-2100, Texas Instruments TMS320C50



# Low-Cost General-Purpose Processor vs. Low-Cost DSP

Speed (BDTImarks™)



## **Third Generation DSPs (1995)**

### **Examples: Motorola DSP56301, TI TMS320C541**

- **Enhanced conventional DSP architectures**
- **3.0 or 3.3 volts**
- **More on-chip memory**
- **Application-specific function units in data path or as co-processors**
- **More sophisticated debugging and application development tools**
- **DSP cores (Pine and Oak from DSP Group, cDSP from TI)**
- **20 ns MAC (10 ns today)**

**Architectural innovation mostly limited to adding application-specific function units and miscellaneous minor refinements.**

- **Also, multiple processors/chip (TI TMS320C80, Motorola MC68356)**

## Fourth Generation (1997-1998)

**Examples: TI TMS320C6201, Intel Pentium with MMX**

Today's top DSP performers adopt architectures far different from conventional DSP processor designs.

- **Blazing clock speeds and superscalar architectures make some general-purpose processors, such as the PowerPC 604e, top floating-point performers, *despite lack of many DSP features***
- **Multimedia SIMD extensions, such as MMX, offer strong fixed-point performance on general-purpose processors**
- ***But strong DSP tools for general-purpose processors are lacking***
- **VLIW-like architectures, such as that of the TI TMS320C6201, achieve top performance via high parallelism and increased clock speeds**
-

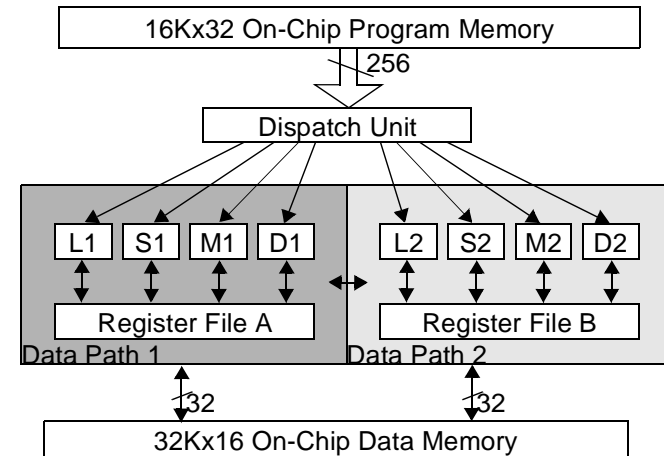
## VLIW

Very long instruction word (VLIW) architectures are garnering increased attention for DSP applications.

Notable recent introductions include Texas Instruments' TMS320C62xx and Philips' TM1000.

Major features:

- Multiple independent operations per cycle
  - Packed into a single large “instruction” or “packet”
- More regular, orthogonal, RISC-like operations
- Large, uniform register sets



## VLIW

### Advantages:

- Increased performance
- More regular architectures
  - Potentially easier to program; better compiler targets
- Scalable?

### Disadvantages:

- New kinds of programming/compiler complexity
- Code size bloat
  - High program memory bandwidth requirements
- High power consumption

## General-Purpose Processors are Catching Up

**“Go where the cycles are...”**

**General-purpose processors are increasingly adding DSP capabilities via a variety of mechanisms:**

- **Add single-instruction, multiple-data instruction set extensions (e.g., MMX Pentium)**
- **Integrate a fixed-point DSP processor-like data path and related resources with an existing  $\mu\text{C}/\mu\text{P}$  core (e.g. Hitachi SH-DSP)**
- **Add a DSP co-processor to an existing  $\mu\text{C}/\mu\text{P}$  core (e.g., ARM Piccolo)**
- **Create an all-new, hybrid architecture (e.g., Siemens TriCore)**

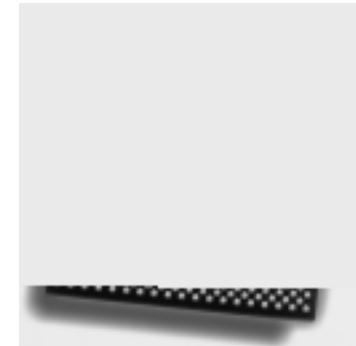
## The General-Purpose Processor Threat

High-performance general-purpose processors for PCs and workstations are increasingly suitable for some DSP applications.

- E.g., Intel MMX Pentium, Motorola/IBM PowerPC 604e

These processors achieve excellent to outstanding floating- and/or fixed-point DSP performance via:

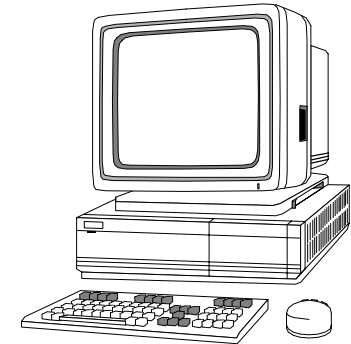
- Very high clock rates (200-500 MHz)
- Superscalar (“multi-issue”) architectures
- Single-cycle multiplication and arithmetic ops.
- Good memory bandwidth
- Branch prediction
- In some cases, single-instruction, multiple-data (SIMD) ops.



# High-Performance General-Purpose Processors

## Advantages:

- Strong DSP performance
- Already present in PCs
- Strong tool support for the major processors
- Cost-performance can rival that of floating-point DSPs



## Disadvantages:

- Lack of execution timing predictability
- Difficulty of developing optimized DSP code
- Limited DSP-oriented tool support
- High power consumption
- Cost-performance does not approach that of fixed-point DSPs

## Real-Time Suitability

The most important DSP applications are real-time applications.

- Many of these are “hard real-time” applications: failure to meet a real-time deadline creates a serious malfunction.

High-performance GPPs make heavy use of dynamic features:

- Caches, branch prediction, dynamic superscalar execution, data-dependent instruction execution times, etc.

These features result in timing behavior that appears to be stochastic.

-

---

## Example of Optimization Challenge

### Vector addition on PowerPC 604e:

@vec\_add\_loop:

```
lfsu fpTemp1,4(rAAddr)      # Load A data, ptr. update
lfsu fpTemp2,4(rBAddr)      # Load B data, ptr. update
fadds fpSum,fpTemp1,fpTemp2  # Perform add operation
stfsu fpSum,4(rCAddr)       # Store sum, ptr. update
bdnz @vec_add_loop          # loop
```

**Q: How many instruction cycles per iteration?**

# TMS320C6201 FIR Filter Inner Loop

LOOP:

```

      ADD      .L1  A0,A3,A0      ; Sum0 += P0
| |
      ADD      .L2  B1,B7,B1      ; Sum1 += P1
| |
      MPYHL    .M1X A2,B2,A3      ; P0 = h(i)*s(i)
| |
      MPYLH    .M2X A2,B2,B7      ; P1 = h(i+1)*s(i+1)
| |
      LDW      .D2  *B4++,B2      ; h(i) & h(i+1)
| |
      LDW      .D1  *A7--,A2      ; s(i) & s(i+1)
| |[B0] ADD    .S2  -1,B0,B0      ; Cond. dec loop counter
| |[B0] B      .S1  LOOP        ; Cond. Branch to LOOP

```

Latencies:

- Multiply: 2 cycles; load: 5 cycles; branch: 6 cycles

Predicated execution for all instructions.



# MMX Pentium FIR Filter Inner Loop

```
loop1:
    pmaddwd mm0, COEFaddr[edi]           4 MADs (reg, mem)
    padd    mm7, mm2                     Complete earlier accum.
    pmaddwd mm1, COEFaddr[edi+8]         next 4 MADs
    padd    mm7, mm3                     Complete earlier accum.
    movq    mm2, [esi+16]                 Load next 4 data items
    movq    mm3, [esi+24]                 Load next 4 data items
    padd    mm7, mm0                     Complete earlier accum.
    pmaddwd mm2, COEFaddr[edi+16]        Again, with feeling
    padd    mm7, mm1                     (unrolled to avoid
    pmaddwd mm3, COEFaddr[edi+24]        load-related stall)
    movq    mm0, [esi+32]
    movq    mm1, [esi+40]
    add     edi, 32                       Update coeff. ptr.
    add     esi, 32                       Update data ptr.
    dec     ecx                           Decrement loop count.
    jnz     loop1                         Branch to top of loop
```



## MMX Pentium FIR Filter Inner Loop

### Latencies:

- **Multiply: 3 cycles (not 3 *instructions*)**

### Superscalar execution

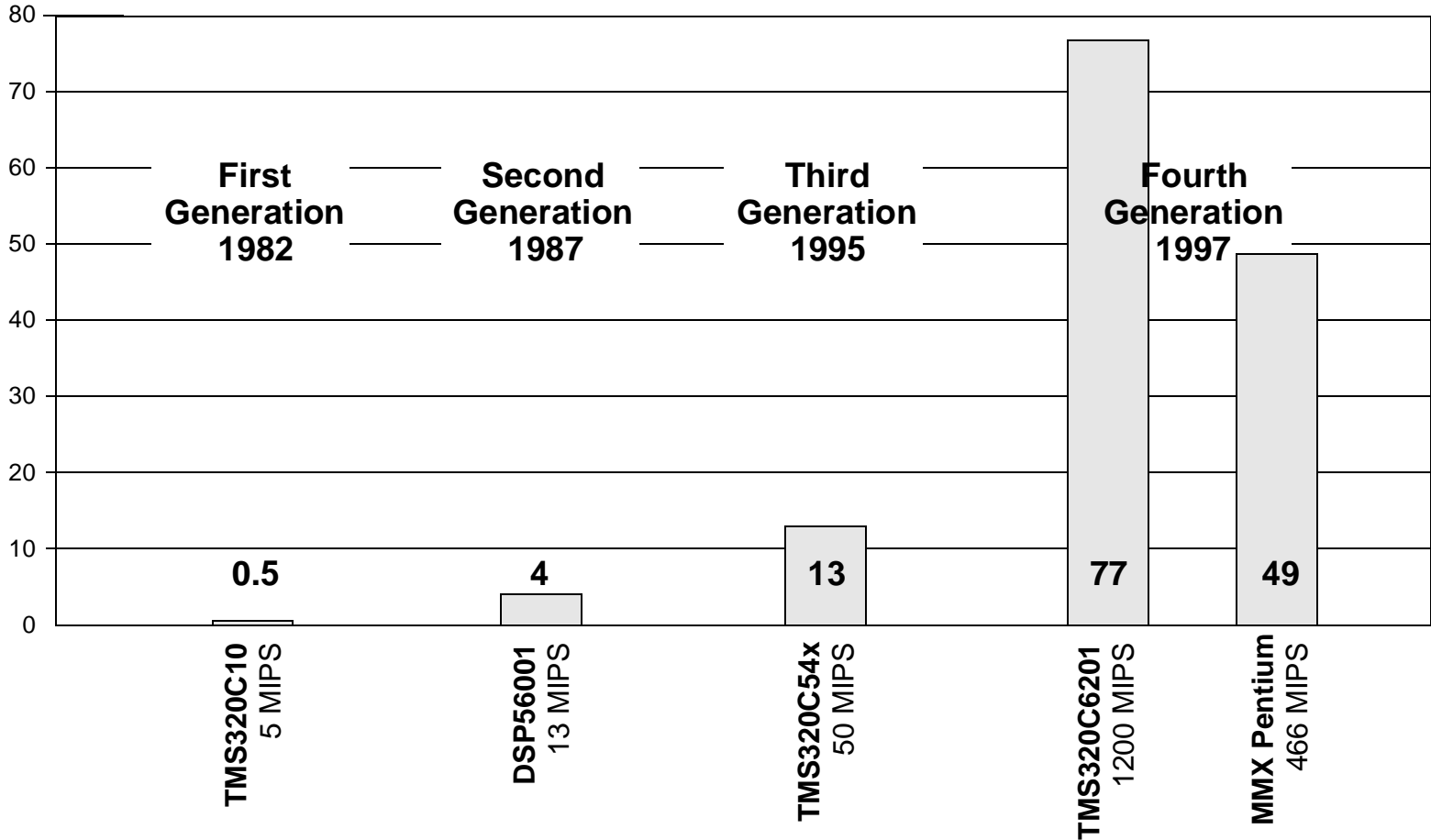
- **Up to two instructions/cycle**
- **Can pair one simple MMX instruction with another simple or complex MMX instruction or non-MMX integer instruction**
- **Complicated pairing rules**

### Branch prediction

**Throughput: ~2 16-bit MACs/cycle**

# Processor DSP Speed

## BDTmarks™



## Conclusions

- **DSP processor performance has increased by a factor of about 150x over the past 15 years (~40%/year)**
- **Processor architectures for DSP will be increasingly specialized for applications, especially communications applications**
- **General-purpose processors will become viable for many DSP applications**
- **Users of processors for DSP will have an expanding array of choices**
- **Selecting processors requires a careful, application-specific analysis**

## For More Information

<http://www.bdti.com>

Collection of BDTI's papers on DSP processors, tools, and benchmarking.

<http://www.eg3.com/dsp>

Links to other good DSP sites.

*Microprocessor Report*

For info on newer DSP processors.

*DSP Processor Fundamentals*,  
BDTI

Textbook on DSP Processors

*IEEE Spectrum*, July, 1996

Article on DSP Benchmarks

*Embedded Systems Prog.*  
October, 1996

Article on Choosing a DSP Processor

Or, Join BDTI...

We're hiring (see [www.bdti.com](http://www.bdti.com))

