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/**
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)**

<table>
<thead>
<tr>
<th>Fetch</th>
<th>Decode</th>
<th>Read</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Pipelined (Most conventional DSP processors)

<table>
<thead>
<tr>
<th>Fetch</th>
<th>Decode</th>
<th>Read</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Superscalar (Pentium, MIPS)

<table>
<thead>
<tr>
<th>Fetch</th>
<th>Decode</th>
<th>Read</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

Data-shifting

<table>
<thead>
<tr>
<th>Time</th>
<th>Buffer contents</th>
<th>Next sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>n=N</td>
<td>$x_{N-K+1}$</td>
<td>$x_{N-K+1}$</td>
</tr>
<tr>
<td>n=N+1</td>
<td>$x_{N-K+2}$</td>
<td>$x_{N-K+3}$</td>
</tr>
<tr>
<td>n=N+2</td>
<td>$x_{N-K+3}$</td>
<td>$x_{N-K+4}$</td>
</tr>
</tbody>
</table>
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:
      - $10^{(120/20)}/(2^{16}) = 15.25$
      - $10^{(120/20)}/(2^{24}) = .0595$
“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

<table>
<thead>
<tr>
<th>Metric</th>
<th>TI TMS320C6202</th>
<th>TI TMS320C549</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>MHz</td>
<td>250</td>
<td>120</td>
<td>(\approx 2 : 1)</td>
</tr>
<tr>
<td>MIPS</td>
<td>2000</td>
<td>120</td>
<td>(\approx 17 : 1)</td>
</tr>
<tr>
<td>MMACS</td>
<td>500</td>
<td>120</td>
<td>(\approx 4 : 1)</td>
</tr>
<tr>
<td># of Pins</td>
<td>384</td>
<td>144</td>
<td>(\approx 2.7 : 1)</td>
</tr>
</tbody>
</table>

* [www.bdti.com](http://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 than 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, i.e., Interface for an unconnected protocol
  - Event activation with different latencies, i.e., 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.
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
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
        movep y:input,y:(r4) // move peripheral data from Input "y"
        clr a x:(r0)+,x0 y:(r4)-,y0 // clear accumulator, memory moves
        rep N-1 // Repeat next instruction
        mac x0,y0,a x:(r0)+,x0 y:(r4)-,y0 // Multiply Accumulate, update registers
        macr c0,y0,a (r4)+ // Rounding and scaling (set by c0)
        movep a,y:output // move accumulator output to peripheral "y"
    }
    // End assembler call

    /* Control logic system setup and whatnot
    ............................................ */
}
Differences in Assembler codes

main:

```assembly
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:

```assembly
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:

```assembly
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:

```assembly
nop
add r9, -1
bnz per_sample	nop
```

filter_done:      /* Set an SDEBUG break-point here */
```assembly
nop /* Note: ZSIM or RTL need a HALT here */
nop
br filter_done
```

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

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Family</th>
<th>Floating, Fixed, or Both</th>
<th>Data Width</th>
<th>Instruction Width</th>
<th>Core Clock Speed [1]</th>
<th>Core Voltage</th>
<th>Unit Price [3]</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADSP-218x</td>
<td>Fixed point</td>
<td>16 bits</td>
<td>24 bits</td>
<td>80 MHz</td>
<td>240</td>
<td>1.8</td>
<td>$4–24</td>
<td>Many family members w/ assorted peripherals</td>
</tr>
<tr>
<td>ADSP-219x</td>
<td>Fixed point</td>
<td>16 bits</td>
<td>24 bits</td>
<td>160 MHz</td>
<td>410</td>
<td>2.5</td>
<td>$10–24</td>
<td>Enhanced version of the ADSP-218x</td>
</tr>
<tr>
<td>ADSP-2116x</td>
<td>Floating point</td>
<td>32/40 bits</td>
<td>48 bits</td>
<td>100 MHz</td>
<td>470</td>
<td>1.8, 2.5</td>
<td>$22–99</td>
<td>Features SIMD, strong multiprocess or support</td>
</tr>
<tr>
<td>ADSP-BF53x</td>
<td>Fixed point</td>
<td>16 bits</td>
<td>16/32 bits</td>
<td>600 MHz</td>
<td>3360 [5]</td>
<td>0.7–1.2, 1.0–1.6</td>
<td>$6–35</td>
<td>Dual-MAC DSP with variable speed and voltage</td>
</tr>
<tr>
<td>ADSP-TS20x</td>
<td>Both</td>
<td>8/16/32/4 0 bits</td>
<td>32 bits</td>
<td>600 MHz</td>
<td>6150 [5]</td>
<td>1.0, 1.2</td>
<td>$35–299</td>
<td>4-way VLIW with SIMD capabilities; uses eDRAM</td>
</tr>
</tbody>
</table>

* From http://www.bdti.com
Motorola Devices Overview

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Family</th>
<th>Floating, Fixed, or Both</th>
<th>Data Width</th>
<th>Instruction Width</th>
<th>Core Clock Speed</th>
<th>BDTImark2 000™</th>
<th>BDTIsimMarkk2000™</th>
<th>Total On-Chip Memory, Bytes</th>
<th>Core Voltage</th>
<th>Unit Price</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motorola</td>
<td>DSP56 3xx</td>
<td>Fixed point</td>
<td>24 bits</td>
<td>24 bits</td>
<td>240 MHz</td>
<td>710</td>
<td>24 K–384 K</td>
<td>1.5, 1.6, 1.8, 3.3</td>
<td>$4–56</td>
<td>PCI bus, DMA, can run '560xx code unmodified</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSP56 8xx</td>
<td>Fixed point</td>
<td>16 bits</td>
<td>16 bits</td>
<td>40 MHz</td>
<td>110</td>
<td>28 K–152 K</td>
<td>2.5, 3.3</td>
<td>$3–15</td>
<td>Contains many microcontroller-like features</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSP56 85x</td>
<td>Fixed point</td>
<td>16 bits</td>
<td>16 bits</td>
<td>120 MHz</td>
<td>340</td>
<td>36 M</td>
<td>1.8</td>
<td>$6–12</td>
<td>Enhanced version of the '568xx</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MSC81 0x (SC140)</td>
<td>Fixed point</td>
<td>16 bits</td>
<td>16 bits</td>
<td>300 MHz</td>
<td>3370 [7]</td>
<td>512 K–1436 K</td>
<td>1.6</td>
<td>$90–195</td>
<td>Based on quad-MAC SC140 core; '8102 uses 4 cores</td>
<td></td>
</tr>
</tbody>
</table>

* From http://www.bdti.com
# TI Devices Overview

*From [http://www.bdti.com](http://www.bdti.com)*

<table>
<thead>
<tr>
<th>CHIPS</th>
<th>Vendor</th>
<th>Family</th>
<th>Floating, Fixed, or Both</th>
<th>Data Width</th>
<th>Instruction Width</th>
<th>Core Clock Speed</th>
<th>BDTImark2 000™ BDTIsimMar k2000™</th>
<th>Total On-Chip Memory, Bytes</th>
<th>Core Voltage</th>
<th>Unit Price</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TI</td>
<td>TMS320 F24x</td>
<td>Fixed point</td>
<td>16 bits</td>
<td>16/32 bits</td>
<td>40 MHz</td>
<td>n/a</td>
<td>18 K–1120 K</td>
<td>3.3, 5.0</td>
<td>$3–15</td>
<td>Hybrid microcontroller/DSP</td>
<td></td>
</tr>
<tr>
<td>TI</td>
<td>TMS320 F28x</td>
<td>Fixed point</td>
<td>32 bits</td>
<td>16/32 bits</td>
<td>150 MHz</td>
<td>n/a</td>
<td>164 K–292 K</td>
<td>1.8</td>
<td>$16–18</td>
<td>Hybrid microcontroller/DSP; compatible w/ 'C24x</td>
<td></td>
</tr>
<tr>
<td>TI</td>
<td>TMS320 C3x</td>
<td>Floating point</td>
<td>32 bits</td>
<td>32 bits</td>
<td>75 MHz</td>
<td>n/a</td>
<td>264 K–2304 K</td>
<td>3.3, 5.0</td>
<td>$10–213</td>
<td>Cost-competitive with fixed point DSPs</td>
<td></td>
</tr>
<tr>
<td>TI</td>
<td>TMS320 C54x</td>
<td>Fixed point</td>
<td>16 bits</td>
<td>16 bits</td>
<td>160 MHz</td>
<td>500</td>
<td>24 K–1280 K</td>
<td>1.5, 1.6, 1.8, 2.5, 3.3</td>
<td>$4–109</td>
<td></td>
<td>Many specialized instructions</td>
</tr>
<tr>
<td>TI</td>
<td>TMS320 C55x</td>
<td>Fixed point</td>
<td>16 bits</td>
<td>8–48 bits</td>
<td>300 MHz</td>
<td>1460</td>
<td>80 K–376 K</td>
<td>1.26, 1.5, 1.6</td>
<td>$5–20</td>
<td>Next generation 'C6xx architecture; dual-issue, dual-MAC DSP</td>
<td></td>
</tr>
<tr>
<td>TI</td>
<td>TMS320 C62x</td>
<td>Fixed point</td>
<td>16 bits</td>
<td>32 bits</td>
<td>300 MHz</td>
<td>1920</td>
<td>72 K–896 K</td>
<td>1.5, 1.8</td>
<td>$9–102</td>
<td>8-way VLIW</td>
<td></td>
</tr>
<tr>
<td>TI</td>
<td>TMS320 C64x</td>
<td>Fixed point</td>
<td>8/16 bits</td>
<td>32 bits</td>
<td>720 MHz</td>
<td>6570</td>
<td>288 K–1056 K</td>
<td>1.0, 1.2, 1.4</td>
<td>$39–277</td>
<td>Next generation 'C6xxx architecture</td>
<td></td>
</tr>
<tr>
<td>TI</td>
<td>TMS320 C67x</td>
<td>Floating point</td>
<td>32 bits</td>
<td>32 bits</td>
<td>225 MHz</td>
<td>1100</td>
<td>64 K–264 K</td>
<td>1.2, 1.26, 1.8, 1.9</td>
<td>$14–110</td>
<td>Floating point version of 'C62x</td>
<td></td>
</tr>
</tbody>
</table>
# Cores versus Chips

<table>
<thead>
<tr>
<th>Licensor</th>
<th>Family</th>
<th>Floating, Fixed, or Both</th>
<th>Data Width</th>
<th>Instruction Width</th>
<th>Core Clock Speed</th>
<th>BDTI\text{mark}2000™ BDTI\text{simMark}2000™</th>
<th>Total Core Memory Space, Bytes</th>
<th>Core Voltage</th>
<th>Process</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>3DSP</td>
<td></td>
<td>Fixed point</td>
<td>32 bits</td>
<td>32 bits</td>
<td>225 MHz</td>
<td>1720</td>
<td>4 G</td>
<td>1.0</td>
<td>0.13µm</td>
<td>Dual-issue, superscalar SIMD DSP</td>
</tr>
<tr>
<td>ARM</td>
<td>ARM7</td>
<td>Fixed point</td>
<td>32 bits</td>
<td>16/32 bits</td>
<td>133 MHz</td>
<td>140</td>
<td>4 G</td>
<td>1.2</td>
<td>0.13µm</td>
<td>Widely licensed 32-bit microprocessor core</td>
</tr>
<tr>
<td>ARM</td>
<td>ARM9</td>
<td>Fixed point</td>
<td>32 bits</td>
<td>16/32 bits</td>
<td>250 MHz</td>
<td>310</td>
<td>4 G</td>
<td>1.2</td>
<td>0.13µm</td>
<td>Adds separate data bus, deeper pipeline to ARM7</td>
</tr>
<tr>
<td>LSI Logic</td>
<td>ZSP500</td>
<td>Fixed point</td>
<td>16/32 bits</td>
<td>16/32 bits</td>
<td>325 MHz</td>
<td>2570</td>
<td>64 M</td>
<td>1.0</td>
<td>0.13µm</td>
<td>2nd-generation ZSP, 4-way superscalar</td>
</tr>
<tr>
<td>ParthusCeva</td>
<td>PalMDSPCore</td>
<td>Fixed point</td>
<td>16/20/24 bits</td>
<td>16/32 bits</td>
<td>180 MHz</td>
<td>n/a</td>
<td>32 M</td>
<td>1.2</td>
<td>0.13µm</td>
<td>Selectable data width, dual-MAC, dual-issue DSP core</td>
</tr>
<tr>
<td>StarCore</td>
<td>SC1400</td>
<td>Fixed point</td>
<td>16 bits</td>
<td>16 bits</td>
<td>305 MHz</td>
<td>3420</td>
<td>4 G</td>
<td>1.2</td>
<td>0.13µm</td>
<td>Synthesizable version of quad-MAC, 6-issue SC140</td>
</tr>
<tr>
<td>SuperH</td>
<td>SH-4</td>
<td>Both</td>
<td>32 bits</td>
<td>16 bits</td>
<td>266 MHz</td>
<td>830</td>
<td>4 G</td>
<td>1.2</td>
<td>0.13µm</td>
<td>Superscalar microprocessor with 3D geometry instructions</td>
</tr>
<tr>
<td>SuperH</td>
<td>SH-5</td>
<td>Fixed point</td>
<td>8/16/32/64 bits</td>
<td>16/32 bits</td>
<td>400 MHz</td>
<td>1560</td>
<td>4 G</td>
<td>1.2</td>
<td>0.13µm</td>
<td>Microprocessor with SIMD, optional floating-point</td>
</tr>
</tbody>
</table>
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
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

ADSP-TS20x TigerSHARC

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