COMP ENG 4TL4:

Digital Signal Processing

Notes for Lectures #31 & #32

Tuesday, November 25 &
Wednesday, November 26, 2003

8. Introduction to DSP Architectures

4TL4 – DSP

Jeff Bondy and Ian Bruce

DSP Applications

- High volume embedded systems
  - Cell phones
  - Hard Drives
  - CD Drives
  - Modems
  - Printers
- High performance data processing
  - Sonar
  - Wireless Basestations
  - Video/Data Transport

Resources

- www.bdti.com (Started kernel speed benchmarking)
- www.eembc.org (Benchmarks for almost any application)
- http://www.techonline.com/community/tech_group/dsp
- (Motorola) http://e-www.motorola.com/webapp/sps/site/homepage.jsp?nodeId=06M10NcX0Fz
- (TI) http://dspvillage.ti.com/
- (Analog Devices) http://www.analog.com/Analog_Root/static/technology/dsp/beginnersGuide/index.html/

In ONE Cycle

- Fetch instruction
- Decode instruction
- Calculate address
- Fetch data
  - L2 hopefully, or else increase latency by going off chip, update L2 state
  - L2 → L1, update L2 and L1 state
  - L1 → Registers
  - Registers → ALU
- Compute instruction
- Write result
- Update data pointers
- Update instruction pointer

FETCH

DECODE

READ

EXECUTE

Intro to DSP Architecture

- What and Why of MACs
- Multiple Memory Accesses
- Fast Address Generation Units
- Fast Looping
- Specialized Instruction Sets
- Lots of I/O

Typical DSP Heart

Data Buses

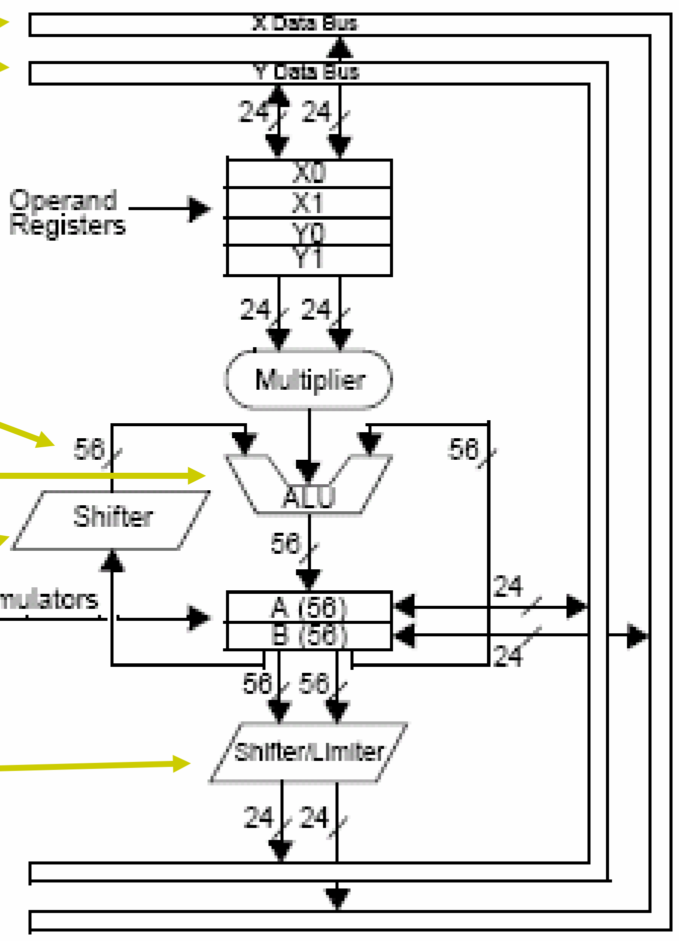
Abundant Instant Memory Access

Huge ALU Dynamic Range

FAST ALU

Chained Shifter for repetitive calculations

Barrel Shifter



MACs – Multiply Accumulates

- In one clock cycle the ALU of a DSP can do a multiply and addition.
  - Used in:
    - Vector dot products
    - Correlation
    - Filters
    - Fourier Transforms
- In addition to ALU changes the bus structure must also change

Multiple Memory Accesses

- Complete MANY memory accesses in a single clock cycle
  - Processor can fetch instructions while also fetching the operands or storing to memory
    - During FIR filter can operate a multiply and accumulate while loading the operands and coefficient for the next cycle
  - Three reads and one or two writes per cycle
- This requires multiple memory buses on the same chip, not simply an address and data bus

Dedicated Address Generation

- One or more address generation units, so the processor doesn’t tie up the ALU/main data path
  - Register indirect addressing with post-increment
  - Modulo addressing
  - Bit reversed addressing

Efficient looping

- For repetitive, or branching calculations. For-next loops in a general purpose algorithm kill performance with calculating conditions, checking loop logic and setting JUMPs.
  - <loop> and <repeat> instructions allow jumping to top of loop while incrementing and testing loop logic in a SINGLE cycle.
- Delayed branching
- Low~Mid range DSPs have 3~5 stage pipelines to get rid of NOPs

Pipelining

None *(Motorola 560xx, ie. OLD)*

Fetch	Decode	Read	Execute				
				Fetch	Decode	Read	Execute

Pipelined *(Most conventional DSP processors)*

Fetch	Decode	Read	Execute		
	Fetch	Decode	Read	Execute	
		Fetch	Decode	Read	Execute

Superscalar *(Pentium, MIPS)*

Fetch	Decode	Read	Execute	
	Fetch	Decode	Read	Execute
	Fetch	Decode	Read	Execute

Instruction Sets

- Maximize use of underlying hardware
  - Increase instruction efficiency, complex instructions, many different operations/accesses per call.
- Minimize amount of memory used
  - Instructions must be short, restrict flexibility such as register choice, multiple operation connections.
    - DSPs have fewer/smaller registers, use mode bits to morph some operations, highly individualized and irregular instructions sets.
- You can compile C code into a DSP target but for efficient code it MUST BE HAND OPTIMIZED.

Lots of I/O

- Large array and amount of I/O versus microprocessor
- Specialized instruction set and hardware to deal with fast off-chip memory access such as DMA

GPP exceptions

- General Purpose Processors have fought back because of the huge market that DSPs were beginning to encroach on
  - MMX (Pentium)
  - SSE (Pentium)
  - SH-2 (Strong Arm)
  - Power PC (AltiVec)
  - UltraSPARC (VIS – Visual Instruction Set)
- Strange? Isn't this what CRAY was saying about vectorizing processors was the most powerful architecture?

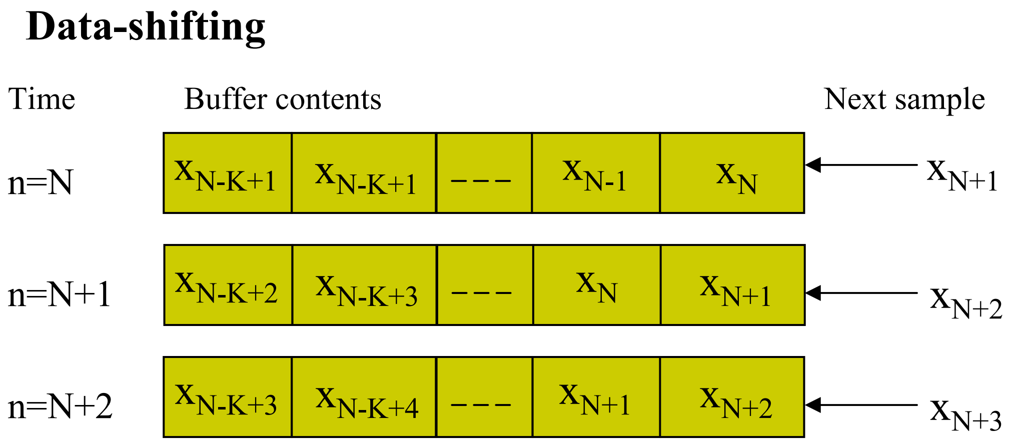
Pentium 266 MMX Versus TMS32062x

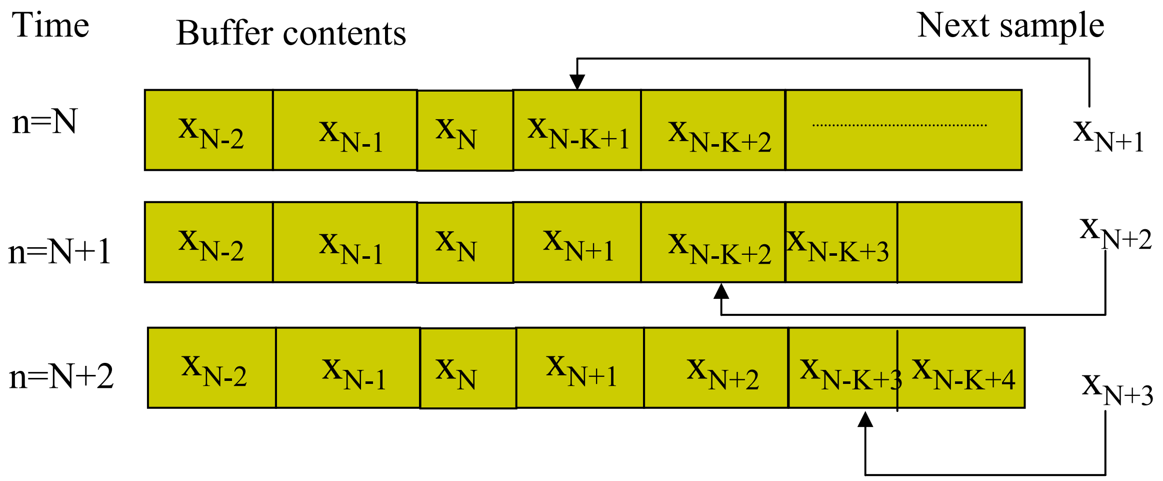
- 4x More power
- 1/3 MIPS
- 1/3 256-FFT completion time
- Same price
- 4x Die Size
- Pentium needs extensive cooling

Modulo Addressing

Modulo addressing

- implementing circular buffers and delay lines





DSP Characteristics

- Arithmetic Format
- Bus Width
- Speed
- Memory/Bus/Instruction architecture
- Development Tools
- Power Consumption
- Cost
- Specialized Hardware

Arithmetic

- Fixed Point or Floating Point?
  - Fixed: numbers are integers in a set range
  - Float: numbers are represented by a mantissa and exponent
  - Fixed: cheaper, higher volume, faster, less power, horrible amounts of time tweaking and rescaling at different points in a calculation. 95% of DSP Market.
  - Float: Wider dynamic range, larger die size, easier, becoming more available. 5% of DSP Market.

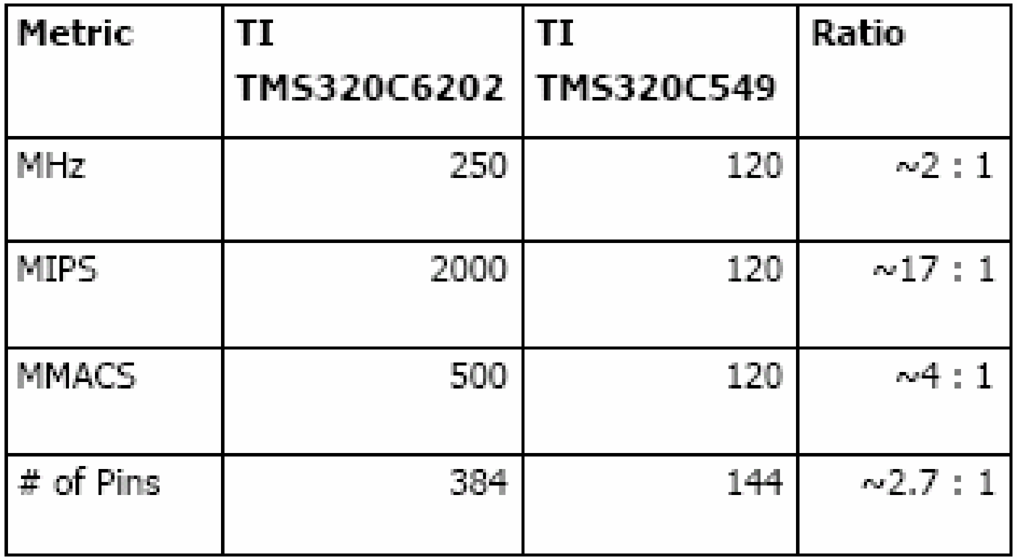
Bus Widths

- Fixed: usually 16 bit data bus
- Float: 32 bit, standard IEEE mantissa-exponent format
  - Motorola DSP56300 family is a widely used, notable exception, it’s 24 bit fixed point.
    - Almost the defacto standard for audio processing applications. Why? Think about the dynamic range of the auditory system: Your ear has about 120 dB of dynamic range.
    - So w/ linear, uniform coding @ 16 bits and 24 bits:
      - 10^(120/20)/(2^16) = 15.25
      - 10^(120/20)/(2^24) = .0595

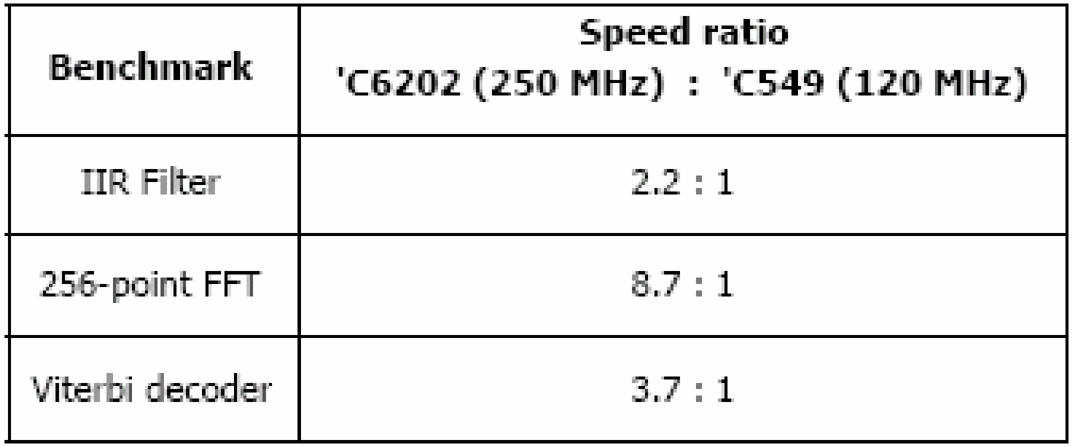
Speed

- "Specmanship" has inundated all aspects of silicon specification so beware
  - MHz: What is the on-chip clock speed?
  - MIPS: Meg. Instructions Per Second, the reciprocal of the fastest instructions time divided by 10^6.
  - MMACS: Meg. Multiply-Accumulates per Second.
  - Kernel Times: For specific tasks, 256 point FIR, Radix-2 FFT, what is the absolute time?

Specmanship of Speed



Metric	TI TMS320C6202	TI TMS320C549	Ratio
MHz	250	120	~2 : 1
MIPS	2000	120	~17 : 1
MMACS	500	120	~4 : 1
# of Pins	384	144	~2.7 : 1



Benchmark	Speed ratio 'C6202 (250 MHz) : 'C549 (120 MHz)
IIR Filter	2.2 : 1
256-point FFT	8.7 : 1
Viterbi decoder	3.7 : 1

* www.bdti.com, "Independent DSP benchmark results for the latest processors"

Memory

- Most built around fast bus architecture
  - Harvard architecture splits Address and Data buses and memory locations (versus von Neumann)
  - Cache to fetch instructions freeing up bus to fetch or write.
- Embedded systems have smaller memory needs
- Variable instruction sizes and memory sizes

Development Tools

- S/W Tools: assemblers, linkers simulators, debuggers, compilers, code libraries, RTOS
  - DSPs are compiler unfriendly. Unusual and complex instruction sets. C/Ada produce bloated code, intricacies of number crunching almost always coded in Assembler. Floating point processors usually compile cleaner then Fixed
- H/W Tools: emulators, development boards
- JTAG: IEEE 1149.1, on chip debugging and emulation. Scan based emulation, set breakpoints like a S/W IDE, poll and set registers while paused.

System Management

- Minimizing Vcc to reduce power consumption
- Sleep modes
    - Turn off entire sections of the chip, ie. Interface for an unconnected protocol
    - Event activation with different latencies, ie. Packet datacom, doesn’t decode a packet unless device address is pinged
- Programmable on-chip clock distribution
    - Clock Dividers for integer differences that arise in digital communication receivers
    - Phase-Locked-Loops (PLLs) for fine control over jitter and frequency

COST!!

- Limiting factor of any REAL design
- Packaging can be 50% of real cost, product plus manufacturing. Many companies are going to BGA (Ball Grid Array) packs versus P/T QFP, (Plastic/Thin Quad Flat Pack), making them more expensive and IMPOSSIBLE to rework.

Analog Devices: ADSP-2116x SHARC

- Has special I/O and instructions that accelerates multiprocessor connections
  - 6 processors strung together with bus arbitration
  - Any processor can access the internal memory of any other processor
- Also replicates the entire operational block, giving you two powerful processors and making extensive use of SIMD (more on this later).

Low Range DSPs

- Analog Devices
  - ADSP-210x
- Motorola
  - DSP-560xx
- Texas Instruments
  - TMS320F28x
- ~40 MHz Clock, usually used as a souped up microcontroller.
- Disk drives, cordless phones, ISM band equipment

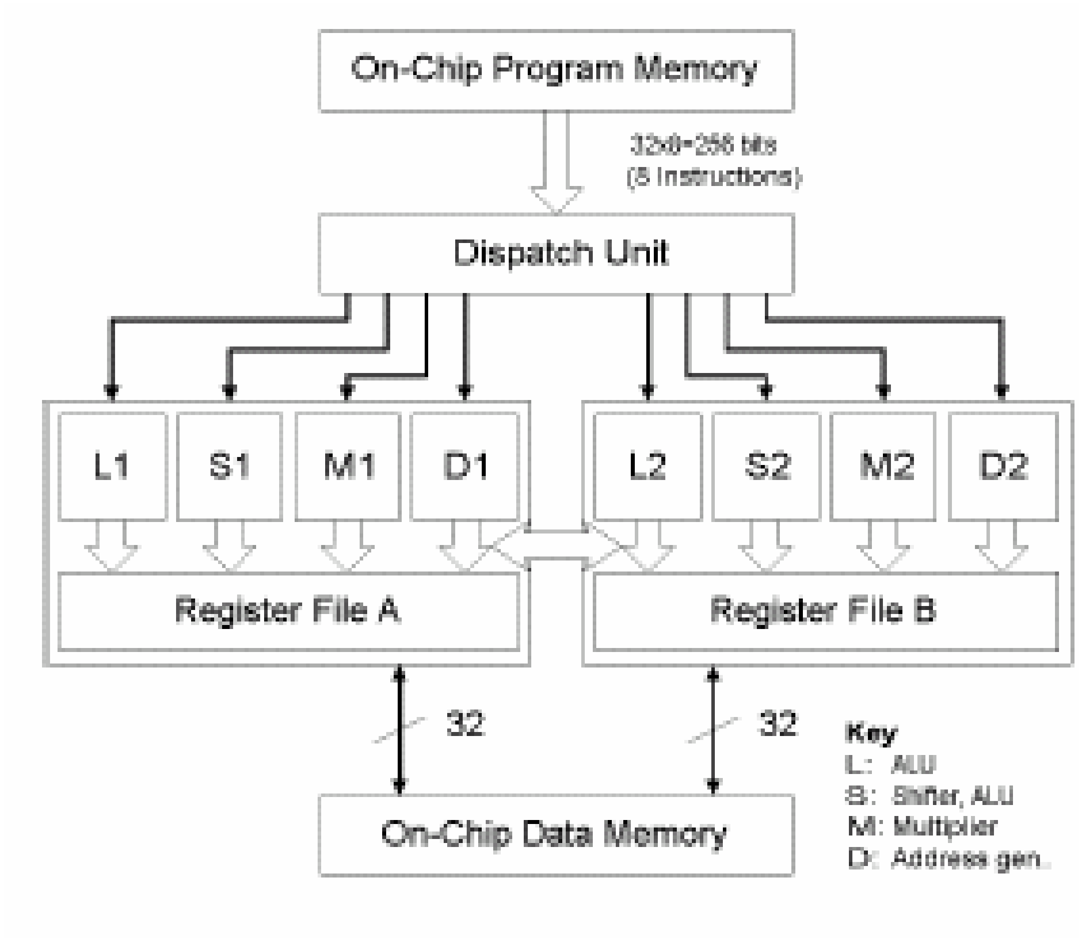
Mid Range DSPs

- Analog Devices
  - ADSP-218x
- Motorola
  - DSP-563xx
- Texas Instruments
  - TMS320C52x
- 150 MHz, cell-phones, modems.

Very Large Instruction Word

- TI – TMS320c62xx First DSP
- VLIW use simple, orthogonal, RISC based instruction sets. String several 4, 8 or 16 bit instructions together that use different parts of the H/W to execute every cycle
- Compile cleaner because of simpler instruction sets, but hand-optimization is harder because of heuristic scheduling for the H/W components.

TMS320C62xx



One instruction is fed into two sets of four execution units.

Instead of the MAC-ALU serial structure you have them in parallel, meaning each top-down operation is less complex, but may take more instructions

VLIW v Superscalar

- VLIW produces code AT COMPILATION that identifies which instructions are completed in parallel
- Superscalar hardware AT EXECUTION identifies which instructions are completed in parallel

!! That means that for different iterations through a loop a different order of instructions could be completed. Unusual processing times

Single-Instruction Multiple Data

- Instead of splitting instructions, splits operational blocks. A 16 bit MAC turns into two 8 bit MACs.
- Allows a processor to execute multiple instances of the same operation using different data.

Choose Your Own Adventure

- What DSP code looks like
- DSP Devices that you might be working with
- Short introduction to DSP on video cards
- MMX/SSE overview
- Reading DSP spec sheets

FIR Filters with Assembler – MOT DSP563xx

```
main()
{
      /* Control logic system setup and whatnot
      ........................................      */

      // Begin with an assembler call
      asm
      {
          move        #AADDR,r0           // Register r0 load, will contain coeffs
          move        #BADDR,r4           // Register r4 load, will contain data
          move        #N-1,m4             // Load loop control
          move        m4,m0               // move loop control
(2)       movep       y:input,y:(r4)      // move peripheral data from Input "y"
(1)       clr  a                  x:(r0)+,x0 y:(r4)-,y0        // clear accumulator, memory moves
(5)       rep #N-1                                             // Repeat next instruction
(N)       mac         x0,y0,a     x:(r0)+,x0 y:(r4)-,y0  // Multiply Accumulate, update registers
(1)       macr        c0,y0,a     (r4)+                        // Rounding and scaling (set by c0)
(1)       movep       a,y:output                         // move accumulator output to peripheral "y"
      }
      // End assembler call

      /* Control logic system setup and whatnot
      ........................................      */
}
```

Differences in Assembler codes

```
main:
        bits   %fmode,  2             /* Enable Q15  */
        lda    r13, Xdata
        lda    r15, Dbuffer
        lda    r11, Yout
        mov    r10, 40                /* Filter size, Nlen = 40 $$$ */
        mov    r9,  200  /* Input data size (Nsamp = 200) $$$ */
        mov    %cb1_beg, r15
        mov    r8,  r10               /* r8 = Nlen */
        add    r8,  1                 /* r8 = Nlen+1 */
        add    r10, -1                /* Adjust for loop counter  */
        add    r8,  r15
        mov    %cb1_end, r8           /* CB size = Nlen+1  */
        bits   %smode,  2             /* Enable CB1 (for r15) */
        mov    r6,  10000
        mov    %timer0, r6            /* Initialize Timer count */
                                      /* Worst case cycle count = */
                                      /* (Nlen + 6)*Nsamp */

per_sample:
        ldu    r7,  r13,  1  /* "Acquire" new sample from "Xdata",*/
                   /* a pre-stored input buffer -- in a */
                   /* real-time application, this new */
                   /* sample may come from a different */
                   /* task or an external device, etc. */
        mov    %loop0,  r10
        lda    r14, Hfilter
        psub.a r0,  r0                /* Clear accumulator's 32-bits */
        st     r7,  r15               /* Store new sample into Dbuffer */
        mov    %guard, 0              /* Clear Guard bits */
        bits   %tc,  7                /* Timer0 starts ticking */
```

```
fir_loop:
        ldu    r4,  r14, 1            /* Filter coefficient  */
        ldu    r2,  r15, 1            /* Sample from Data buffer (circular) */
        mac.a  r2,  r4
        agn0   fir_loop
        bitc   %tc, 7                 /* Timer0 frozen */
        round.e r0, r0                /* Filter output is rounded */
        stu    r1,  r11, 1            /* Filter output is stored  */
flag1:
        nop
        add    r9,  -1
        bnz    per_sample
        nop
filter_done:                          /* Set an SDBUG break-point here */
     nop          /* Note: ZSIM or RTL need a HALT here */
     nop
     br filter_done
     nop
```

This is from the LSI website, and in my mind, one of the reasons why they have lost some market share

Analog Devices Overview

CHIPS

Vendor	Family	Floating, Fixed, or Both	Data Width	Instruction Width	Core Clock Speed [1]	**BDTImark2000™** *BDTIsimMark2000™* [2]	Total On-Chip Memory, Bytes	Core Voltage	Unit Price [3]	Notes
Analog Devices	ADSP-218x	Fixed point	16 bits	24 bits	80 MHz	**240**	20 K–256 K	1.8	$4–24	Many family members w/ assorted peripherals
	ADSP-219x	Fixed point	16 bits	24 bits	160 MHz	**410**	20 K–160 K	2.5	$10–24	Enhanced version of the ADSP-218x
	ADSP-2116x (SHARC)	Floating point	32/40 bits	48 bits	100 MHz	**470**	128 K–512 K	1.8, 2.5	$22–99	Features SIMD, strong multiprocessor support
	ADSP-BF53x (Blackfin)	Fixed point	16 bits	16/32 bits	600 MHz	**3360** [5]	84 K–308 K	0.7–1.2, 1.0–1.6	$6–35	Dual-MAC DSP with variable speed and voltage
	ADSP-TS20x (TigerSHARC)	Both	8/16/32/40 bits	32 bits	600 MHz	**6150** [5]	512 K–3 M	1.0, 1.2	$35–299	4-way VLIW with SIMD capabilities; uses eDRAM

*** From http://www.bdti.com**

Motorola Devices Overview

CHIPS										
Vendor	Family	Floating, Fixed, or Both	Data Width	Instruction Width	Core Clock Speed [1]	**BDTImark2000™** *BDTIsimMark2000™* [2]	Total On-Chip Memory, Bytes	Core Voltage	Unit Price [3]	Notes
Motorola	DSP563xx	Fixed point	24 bits	24 bits	240 MHz	**710**	24 K–384 K	1.5, 1.6, 1.8, 3.3	$4–56	PCI bus, DMA, can run '560xx code unmodified
	DSP568xx	Fixed point	16 bits	16 bits	40 MHz [6]	**110**	28 K–152 K	2.5, 3.3	$3–15	Contains many microcontroller-like features
	DSP5685x	Fixed point	16 bits	16 bits	120 MHz	*340*	36 M	1.8	$6–12	Enhanced version of the '568xx
	MSC810x (SC140)	Fixed point	16 bits	16 bits	300 MHz	**3370** [7]	512 K–1436 K	1.6	$90–195	Based on quad-MAC SC140 core; '8102 uses 4 cores

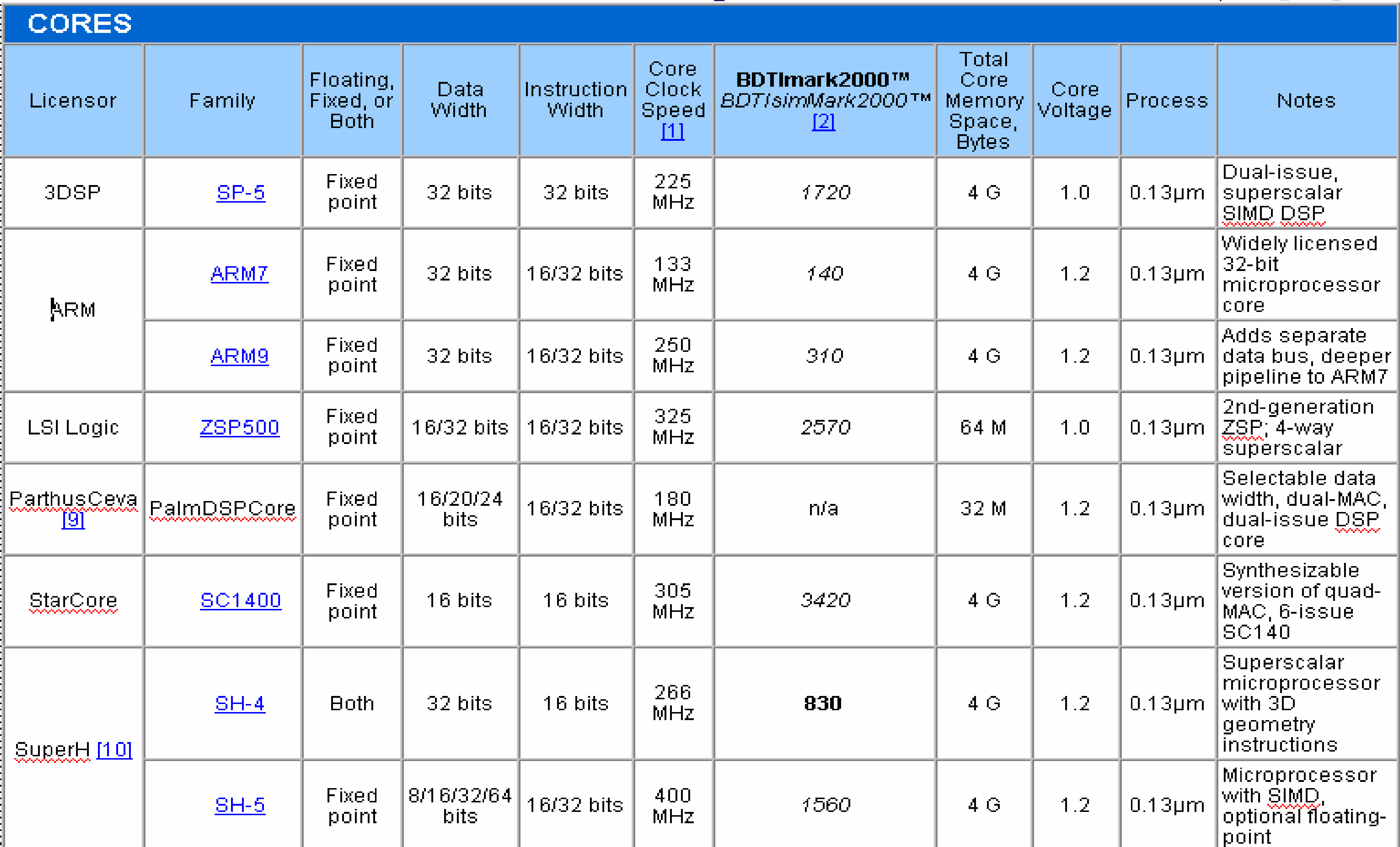
*** From http://www.bdti.com**

TI Devices Overview

* From http://www.bdti.com

CHIPS										
Vendor	Family	Floating, Fixed, or Both	Data Width	Instruction Width	Core Clock Speed [1]	**BDTImark2000™** *BDTIsimMark2000™* [2]	Total On-Chip Memory, Bytes	Core Voltage	Unit Price [3]	Notes
TI	TMS320 F24x	Fixed point	16 bits	16/32 bits	40 MHz	n/a	18 K–1120 K	3.3, 5.0	$3–15	Hybrid microcontroller/DSP
	TMS320 F28x	Fixed point	32 bits	16/32 bits	150 MHz	n/a	164 K–292 K	1.8	$16–18	Hybrid microcontroller/DSP; compatible w/ ’C24x
	TMS320 C3x	Floating point	32 bits	32 bits	75 MHz [6]	n/a	264 K–2304 K	3.3, 5.0	$10–213	Cost-competitive with fixed point DSPs
	TMS320 C54x	Fixed point	16 bits	16 bits	160 MHz	**500**	24 K–1280 K	1.5, 1.6, 1.8, 2.5, 3.3	$4–109	Many specialized instructions
	TMS320 C55x	Fixed point	16 bits	8–48 bits	300 MHz	**1460**	80 K–376 K	1.26, 1.5, 1.6	$5–20	Next generation ’C5xxx architecture; dual-issue, dual-MAC DSP
	TMS320 C62x	Fixed point	16 bits	32 bits	300 MHz	**1920**	72 K–896 K	1.5, 1.8	$9–102	8-way VLIW
	TMS320 C64x	Fixed point	8/16 bits	32 bits	720 MHz	**6570**	288 K–1056 K	1.0, 1.2, 1.4	$39–277	Next generation ’C6xxx architecture
	TMS320 C67x	Floating point	32 bits	32 bits	225 MHz	**1100**	64 K–264 K	1.2, 1.26, 1.8, 1.9	$14–110	Floating point version of ’C62x

Cores versus Chips



CORES

Licensor	Family	Floating, Fixed, or Both	Data Width	Instruction Width	Core Clock Speed [1]	**BDTImark2000™** *BDTIsimMark2000™* [2]	Total Core Memory Space, Bytes	Core Voltage	Process	Notes
3DSP	SP-5	Fixed point	32 bits	32 bits	225 MHz	*1720*	4 G	1.0	0.13µm	Dual-issue, superscalar SIMD DSP
ARM	ARM7	Fixed point	32 bits	16/32 bits	133 MHz	*140*	4 G	1.2	0.13µm	Widely licensed 32-bit microprocessor core
	ARM9	Fixed point	32 bits	16/32 bits	250 MHz	*310*	4 G	1.2	0.13µm	Adds separate data bus, deeper pipeline to ARM7
LSI Logic	ZSP500	Fixed point	16/32 bits	16/32 bits	325 MHz	*2570*	64 M	1.0	0.13µm	2nd-generation ZSP; 4-way superscalar
ParthusCeva [9]	PalmDSPCore	Fixed point	16/20/24 bits	16/32 bits	180 MHz	n/a	32 M	1.2	0.13µm	Selectable data width, dual-MAC, dual-issue DSP core
StarCore	SC1400	Fixed point	16 bits	16 bits	305 MHz	*3420*	4 G	1.2	0.13µm	Synthesizable version of quad-MAC, 6-issue SC140
SuperH [10]	SH-4	Both	32 bits	16 bits	266 MHz	**830**	4 G	1.2	0.13µm	Superscalar microprocessor with 3D geometry instructions
	SH-5	Fixed point	8/16/32/64 bits	16/32 bits	400 MHz	*1560*	4 G	1.2	0.13µm	Microprocessor with SIMD, optional floating-point

NVidia NV3x Video Card Core - NVIDIA GEFORCE FX 5900

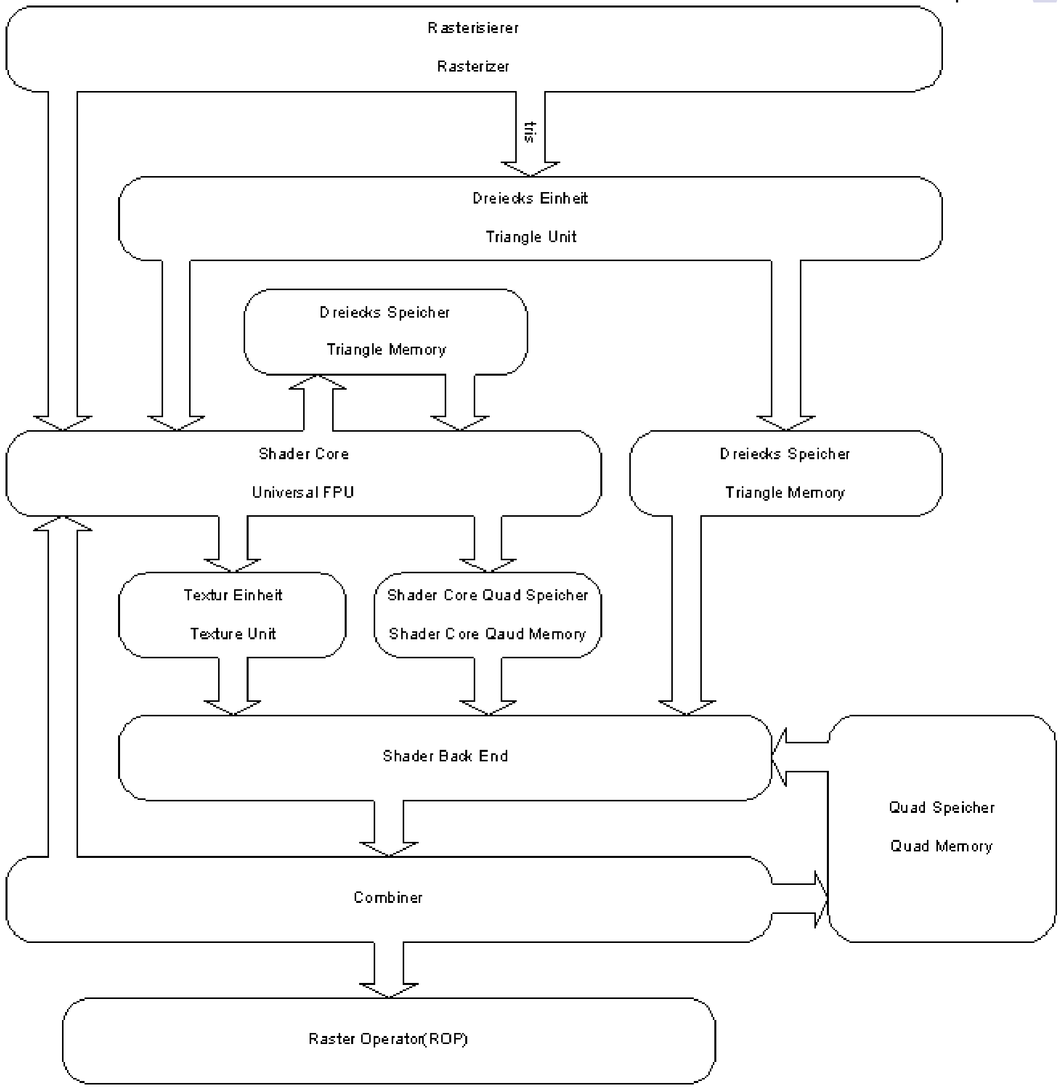
Cut input into little quads

Interpolater

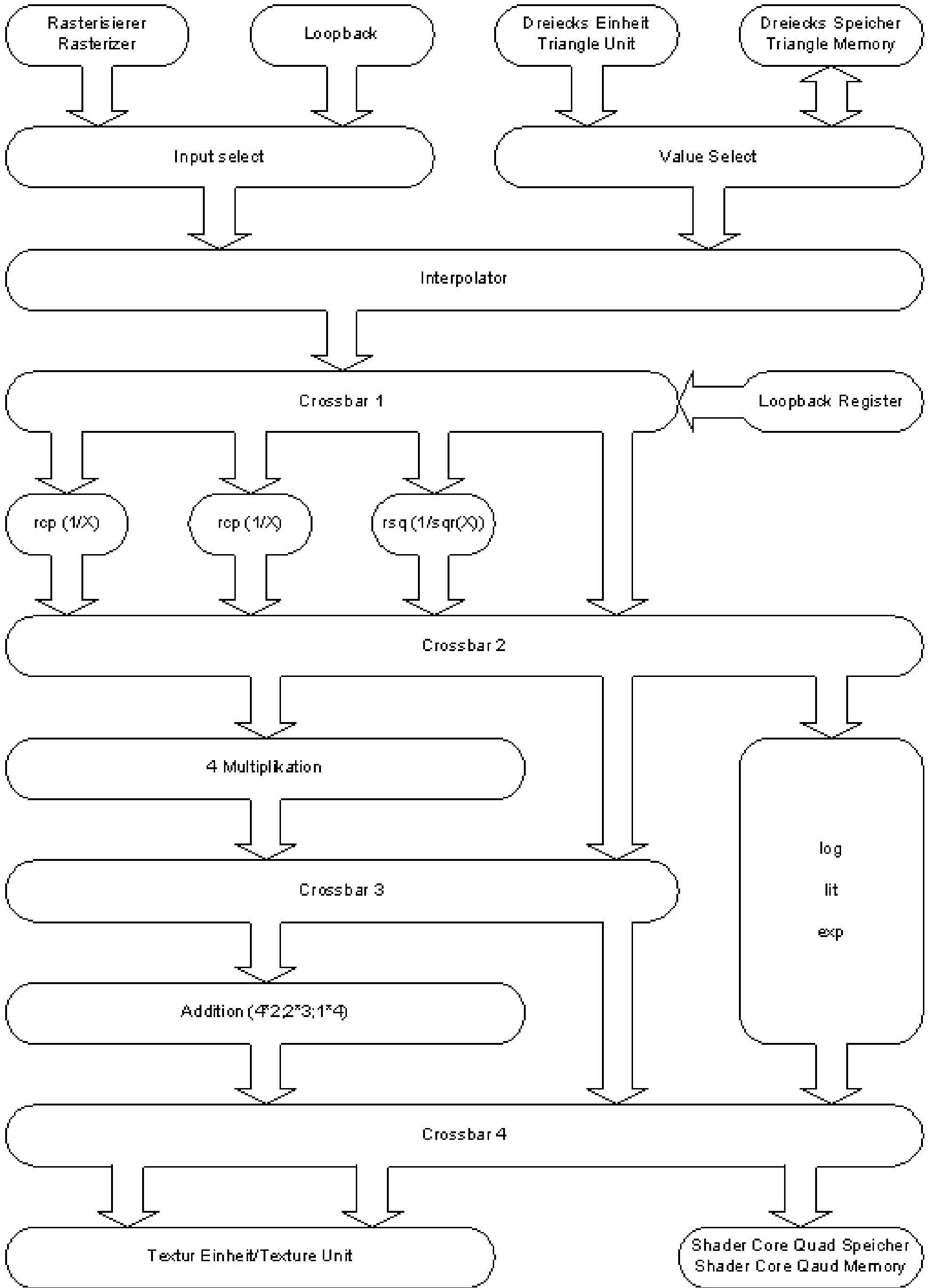
Programmable DSP Core

Different Units for different processes

Fusing and smoothing



NV3x Guts



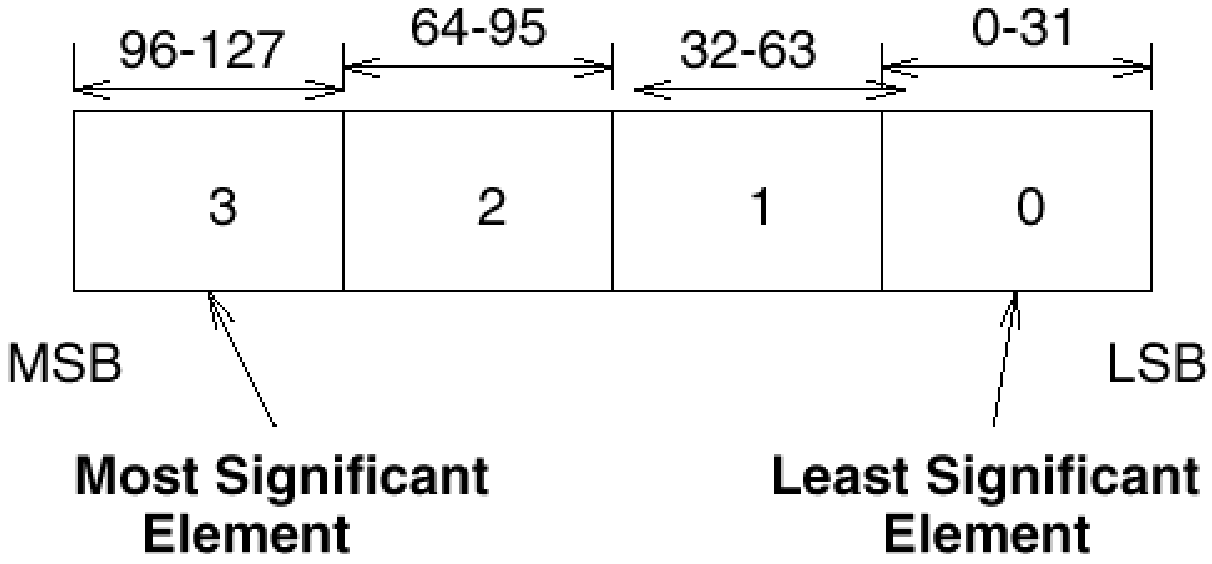
Rasterisierer Rasterizer
Loopback
Dreiecks Einheit Triangle Unit
Dreiecks Speicher Triangle Memory
Input select
Value Select
Interpolator
Crossbar 1
Loopback Register
rcp (1/X)
rcp (1/X)
rsq (1/sqr(X))
Crossbar 2
4 Multiplikation
log
lit
exp
Crossbar 3
Addition (4*2,2*3,1*4)
Crossbar 4
Textur Einheit/Texture Unit
Shader Core Quad Speicher Shader Core Qaud Memory

MMX versus SSE

- MMX: 51 New processor instructions for Pentium II
  - MMX = MultiMedia eXtensions
  - SIMD for integers
  - MMX instructions operate on two 32-bit integers simultaneously
- SSE: 70 New processor instructions and subtle architecture differences for the Pentium III and later
  - SSE = Streaming SIMD extensions
  - Pentium III introduction did not follow Moore's law on clock speed, but on most operations because of it
  - SIMD for single-precision floating-point numbers
  - SSE instructions operate on four 32-bit *floats* simultaneously.

SSE Architecture Changes

- New registers, each is 128 bits long and can hold four single-precision (32 bit) floating-point numbers



SSE Advantages

- An application cannot execute MMX instructions and perform floating-point operations simultaneously.
- Operations accelerated with SSE instructions are matrix multiplication, matrix transposition, matrix-matrix operations like addition, subtraction, and multiplication, matrix-vector multiplication, vector normalization, vector dot product, and lighting calculations.

MMX Benchmark

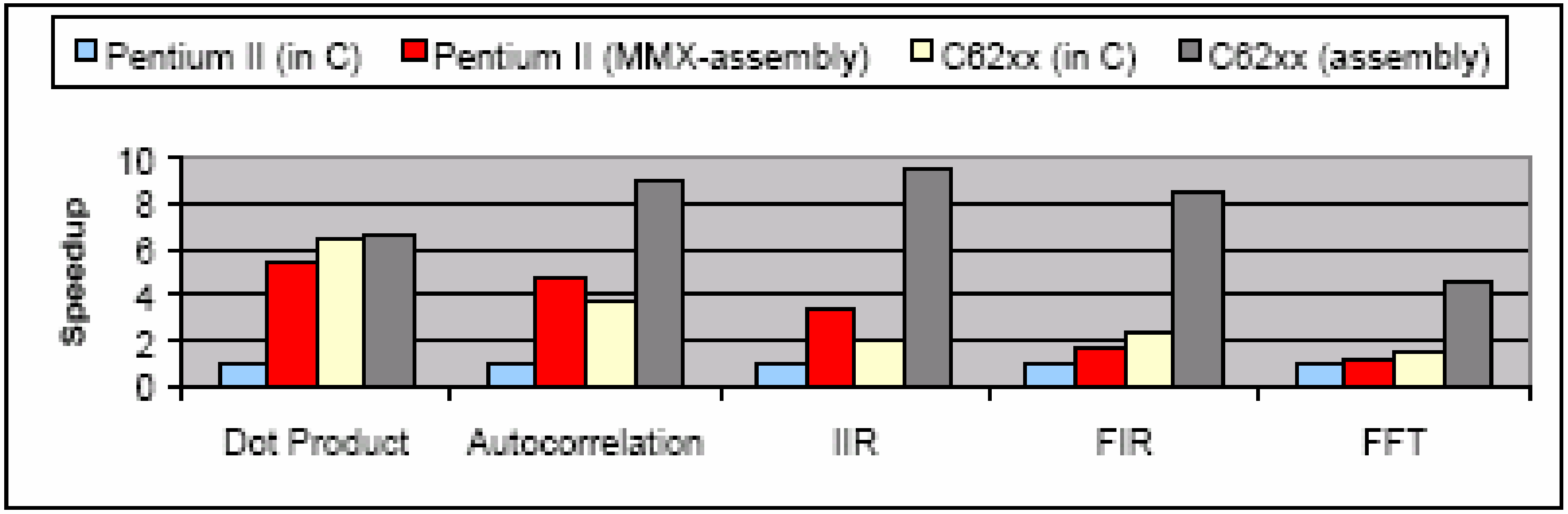
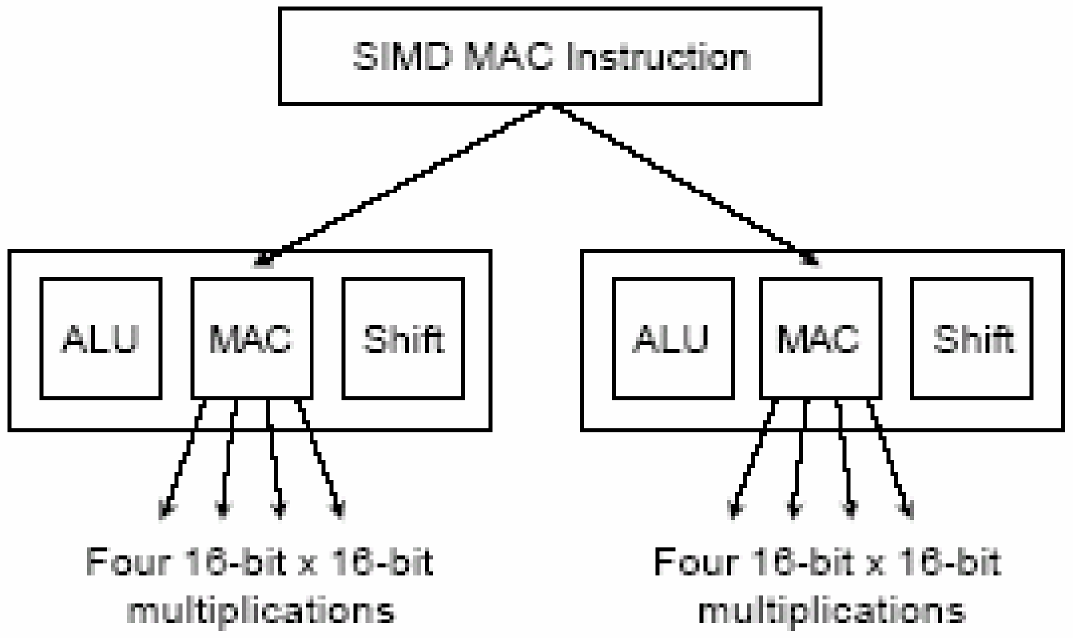


Figure. 1. Comparison of the DSP versus the general-purpose processor

Deependra Talla and Lizy K. John (1999) “Performance Evaluation and Benchmarking of Native Signal Processing” European Conference on Parallel Processing

ADSP-TS20x TigerSHARC


SIMD MAC Instruction
ALU
MAC
Shift
ALU
MAC
Shift
Four 16-bit x 16-bit multiplications
Four 16-bit x 16-bit multiplications

VLIW and SIMD:

Split one instruction between two units (VLIW), and each of those units can split their part of the instruction into sub units.

In this example we can see one uber-instruction can call 8 16-bit multiplies.

*** Walkthrough of ADSP-TS201 Spec Sheet**

Motorola DSP56367

- Walkthrough of SPECSHEET

Texas Instruments TMS320VC5421

- Spec Sheet Walkthrough