DSP Processor Architecture

- Classification of Processor Applications
- Requirements of Embedded Processors
- DSP vs. General Purpose CPUs
- DSP Algorithm Format
- Classification of DSP Applications
- DSP Benchmarks
- Basic Architectural Features of DSPs
- DSP Software Development Considerations
- DSP Cores vs. Chips
- Classification of Current DSP Architectures and example DSPs:
  - Conventional DSPs: TI TMSC54xx
  - Enhanced Conventional DSPs: TI TMSC55xx
  - VLIW DSPs: TI TMS320C62xx, TMS320C64xx
  - Superscalar DSPs: LSI Logic ZSP400 DSP core
Processor Applications

- General Purpose - high performance
  - Alpha’s, SPARC, MIPS ..
  - Used for general purpose software
  - Heavy weight OS - UNIX, NT
  - Workstations, PC’s

- Embedded processors and processor cores
  - ARM, 486SX, Hitachi SH7000, NEC V800
  - Often require Digital signal processing (DSP) support.
  - Single program
  - Lightweight, often realtime OS
  - Cellular phones, consumer electronics .. (e.g. CD players)

- Microcontrollers
  - Extremely cost sensitive
  - Small word size - 8 bit common
  - Highest volume processors by far
  - Automobiles, toasters, thermostats, ...
Processor Markets

- 32-bit micro: $30B
  - 32-bit DSP: $5.2B/17%
  - DSP: $10B/33%
  - 16-bit micro: $5.7B/19%
  - 8-bit micro: $9.3B/31%

Total: $30B

- 8-bit micro: $1.2B/4%
- Total: $31.2B/100%
The Processor Design Space

Cost

Performance

Application specific architectures for performance

Embedded processors

Microprocessors

Performance is everything & Software rules

Cost is everything

Microcontrollers

EECC722 - Shaaban

#4 lec # 8 Fall 2002 10-7-2002
Requirements of Embedded Processors

- Optimized for a single program - code often in on-chip ROM or off chip EPROM
- Minimum code size (one of the motivations initially for Java)
- Performance obtained by optimizing datapath
- Low cost
  - Lowest possible area
  - Technology behind the leading edge
  - High level of integration of peripherals (reduces system cost)
- Fast time to market
  - Compatible architectures (e.g. ARM) allows reusable code
  - Customizable core
- Low power if application requires portability
Area of processor cores = Cost

![Bar chart showing the comparison of core area in mm² for different processor cores, including the Nintendo processor and cellular phones.](image)
Another figure of merit: Computation per unit area

- Nintendo processor
- Cellular phones

<table>
<thead>
<tr>
<th>Processor</th>
<th>MIPS/mm²</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel 386</td>
<td>0.16</td>
</tr>
<tr>
<td>Motorola CPU32+</td>
<td>0.53</td>
</tr>
<tr>
<td>NEC V810</td>
<td>1.41</td>
</tr>
<tr>
<td>LSI R3000</td>
<td>1.67</td>
</tr>
<tr>
<td>ARM6</td>
<td>2.0</td>
</tr>
<tr>
<td>Hitachi SH-2</td>
<td>2.4</td>
</tr>
<tr>
<td>ARM7</td>
<td>3.75</td>
</tr>
<tr>
<td>Piranha-32</td>
<td>4.6</td>
</tr>
<tr>
<td>Piranha-16</td>
<td>5.23</td>
</tr>
</tbody>
</table>
- If a majority of the chip is the program stored in ROM, then code size is a critical issue.
- The Piranha has 3 sized instructions - basic 2 byte, and 2 byte plus 16 or 32 bit immediate.
## Embedded Systems vs. General Purpose Computing

<table>
<thead>
<tr>
<th>Embedded System</th>
<th>General purpose computing</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Runs a few applications often known at design time</td>
<td>• Intended to run a fully general set of applications</td>
</tr>
<tr>
<td>• Not end-user programmable</td>
<td>• End-user programmable</td>
</tr>
<tr>
<td>• Operates in fixed run-time constraints that must be met, additional performance may not be useful/valuable</td>
<td>• Faster is always better</td>
</tr>
<tr>
<td>• Differentiating features:</td>
<td>• Differentiating features</td>
</tr>
<tr>
<td>– Application-specific capability (e.g. DSP).</td>
<td>– speed (need not be fully predictable)</td>
</tr>
<tr>
<td>– power</td>
<td>– cost (largest component power)</td>
</tr>
<tr>
<td>– cost</td>
<td></td>
</tr>
<tr>
<td>– speed (must be predictable)</td>
<td></td>
</tr>
</tbody>
</table>
Evolution of GP and DSP

• General Purpose Microprocessor traces roots back to Eckert, Mauchly, Von Neumann (ENIAC)

• DSP processors are microprocessors designed for efficient mathematical manipulation of digital signals.
  – DSP evolved from Analog Signal Processors (ASPs), using analog hardware to transform physical signals (classical electrical engineering)
  – ASP to DSP because
    • DSP insensitive to environment (e.g., same response in snow or desert if it works at all)
    • DSP performance identical even with variations in components; 2 analog systems behavior varies even if built with same components with 1% variation

• Different history and different applications led to different terms, different metrics, some new inventions.

• Convergence of markets will lead to architectural showdown.
DSP vs. General Purpose CPUs

- DSPs tend to run one program, not many programs.
  - Hence OSes are much simpler, there is no virtual memory or protection, ...
- DSPs usually run applications with hard real-time constraints:
  - You must account for anything that could happen in a time slot
  - All possible interrupts or exceptions must be accounted for and their collective time be subtracted from the time interval.
  - Therefore, exceptions are BAD.
- DSPs usually process infinite continuous data streams.
- The design of DSP architectures and ISAs driven by the requirements of DSP algorithms.
DSP vs. General Purpose MPU

• The “MIPS/MFLOPS” of DSPs is speed of Multiply-Accumulate (MAC).
  – MAC is common in DSP algorithms that involve computing a vector dot product, such as digital filters, correlation, and Fourier transforms.
  – DSP are judged by whether they can keep the multipliers busy 100% of the time and by how many MACs are performed in each cycle.

• The "SPEC" of DSPs is 4 algorithms:
  – Infinite Impulse Response (IIR) filters
  – Finite Impulse Response (FIR) filters
  – FFT, and
  – convolvers

• In DSPs, target algorithms are important:
  – Binary compatibility not a major issue

• High-level Software is not (yet) very important in DSPs.
  – People still write in assembly language for a product to minimize the die area for ROM in the DSP chip.
DSP Algorithm Format

- DSP culture has a graphical format to represent formulas.
- Like a flowchart for formulas, inner loops, not programs.
- Some seem natural:
  \[ \Sigma \] is add, \( X \) is multiply
- Others are obtuse:
  \[ z^{-1} \] means take variable from earlier iteration.
- These graphs are trivial to decode
DSP Algorithm Notation

- Uses “flowchart” notation instead of equations
- Multiply is or
  - $X$
- Add is or
  - $+$
- Delay/Storage is or
  - $\Sigma$
  - $z^{-1}$
  - $D$
Sample DSP Algorithm:
Finite-impulse Response (FIR) Filter

- M most recent samples in the delay line (Xi)
- New sample moves data down delay line
- “Tap” is a multiply-add
- Each tap (M+1 taps total) nominally requires:
  - Two data fetches
  - Multiply
  - Accumulate
  - Memory write-back to update delay line
- Goal: at least 1 FIR Tap / DSP instruction cycle
FINITE-IMPULSE RESPONSE (FIR) FILTER

\[ Z^{-1} \quad Z^{-1} \quad \ldots \quad Z^{-1} \]

\[ C_1 \quad C_2 \quad \ldots \quad C_{N-1} \quad C_N \]
FIR filter on (simple) General Purpose Processor

loop:
  lw  x0, 0(r0)
  lw  y0, 0(r1)
  mul a, x0,y0
  add y0,a,b
  sw  y0,(r2)
  inc r0
  inc r1
  inc r2
  dec ctr
  tst ctr
  jnz loop

• Problems: Bus / memory bandwidth bottleneck, control code overhead
DSP Applications

- Digital audio applications
  - MPEG Audio
  - Portable audio
- Digital cameras
- Cellular telephones
- Wearable medical appliances
- Storage products:
  - disk drive servo control
- Military applications:
  - radar
  - sonar

  - Industrial control
  - Seismic exploration
  - Networking:
    - Wireless
    - Base station
    - Cable modems
    - ADSL
    - VDSL
Another Look at DSP Applications

- High-end
  - Military applications
  - Wireless Base Station - TMS320C6000
  - Cable modem
  - Gateways
- Mid-end
  - Industrial control
  - Cellular phone - TMS320C540
  - Fax/voice server
- Low end
  - Storage products - TMS320C27
  - Digital camera - TMS320C5000
  - Portable phones
  - Wireless headsets
  - Consumer audio
  - Automobiles, toasters, thermostats, ...
DSP range of applications

- **Carrier Class/Enterprise**
  - Remote access servers
  - Basetations
  - VOP gateways + modem
  - CO switches

- **Mid-Range Telecom**
  - Fax/Voice servers
  - PBX add-ons
  - Voice-over-packet
  - Voice add-ons to LAN
  - SOHO voice + data systems

- **Portable/Consumer**
  - Solid state audio
  - G.Lite or wireless modems
  - Digital radios/phones

- **Client-Side Telephony**
  - Feature phones/web phones
  - POS, metering, pay phones
  - Speaker phones, security

- **Ultra-Low Power**
  - Biometric, personal medical
  - Wireless headsets
DSP is the fastest growing segment of the semiconductor market

Mixed/Signal Analog

DSP

Market for DSP Products

DSP is the fastest growing segment of the semiconductor market
HW/SW/IC PARTITIONING

PHYSICAL LAYER PROCESSING

CONTROLLER

BASEBAND CONVERTER

RF MODEM

A/D

SPEECH ENCODE

SPEECH DECODE

DAC

DSP

ANALOG IC

MICROCONTROLLER

415-555-1212
Mapping Onto A System-on-a-chip

- RAM
- S/P
- DMA
- µC
- DSP Core
- ASIC Logic
- S/P
- DMA

- S/P
- DMA
- phone book
- keypad intfc
- control
- protocol
- speech quality enhancement
- voice recognition
- de-intl & decoder
- RPE-LTP speech decoder
- demodulator and synchronizer
- Viterbi equalizer

EECC722 - Shaaban
#24 lec # 8 Fall 2002 10-7-2002
Example Wireless Phone Organization

Key Advantages
- Flexible
- Reusable

---

EECC722 - Shaaban

#25 lec # 8 Fall 2002 10-7-2002
Multimedia I/O Architecture

- Radio Modem
- Embedded Processor
- Sched ECC Pact
- Interface
- Low Power Bus
- FB
- Fifo
- SRAM
- Graphics
- Pen
- Audio
- Video Decomp
- Video

Data Flow
Future chips will be a mix of processors, memory and dedicated hardware for specific algorithms and I/O
## DSP ARCHITECTURE

### Enabling Technologies

<table>
<thead>
<tr>
<th>Time Frame</th>
<th>Approach</th>
<th>Primary Application</th>
<th>Enabling Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Early 1970’s</td>
<td>Discrete logic</td>
<td>Non-real time processing, Simulation, Military radars, Digital Comm.</td>
<td>Bipolar SSI, MSI, FFT algorithm</td>
</tr>
<tr>
<td>Late 1970’s</td>
<td>Building block</td>
<td></td>
<td>Single chip bipolar multiplier, Flash A/D</td>
</tr>
<tr>
<td>Early 1980’s</td>
<td>Single Chip DSP μP</td>
<td>Telecom, Control</td>
<td>μP architectures, NMOS/CMOS</td>
</tr>
<tr>
<td>Late 1980’s</td>
<td>Function/Application specific chips</td>
<td>Computers, Communication</td>
<td>Vector processing, Parallel processing</td>
</tr>
<tr>
<td>Early 1990’s</td>
<td>Multiprocessing</td>
<td>Video/Image Processing</td>
<td>Advanced multiprocessing, VLIW, MIMD, etc.</td>
</tr>
<tr>
<td>Late 1990’s</td>
<td>Single-chip multiprocessing</td>
<td>Wireless telephony, Internet related</td>
<td>Low power single-chip DSP, Multiprocessing</td>
</tr>
</tbody>
</table>
Texas Instruments TMS320 Family
Multiple DSP µP Generations

<table>
<thead>
<tr>
<th></th>
<th>First Sample</th>
<th>Bit Size</th>
<th>Clock speed (MHz)</th>
<th>Instruction Throughput</th>
<th>MAC execution (ns)</th>
<th>MOPS</th>
<th>Device density (# of transistors)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Uniprocessor</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Based</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>(Harvard Architecture)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMS32010</td>
<td>1982</td>
<td>16 integer</td>
<td>20</td>
<td>5 MIPS</td>
<td>400</td>
<td>5</td>
<td>58,000 (3µ)</td>
</tr>
<tr>
<td>TMS320C25</td>
<td>1985</td>
<td>16 integer</td>
<td>40</td>
<td>10 MIPS</td>
<td>100</td>
<td>20</td>
<td>160,000 (2µ)</td>
</tr>
<tr>
<td>TMS320C30</td>
<td>1988</td>
<td>32 flt.pt.</td>
<td>33</td>
<td>17 MIPS</td>
<td>60</td>
<td>33</td>
<td>695,000 (1µ)</td>
</tr>
<tr>
<td>TMS320C50</td>
<td>1991</td>
<td>16 integer</td>
<td>57</td>
<td>29 MIPS</td>
<td>35</td>
<td>60</td>
<td>1,000,000 (0.5µ)</td>
</tr>
<tr>
<td>TMS320C2XXX</td>
<td>1995</td>
<td>16 integer</td>
<td>40</td>
<td>40 MIPS</td>
<td>25</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td><strong>Multiprocessor</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Based</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMS320C80</td>
<td>1996</td>
<td>32 integer/flt.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2 GOPS MIMD</td>
</tr>
<tr>
<td>TMS320C62XX</td>
<td>1997</td>
<td>16 integer</td>
<td>1600 MIPS</td>
<td>5</td>
<td></td>
<td>20 GOPS VLIW</td>
<td></td>
</tr>
<tr>
<td>TMS310C67XX</td>
<td>1997</td>
<td>32 flt. pt.</td>
<td>5</td>
<td></td>
<td></td>
<td>1 GFLOP VLIW</td>
<td></td>
</tr>
</tbody>
</table>
TYPES OF DSP PROCESSORS

• **32-BIT FLOATING POINT** (5% of market):
  – TI TMS320C3X, TMS320C67xx
  – AT&T DSP32C
  – ANALOG DEVICES ADSP21xxx
  – Hitachi SH-4

• **16-BIT FIXED POINT** (95% of market):
  – TI TMS320C2X, TMS320C62xx
  – Infineon TC1xxx (TriCore1)
  – MOTOROLA DSP568xx, MSC810x
  – ANALOG DEVICES ADSP21xx
  – Agere Systems DSP16xxx, Starpro2000
  – LSI Logic LSI140x (ZPS400)
  – Hitachi SH3-DSP
  – StarCore SC110, SC140
DSP BENCHMARKS

• **DSPstone**: University of Aachen, application benchmarks
  - ADPCM TRANSCODER - CCITT G.721, REAL_UPDATE, COMPLEX_UPDATES
  - DOT_PRODUCT, MATRIX_1X3, CONVOLUTION
  - FIR, FIR2DIM, HR_ONE_BIQUAD
  - LMS, FFT_INPUT_SCALED

• **BDTImark2000**: Berkeley Design Technology Inc
  - 12 DSP kernels in hand-optimized assembly language
  - Returns single number (higher means faster) per processor
  - Use only on-chip memory (memory bandwidth is the major bottleneck in performance of embedded applications).

• **EEMBC (pronounced “embassy”)**: EDN Embedded Microprocessor Benchmark Consortium
  - 30 companies formed by Electronic Data News (EDN)
  - Benchmark evaluates compiled C code on a variety of embedded processors (microcontrollers, DSPs, etc.)
  - Application domains: automotive-industrial, consumer, office automation, networking and telecommunications
Processor DSP Speed:
BDTImarks™ (Higher is Better)

<table>
<thead>
<tr>
<th>Generation</th>
<th>Year</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st gen</td>
<td>1982</td>
<td>0.5</td>
</tr>
<tr>
<td>2nd gen</td>
<td>1987</td>
<td>4</td>
</tr>
<tr>
<td>3rd gen</td>
<td>1995</td>
<td>13</td>
</tr>
<tr>
<td>4th gen</td>
<td>2000</td>
<td>148</td>
</tr>
</tbody>
</table>

- TMS32010 5 MHz
- DSP56001 13 MHz
- TMS320C54x 50 MHz
- TMS320C6203 300 MHz
Basic Architectural Features of DSPs

• Data path configured for DSP
  – Fixed-point arithmetic
  – MAC- Multiply-accumulate
• Multiple memory banks and buses -
  – Harvard Architecture
  – Multiple data memories
• Specialized addressing modes
  – Bit-reversed addressing
  – Circular buffers
• Specialized instruction set and execution control
  – Zero-overhead loops
  – Support for MAC
• Specialized peripherals for DSP
DSP Data Path: Arithmetic

- DSPs dealing with numbers representing real world
  => Want “reals”/ fractions
- DSPs dealing with numbers for addresses
  => Want integers
- Support “fixed point” as well as integers

\[
S \cdot \begin{array}{c}
\text{radix point}
\end{array} -1 \leq x < 1
\]

\[
S \cdot \begin{array}{c}
\text{radix point}
\end{array} -2^{N-1} \leq x < 2^{N-1}
\]
DSP Data Path: Precision

• Word size affects precision of fixed point numbers
• DSPs have 16-bit, 20-bit, or 24-bit data words
• Floating Point DSPs cost 2X - 4X vs. fixed point, slower than fixed point
• DSP programmers will scale values inside code
  – SW Libraries
  – Separate explicit exponent
• “Blocked Floating Point” single exponent for a group of fractions
• Floating point support simplify development
DSP Data Path: Overflow

- DSP are descended from analog:
  - Modulo Arithmetic.
- Set to most positive ($2^{N-1}-1$) or most negative value ($-2^{N-1}$): “saturation”
- Many DSP algorithms were developed in this model.
DSP Data Path: Multiplier

- Specialized hardware performs all key arithmetic operations in 1 cycle
- 50% of instructions can involve multiplier => single cycle latency multiplier
- Need to perform multiply-accumulate (MAC)
- n-bit multiplier => 2n-bit product
DSP Data Path: Accumulator

- Don’t want overflow or have to scale accumulator
- Option 1: accumulator wider than product: “guard bits”
  - Motorola DSP: 24b x 24b => 48b product, 56b Accumulator
- Option 2: shift right and round product before adder
DSP Data Path: Rounding

• Even with guard bits, will need to round when store accumulator into memory
• 3 DSP standard options
• **Truncation: chop results**  
  => biases results up
• **Round to nearest:**  
  < 1/2 round down, • 1/2 round up (more positive)  
  => smaller bias
• **Convergent:**  
  < 1/2 round down, > 1/2 round up (more positive), =  
  1/2 round to make lsb a zero (+1 if 1, +0 if 0)  
  => no bias  
  IEEE 754 calls this **round to nearest even**
# Data Path Comparison

<table>
<thead>
<tr>
<th>DSP Processor</th>
<th>General-Purpose Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Specialized hardware performs all key arithmetic operations in 1 cycle.</td>
<td>• Multiplies often take &gt;1 cycle</td>
</tr>
<tr>
<td>• Hardware support for managing numeric fidelity:</td>
<td>• Shifts often take &gt;1 cycle</td>
</tr>
<tr>
<td>– Shifters</td>
<td>• Other operations (e.g., saturation, rounding) typically take multiple cycles.</td>
</tr>
<tr>
<td>– Guard bits</td>
<td></td>
</tr>
<tr>
<td>– Saturation</td>
<td></td>
</tr>
</tbody>
</table>
First Commercial DSP (1982): Texas Instruments TMS32010

- 16-bit fixed-point arithmetic
- Introduced at 5Mhz (200ns) instruction cycle.
- “Harvard architecture”
  - separate instruction, data memories
- Accumulator
- Specialized instruction set
  - Load and Accumulate
- Two-cycle (400 ns) Multiply-Accumulate (MAC) time.
First Generation DSP µP
Texas Instruments TMS32010 - 1982

Features

• 200 ns instruction cycle (5 MIPS)
• 144 words (16 bit) on-chip data RAM
• 1.5K words (16 bit) on-chip program ROM - TMS32010
• External program memory expansion to a total of 4K words at full speed
• 16-bit instruction/data word
• single cycle 32-bit ALU/accumulator
• Single cycle 16 x 16-bit multiply in 200 ns
• Two cycle MAC (5 MOPS)
• Zero to 15-bit barrel shifter
• Eight input and eight output channels
TMS32010 FIR Filter Code

• Here X4, H4, ... 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

... 

• Two instructions per tap, but requires unrolling
Micro-architectural impact - MAC

\[ y(n) = \sum_{m=0}^{N-1} h(m) x(n-m) \]

element of finite-impulse response filter computation
• The critical hardware unit in a DSP is the multiplier - much of the architecture is organized around allowing use of the multiplier on every cycle.

• This means providing two operands on every cycle, through multiple data and address busses, multiple address units and local accumulator feedback.
MAC Eg. - 320C54x DSP Functional Block Diagram

Legend:
A = Accumulator A
B = Accumulator B
C = CB Data Bus
D = DB Data Bus
E = EB Data Bus
M = MAC Unit
P = PB Program Bus
S = Barrel Shifter
T = T Register
U = ALU

System control interface
Program address generation logic (PAGEN)
Data address generation logic (DAGEN)

Memory and external interface
DMA controller
Peripherals (serial ports, HPI, etc.)

EXP encoder
T register
MUX
Multiplier (17 x 17)
Fractional
Adder(40)
ZERO SAT ROUND
DSP Memory

- FIR Tap implies multiple memory accesses
- DSPs require multiple data ports
- Some DSPs have ad hoc techniques to reduce memory bandwidth demand:
  - Instruction repeat buffer: do 1 instruction 256 times
  - Often disables interrupts, thereby increasing interrupt response time
- Some recent DSPs have instruction caches
  - Even then may allow programmer to “lock in” instructions into cache
  - Option to turn cache into fast program memory
- No DSPs have data caches.
- May have multiple data memories
Conventional ``Von Neumann’’ memory
HARVARD MEMORY ARCHITECTURE in DSP

PROGRAM MEMORY  → GLOBAL  ← Y MEMORY

P DATA  ← X MEMORY

X DATA  → Y DATA
Memory Architecture Comparison

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

General-Purpose Processor
- Von Neumann architecture
- Typically 1 access/cycle
- Use caches
Eg. TMS320C3x MEMORY BLOCK DIAGRAM - Harvard Architecture
Eg. TI 320C62x/67x DSP (1997)
DSP Addressing

• Have standard addressing modes: immediate, displacement, register indirect
• Want to keep MAC datapath busy
• Assumption: any extra instructions imply clock cycles of overhead in inner loop
  => complex addressing is good
  => don’t use datapath to calculate fancy address
• Autoincrement/Autodecrement register indirect
  – lw r1,0(r2)+ => r1 <- M[r2]; r2<-r2+1
  – Option to do it before addressing, positive or negative
DSP Addressing: FFT

• FFTs start or end with data in butterfly order
  
<table>
<thead>
<tr>
<th>Data in Bufferfly Order</th>
<th>Data after Butterfly Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (000) =&gt; 0 (000)</td>
<td></td>
</tr>
<tr>
<td>1 (001) =&gt; 4 (100)</td>
<td></td>
</tr>
<tr>
<td>2 (010) =&gt; 2 (010)</td>
<td></td>
</tr>
<tr>
<td>3 (011) =&gt; 6 (110)</td>
<td></td>
</tr>
<tr>
<td>4 (100) =&gt; 1 (001)</td>
<td></td>
</tr>
<tr>
<td>5 (101) =&gt; 5 (101)</td>
<td></td>
</tr>
<tr>
<td>6 (110) =&gt; 3 (011)</td>
<td></td>
</tr>
<tr>
<td>7 (111) =&gt; 7 (111)</td>
<td></td>
</tr>
</tbody>
</table>

• What can do to avoid overhead of address checking instructions for FFT?
  • Have an optional “bit reverse” address addressing mode for use with autoincrement addressing
  • Many DSPs have “bit reverse” addressing for radix-2 FFT
BIT REVERSED ADDRESSING

Data flow in the radix-2 decimation-in-time FFT algorithm
DSP Addressing: Buffers

- DSPs dealing with continuous I/O
- Often interact with an I/O buffer (delay lines)
- To save memory, buffers often organized as circular buffers
- What can do to avoid overhead of address checking instructions for circular buffer?
  - Option 1: Keep start register and end register per address register for use with autoincrement addressing, reset to start when reach end of buffer
  - Option 2: Keep a buffer length register, assuming buffers starts on aligned address, reset to start when reach end
- Every DSP has “modulo” or “circular” addressing
CIRCULAR BUFFERS

Instructions accommodate three elements:

- buffer address
- buffer size
- increment

Allows for cycling through:

- delay elements
- coefficients in data memory
Addressing Comparison

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
Address calculation unit for DSPs

- Supports modulo and bit reversal arithmetic
- Often duplicated to calculate multiple addresses per cycle
DSP Instructions and Execution

• May specify multiple operations in a single instruction
• Must support Multiply-Accumulate (MAC)
• Need parallel move support
• Usually have special loop support to reduce branch overhead
  – Loop an instruction or sequence
  – 0 value in register usually means loop maximum number of times
  – Must be sure if calculate loop count that 0 does not mean 0
• May have saturating shift left arithmetic
• May have conditional execution to reduce branches
ADSP 2100: ZERO-OVERHEAD LOOP

DO <addr> UNTIL condition”

Address Generation
PCS = PC + 1
if (PC = x && ! condition)
   PC = PCS
else
   PC = PC +1

• Eliminates a few instructions in loops -
• Important in loops with small bodies
Instruction Set Comparison

**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 rl
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.
Specialized DSP peripherals

Block Diagram of the AM265
(for digital answering machine application)
TI TMS320C203/LC203 BLOCK DIAGRAM
DSP Core Approach - 1995
Summary of Architectural Features of DSPs

• Data path configured for DSP
  – Fixed-point arithmetic
  – MAC- Multiply-accumulate
• Multiple memory banks and buses -
  – Harvard Architecture
  – Multiple data memories
• Specialized addressing modes
  – Bit-reversed addressing
  – Circular buffers
• Specialized instruction set and execution control
  – Zero-overhead loops
  – Support for MAC
• Specialized peripherals for DSP

• THE ULTIMATE IN BENCHMARK DRIVEN ARCHITECTURE DESIGN.
DSP Software Development Considerations

- Different from general-purpose software development:
  - Resource-hungry, complex algorithms.
  - Specialized and/or complex processor architectures.
  - Severe cost/storage limitations.
  - Hard real-time constraints.
  - Optimization is essential.
  - Increased testing challenges.

- Essential tools:
  - Assembler, linker.
  - Instruction set simulator.
  - HLL Code generation: C compiler.
  - Debugging and profiling tools.

- Increasingly important:
  - Software libraries.
  - Real-time operating systems.
DSP Cores vs. Chips

DSP are usually available as synthesizable cores or off-the-shelf chips

- **Synthesizable Cores:**
  - Map into chosen fabrication process
    - Speed, power, and size vary
  - Choice of peripherals, etc.
  - Requires extensive hardware development effort.

- **Off-the-shelf chips:**
  - Highly optimized for speed, energy efficiency, and/or cost.
  - Limited performance, integration options.
  - Tools, 3rd-party support often more mature
Classification of Current DSP Architectures

• Modern Conventional DSPs:
  – Similar to the original DSPs of the early 1980s
  – Single instruction/cycle. Example: TI TMS320C54x

• Enhanced Conventional DSPs:
  – Add parallel execution units: SIMD operation
  – Complex, compound instructions. Example: TI TMS320C55x

• Multiple-Issue DSPs:
  – VLIW Example: TI TMS320C62xx, TMS320C64xx
  – Superscalar, Example: LSI Logic ZPS400
A Conventional DSP:
TI TMSC54xx

- 16-bit fixed-point DSP.
- Issues one 16-bit instruction/cycle
- Modified Harvard memory architecture
- Peripherals typical of conventional DSPs:
  - 2-3 synch. Serial ports, parallel port
  - Bit I/O, Timer, DMA
- Inexpensive (100 MHz ~$5 qty 10K).
- Low power (60 mW @ 1.8V, 100 MHz).
A Current Conventional DSP:
TI TMSC54xx

- Memory
  - Prog/Data ROM
  - Prog/Data SARAM
  - Prog/Data DARAM
  - Prog. Ctrl. Unit

- Instruction Bus (1 x 16 bits)
- Data Buses (2 x 16 bits read, 1 x 16 bits write)
- Address Buses (4 x 16 bits)

- Data Path
  - MAC
  - ALU
  - Shifter

- Addr. Gen.
  - Addr./Data Registers
  - Addr. Units (2)

- Data (16-bit)
- Addr. (16-bit to 23-bit)
An Enhanced Conventional DSP:
TI TMSC55xx

- The TMS320C55xx is based on Texas Instruments' earlier TMS320C54xx family, but adds significant enhancements to the architecture and instruction set, including:
  - Two instructions/cycle
    - Instructions are scheduled for parallel execution by the assembly programmer or compiler.
  - Two MAC units.
- Complex, compound instructions:
  - Assembly source code compatible with C54xx
  - Mixed-width instructions: 8 to 48 bits.
  - 200 MHz @ 1.5 V, ~130 mW, $17 qty 10k
- Poor compiler target.
An Enhanced Conventional DSP:
TI TMSC55xx

- Memory:
  - Prog/Data ROM
  - Prog/Data SARAM
  - Prog/Data DARAM
  - Instr. Cache
  - Instr. Buffer Unit
  - Prog. Flow Unit

- Instruction Bus: (1 x 32 bits)
- Data Buses: (3 x 16 bits for read, 2 x 16 bits for write)
- Address Buses: (6 x 24 bits)

- Data Path:
  - MAC
  - ALU
  - Shifter

- Addr. Gen.:
  - Addr./Data Registers
  - Addr. Units (3)

- Same as 54xx
- New on 55xx
- Enhanced on 55xx
The TMS320C62xx is the first fixed-point DSP processor from Texas Instruments that is based on a VLIW-like architecture which allows it to execute up to eight 32-bit RISC-like instructions per clock cycle.
C6201 Internal Memory Architecture

- Separate Internal Program and Data Spaces
  - Program
    - 16K 32-bit instructions (2K Fetch Packets)
    - 256-bit Fetch Width
    - Configurable as either
      - Direct Mapped Cache, Memory Mapped Program Memory
  - Data
    - 32K x 16
    - Single Ported Accessible by Both CPU Data Buses
    - 4 x 8K 16-bit Banks
      - 2 Possible Simultaneous Memory Accesses (4 Banks)
      - 4-Way Interleave, Banks and Interleave Minimize Access Conflicts
C62x Datapaths

Registers A0 - A15

Registers B0 - B15

DDATA_I1 (load data)

DDATA_O1 (store data)

DADDR1 (address)

DADDR2 (address)

DDATA_O2 (store data)

Cross Paths

40-bit Write Paths (8 MSBs)

40-bit Read Paths/Store Paths
C62x Functional Units

- **L-Unit (L1, L2)**
  - 40-bit Integer ALU, Comparisons
  - Bit Counting, Normalization

- **S-Unit (S1, S2)**
  - 32-bit ALU, 40-bit Shifter
  - Bitfield Operations, Branching

- **M-Unit (M1, M2)**
  - 16 x 16 -> 32

- **D-Unit (D1, D2)**
  - 32-bit Add/Subtract
  - Address Calculations
C62x Instruction Packing
Instruction Packing Advanced VLIW

Example 1

- Fetch Packet
  - CPU fetches 8 instructions/cycle
- Execute Packet
  - CPU executes 1 to 8 instructions/cycle
  - Fetch packets can contain multiple execute packets
- Parallelism determined at compile / assembly time
- Examples
  - 1) 8 parallel instructions
  - 2) 8 serial instructions
  - 3) Mixed Serial/Parallel Groups
    - A // B
    - C
    - D
    - E // F // G // H

Example 2

- A
- B
- C
- D
- E
- F
- G
- H

Example 3

- A
- B
- C
- D
- E
- F
- G
- H

Reduces Codesize, Number of Program Fetches, Power Consumption
C62x Pipeline Operation
Pipeline Phases

- Single-Cycle Throughput
- Operate in Lock Step
- Fetch
  - PG  Program Address Generate
  - PS  Program Address Send
  - PW  Program Access Ready Wait
  - PR  Program Fetch Packet Receive
- Decode
  - DP  Instruction Dispatch
  - DC  Instruction Decode
- Execute
  - E1 - E5  Execute 1 through Execute 5
C62x Pipeline Operation

Delay Slots

- Delay Slots: number of extra cycles until result is:
  - written to register file
  - available for use by a subsequent instructions
  - Multi-cycle NOP instruction can fill delay slots while minimizing code size impact

Most Instructions: **E1** No Delay

Integer Multiply: **E1 E2** 1 Delay Slots

Loads: **E1 E2 E3 E4 E5** 4 Delay Slots

Branches: **E1**

Branch Target: **PG PSP WPR DP DC E1** 5 Delay Slots
C6000 Instruction Set Features
Conditional Instructions

- All Instructions can be Conditional
  - A1, A2, B0, B1, B2 can be used as Conditions
  - Based on Zero or Non-Zero Value
  - Compare Instructions can allow other Conditions (<, >, etc)

- Reduces Branching
- Increases Parallelism
C6000 Instruction Set Addressing

Features

- Load-Store Architecture
- Two Addressing Units (D1, D2)
- Orthogonal
  - Any Register can be used for Addressing or Indexing
- Signed/Unsigned Byte, Half-Word, Word, Double-Word Addressable
  - Indexes are Scaled by Type
- Register or 5-Bit Unsigned Constant Index
C6000 Instruction Set Addressing Features

- Indirect Addressing Modes
  - Pre-Increment *++R[index]
  - Post-Increment *R++[index]
  - Pre-Decrement *--R[index]
  - Post-Decrement *R--[index]
  - Positive Offset *+R[index]
  - Negative Offset *-R[index]

- 15-bit Positive/Negative Constant Offset from Either B14 or B15

- Circular Addressing
  - Fast and Low Cost: Power of 2 Sizes and Alignment
  - Up to 8 Different Pointers/Buffers, Up to 2 Different Buffer Sizes

- Dual Endian Support
Application: FIR Filter on a TMS320C5x

```
COEFFP .set 02000h ; Program mem address
X .set 037Fh ; Newest data sample
LASTAP .set 037FH ; Oldest data sample

... LAR AR3, #LASTAP ; Point to oldest sample
RPT #127
MACD COEFFP, *- ; Do the thing
APAC
SACH Y,1 ; Store result -- note shift
```
Application: FIR Filter on a TMS320C62x

Single-Cycle Loop

... 

C7:  ldh .D1 *A1++, A2  ; Read coefficient
    ldh .D2 *B1++, B2  ; Read data
    [B0] sub .L2 B0, 1, B0  ; Decrement counter
    [B0] B .S2 c7  ; Branch if not zero
    mpy .Mlx A2, B2, A3  ; Form product
    add .L1 A4, A3, A4  ; Accumulate result

...
TI TMS320C64xx

• Announced in February 2000, the TMS320C64xx is an extension of Texas Instruments' earlier TMS320C62xx architecture.

• The TMS320C64xx has 64 32-bit general-purpose registers, twice as many as the TMS320C62xx.

• The TMS320C64xx instruction set is a superset of that used in the TMS320C62xx, and, among other enhancements, adds significant SIMD processing capabilities:
  – 8-bit operations for image/video processing.

• 600 MHz clock speed, but:
  – 11-stage pipeline with long latencies
  – Dynamic caches.

• $100 qty 10k.

• The only DSP family with compatible fixed and floating-point versions.
Superscalar DSP:
LSI Logic ZSP400

- A 4-way superscalar dynamically scheduled 16-bit fixed-point DSP core.
- 16-bit RISC-like instructions
- Separate on-chip caches for instructions and data
- Two MAC units, two ALU/shifter units  
  - Limited SIMD support.
  - MACS can be combined for 32-bit operations.
- Disadvantage:
  - Dynamic behavior complicates DSP software development:
    - Ensuring real-time behavior
    - Optimizing code.