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
  - Single program
  - Lightweight, often realtime OS
  - DSP support
  - 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

- **$30B**
  - 32-bit micro
    - $1.2B/4%
    - 32 bit DSP
      - $5.2B/17%
      - DSP
        - $10B/33%
        - 16-bit micro
          - $5.7B/19%
          - 8-bit micro
            - $9.3B/31%
The Processor Design Space

Performance vs. Cost

- **Microcontrollers**: Cost is everything
- **Embedded processors**: Application specific architectures for performance
- **Microprocessors**: Performance is everything & Software rules
DSP is the fastest growing segment of the semiconductor market.
DSP Applications

- Audio applications
- MPEG Audio
- Portable audio
- Digital cameras
- Wireless
- Cellular telephones
- Base station

- Networking
- Cable modems
- ADSL
- VDSL
Another Look at DSP Applications

- **High-end**
  - Wireless Base Station - TMS320C6000
  - Cable modem
  - gateways
- **Mid-end**
  - 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
  - Basestations
  - 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

---

EECC551 - Shaaban

#7  lec # 14  Fall 2000  10-31-2000
# 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</td>
<td>• Bipolar SSI, MSI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Simulation</td>
<td>• FFT algorithm</td>
</tr>
<tr>
<td>Late 1970’s</td>
<td>• Building block</td>
<td>• Military radars</td>
<td>• Single chip bipolar multiplier</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Digital Comm.</td>
<td>• Flash A/D</td>
</tr>
<tr>
<td>Early 1980’s</td>
<td>• Single Chip DSP $^\text{HP}$</td>
<td>• Telecom</td>
<td>• $^\text{HP}$ architectures</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Control</td>
<td>• NMOS/CMOS</td>
</tr>
<tr>
<td>Late 1980’s</td>
<td>• Function/Application specific chips</td>
<td>• Computers</td>
<td>• Vector processing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Communication</td>
<td>• Parallel processing</td>
</tr>
<tr>
<td>Early 1990’s</td>
<td>• Multiprocessing</td>
<td>• Video/Image Processing</td>
<td>• Advanced multiprocesssing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Wireless telephony</td>
<td>• VLIW, MIMD, etc.</td>
</tr>
<tr>
<td>Late 1990’s</td>
<td>• Single-chip multiprocessing</td>
<td>• Internet related</td>
<td>• Low power single-chip DSP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Multiprocessing</td>
</tr>
</tbody>
</table>
CELLULAR TELEPHONE SYSTEM

CONTROLLER

RF MODEM

PHYSICAL LAYER PROCESSING

BASEBAND CONVERTER

A/D

SPEECH ENCODE

SPEECH DECODE

DAC

1 2 3
4 5 6
7 8 9
0

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

- RAM
- S/P
- DMA
- μC
- ASIC LOGIC
- DSP CORE
- S/P DMA
- phone book intfc
- control protocol
- speech quality enhancement
- de-intl & decoder
- voice recognition
- RPE-LTP speech decoder
- demodulator and synchronizer
- Viterbi equalizer

EECC551 - Shaaban
#11 lec # 14 Fall 2000 10-31-2000
Example Wireless Phone Organization
Multimedia I/O Architecture

- Radio Modem
- Embedded Processor
- Sched ECC Pact
- Interface

Low Power Bus

- FB
- Fifo
- SRAM
- Graphics
- Pen

- Fifo
- Audio

- Video Decomp
- Video

Data Flow
Multimedia System-on-a-Chip

E.g. Multimedia terminal electronics

- Future chips will be a mix of processors, memory and dedicated hardware for specific algorithms and I/O
Requirements of the 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 reuseable code
  - Customizable core
- Low power if application requires portability
Area of processor cores = Cost

- Nintendo processor
- Cellular phones
Another figure of merit: Computation per unit area

Nintendo processor

Cellular phones

MIPS/mm²

<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.
DSP BENCHMARKS - DSPstone

- ZIVOJNOVIC, VERLADE, SCHLAGER: 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
Evolution of GP and DSP

• General Purpose Microprocessor traces roots back to Eckert, Mauchly, Von Neumann (ENIAC)
• DSP evolved from Analog Signal Processors, 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
Embedded Systems vs. General Purpose Computing - 1

Embedded System
- Runs a few applications often known at design time
- Not end-user programmable
- Operates in fixed run-time constraints, additional performance may not be useful/valuable
- Differentiating features:
  - power
  - cost
  - speed (must be predictable)

General purpose computing
- Intended to run a fully general set of applications
- End-user programmable
- Faster is always better
- Differentiating features
  - speed (need not be fully predictable)
  - cost (largest component power)
DSP vs. General Purpose MPU

• DSPs tend to be written for 1 program, not many programs.
  – Hence OSes are much simpler, there is no virtual memory or protection, ...

• DSPs sometimes run hard real-time apps
  – 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 have an infinite continuous data stream
DSP vs. General Purpose MPU

• The “MIPS/MFLOPS” of DSPs is speed of Multiply-Accumulate (MAC).
  – DSP are judged by whether they can keep the multipliers busy 100% of the time.
• The "SPEC" of DSPs is 4 algorithms:
  – Infinite Impulse Response (IIR) filters
  – Finite Impulse Response (FIR) filters
  – FFT, and
  – convolvers
• In DSPs, algorithms are important:
  – Binary compatibility not an issue
• High-level Software is not (yet) important in DSPs.
  – People still write in assembly language for a product to minimize the die area for ROM in the DSP chip.
TYPES OF DSP PROCESSORS

• DSP Multiprocessors on a die
  – TMS320C80
  – TMS320C6000

• 32-BIT FLOATING POINT
  – TI TMS320C4X
  – MOTOROLA 96000
  – AT&T DSP32C
  – ANALOG DEVICES ADSP21000

• 16-BIT FIXED POINT
  – TI TMS320C2X
  – MOTOROLA 56000
  – AT&T DSP16
  – ANALOG DEVICES ADSP2100
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

\[
\begin{align*}
S_{-1} & \leq x < 1 \\
S_{-2^{N-1}} & \leq x < 2^{N-1}
\end{align*}
\]
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 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

DSP Processor

- Specialized hardware performs all key arithmetic operations in 1 cycle.
- Hardware support for managing numeric fidelity:
  - Shifters
  - Guard bits
  - Saturation

General-Purpose Processor

- Multiplies often take >1 cycle
- Shifts often take >1 cycle
- Other operations (e.g., saturation, rounding) typically take multiple cycles.
320C54x DSP Functional Block Diagram
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 $\times$ or
- Add is $+$ or
- Delay/Storage is $z^{-1}$ or

\[ \text{Delay} \quad D \]
FIR Filtering: A Motivating Problem

• 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: 1 FIR Tap / DSP instruction cycle
FINITE-IMPULSE RESPONSE (FIR) FILTER
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
First Generation DSP (1982): Texas Instruments TMS32010

- 16-bit fixed-point
- “Harvard architecture”
  - separate instruction, data memories
- Accumulator
- Specialized instruction set
  - Load and Accumulate
- 390 ns Multiple-Accumulate (MAC) time; 228 ns today
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

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

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.
DSP Memory

• FIR Tap implies multiple memory accesses
• DSPs want 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

Diagram:
- **READ/WRITE**
- **PROCESSOR**
- **DATA AND PROGRAM MEMORY**
- **SERIAL PORT**
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. 320C62x/67x DSP

Program RAM/cache
32-bit address
256-bit data
512K bits RAM

Data RAM
32-bit address
8-, 16-, 32-bit data
512K bits RAM

JTAG test/emulation control

Multichannel
(T1/E1) buffered
serial port

Multichannel
(T1/E1) buffered
serial port

Timer

Timer

PLL clock
generator

'86000 CPU core
Program fetch
Instruction dispatch
Instruction decode
Data path 1
A register file
.L1, S1, M1, D1
Data path 2
B register file
.L2, S2, M2, D2

Control registers
Control logic
Test
Emulation
Interrupts

DMA
(four channel) or
EDMA (16 channel)

EXB or
Host port

Power management
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

  0 (000) => 0 (000)
  1 (001) => 4 (100)
  2 (010) => 2 (010)
  3 (011) => 6 (110)
  4 (100) => 1 (001)
  5 (101) => 5 (101)
  6 (110) => 3 (011)
  7 (111) => 7 (111)

• 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
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)

- CompactRISC CORE
- DSPM
- 2 Kbyte DSPM RAM
- 40 Kbyte ROM
- 1.4 Kbyte System RAM
- Adapter
- CORE BUS
- Peripheral BUS
- Memory Controller
- I/O Ports
- ICU
- MICROWIRE I/F
- PWM
- Watchdog
- Codec I/F
- Clock & Timers

11/10/95
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.
## 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 Based</strong> (Harvard Architecture)</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></td>
<td>40 MIPS</td>
<td>25</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td><strong>Multiprocessor 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>1600 MIPS</td>
<td>5</td>
<td>2 GOPS</td>
<td>MIMD</td>
</tr>
<tr>
<td>TMS320C62XX</td>
<td>1997</td>
<td>16 integer</td>
<td></td>
<td></td>
<td></td>
<td>20</td>
<td>20 GOPS VLIW</td>
</tr>
<tr>
<td>TMS310C67XX</td>
<td>1997</td>
<td>32 flt. pt.</td>
<td></td>
<td></td>
<td>5</td>
<td>1 GFLOP VLIW</td>
<td></td>
</tr>
</tbody>
</table>
First Generation DSP µP Case Study
TMS32010 (Texas Instruments) - 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 BLOCK DIAGRAM

Legend:
ACC = Accumulator
ALU = Arithmetic Logic Unit
ARP = Auxiliary Register Pointer
AR0 = Auxiliary Register 0
AR1 = Auxiliary Register 1
DP = Data Page Pointer
P = P Register
PC = Program Counter
T = T Register

EECC551 - Shaaban
#66 lec # 14  Fall 2000  10-31-2000
Third Generation DSP µP Case Study
TMS320C30 - 1988

TMS320C30 Key Features

- 60 ns single-cycle instruction execution time
  - 33.3 MFLOPS (million floating-point operations per second)
  - 16.7 MIPS (million instructions per second)
- One 4K x 32-bit single-cycle dual-access on-chip ROM block
- Two 1K x 32-bit single-cycle dual-access on-chip RAM blocks
- 64 x 32-bit instruction cache
- 32-bit instruction and data words, 24-bit addresses
- 40/32-bit floating-point/integer multiplier and ALU
- 32-bit barrel shifter
- Eight extended precision registers (accumulators)
- Two address generators with eight auxiliary registers and two auxiliary register arithmetic units
- On-chip direct memory Access (DMA) controller for concurrent I/O and CPU operation
- Parallel ALU and multiplier instructions
- Block repeat capability
- Interlocked instructions for multiprocessing support
- Two serial ports to support 8/16/32-bit transfers
- Two 32-bit timers
- 1 µ CDMOS Process
TMS320C3x CPU BLOCK DIAGRAM
TMS320C30 FIR FILTER PROGRAM

\[ Y(n) = x[n-(N-1)] \cdot h(N-1) + x[n-(N-2)] \cdot h(N-2) + \ldots + x(n) \cdot h(0) \]

For \( N=50 \), \( t=3.6 \ \mu s \) (277 KHz)
Texas Instruments TMS320C80
MIMD MULTIPROCESSOR DSP (1996)
16 bit Fixed Point VLIW DSP:
TMS320C6201 Revision 2 (1997)
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)

DDATA_I2 (load data)

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
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

Example 3

- 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

Execute Packet 2
Execute Packet 3
Execute Packet 4
Execute Packet 5
Execute Packet 6
Execute Packet 7
C62x Pipeline Operation

Delay Slots

- Delay Slots: number of extra cycles until result is:
  - written to register file
  - available for use by a subsequent instruction
  - Multi-cycle NOP instruction can fill delay slots while minimizing codesize 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 PSPWPR 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
32 Bit Floating Point VLIW DSP: TMS320C6701 (1997)

- Program Cache/Program Memory
  - 32-bit address, 256-Bit data
  - 512K Bits RAM

- 'C67x Floating-Point CPU Core
  - Program Fetch
  - Instruction Dispatch
  - Instruction Decode
  - Data Path 1: A Register File
    - L1, S1, M1, D1
  - Data Path 2: B Register File
    - D2, M2, S2, L2

- Control Registers
- Control Logic
- Test
- Emulation
- Interrupts

- Data Memory
  - 32-Bit address
  - 8-, 16-, 32-Bit data
  - 512K Bits RAM

- External Memory Interface
- 4 Channel DMA
- Host Port Interface
- Power Down

- 2 Timers
- 2 Multi-channel buffered serial ports (T1/E1)
TMS320C6701
Advanced VLIW CPU (VelociTI™)

• 1 GFLOPS @ 167 MHz
  – 6-ns cycle time
  – 6 x 32-bit floating-point instructions/cycle
• Load store architecture
• 3.3-V I/Os, 1.8-V internal
• Single- and double-precision IEEE floating-point
• Dual data paths
  – 6 floating-point units / 8 x 32-bit instructions
• External interface supports
  – SDRAM, SRAM, SBSRAM
• 4-channel bootloading DMA
• 16-bit host port interface
• 1Mbit on-chip SRAM
• 2 multichannel buffered serial ports (T1/E1)
• Pin compatible with ’C6201
TMS320C67x CPU Core

'C67x Floating-Point CPU Core

Program Fetch
Instruction Dispatch
Instruction Decode

Data Path 1
A Register File
L1 S1 M1 D1

Data Path 2
B Register File
D2 M2 S2 L2

Control Registers
Control Logic
Test
Emulation
Interrupts

Floating-Point Capabilities

Arithmetic Logic Unit
Auxiliary Logic Unit
Multiplier Unit
C67x New Instructions

.L Unit
- ADDSP
- ADDDP
- SUBSP
- SUBDP
- INTSP
- INTDP
- SPINT
- DPRINT
- SPTRUNC
- DPTRUNC
- DSPS

.M Unit
- MPYSP
- MPYDP
- MPYI
- MPYID
- MPY24
- MPY24H

.S Unit
- ABSSP
- ABSDP
- CMPGTSP
- CMPEQSP
- CMPLTSP
- CMPGTDP
- CMPEQDP
- CMPLTDP
- RCPSP
- RCPDP
- RSQRSP
- RSQRDP
- SPD
C67x Datapaths

- 2 Data Paths
- 8 Functional Units
  - Orthogonal/Independent
  - 2 Floating Point Multipliers
  - 2 Floating Point Arithmetic
  - 2 Floating Point Auxiliary
- Control
  - Independent
  - Up to 8 32-bit Instructions
- Registers
  - 2 Files
  - 32, 32-bit registers total
- Cross paths (1X, 2X)

L-Unit (L1, L2)
  - Floating-Point, 40-bit Integer ALU
  - Bit Counting, Normalization
S-Unit (S1, S2)
  - Floating Point Auxiliary Unit
  - 32-bit ALU/40-bit shifter
  - Bitfield Operations, Branching
M-Unit (M1, M2)
  - Multiplier: Integer & Floating-Point
D-Unit (D1, D2)
  - 32-bit add/subtract Addr Calculations
C67x Instruction Packing
Instruction Packing Enhanced 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
- Reduces
  - Codesize
  - Number of Program Fetches
  - Power Consumption
C67x Pipeline Operation: Pipeline Phases

- 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
  - E6 - E10 Double Precision Only

Execute Packet 1
PG PS PW PR DP DC E1 E2 E3 E4 E5 E6 E7 E8 E9 E10

Execute Packet 2
PG PS PW PR DP DC E1 E2 E3 E4 E5 E6 E7 E8 E9 E10

Execute Packet 3
PG PS PW PR DP DC E1 E2 E3 E4 E5 E6 E7 E8 E9 E10

Execute Packet 4
PG PS PW PR DP DC E1 E2 E3 E4 E5 E6 E7 E8 E9 E10

Execute Packet 5
PG PS PW PR DP DC E1 E2 E3 E4 E5 E6 E7 E8 E9 E10
C67x 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 codesize impact

Most Integer

- No Delay

Single-Precision

- 3 Delay Slots

Loads

- 4 Delay Slots

Branches

- Branch Target

- 5 Delay Slots
’C67x and ’C62x Commonality

- Driving commonality between ’C67x & ’C62x shortens ’C67x design time.
- Maintaining symmetry between datapaths shortens the ’C67x design time.

<table>
<thead>
<tr>
<th>’C62x CPU</th>
<th>’C67x CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>M-Unit 1 Multiplier Unit</td>
<td>M-Unit 1 Multiplier Unit with Floating Point</td>
</tr>
<tr>
<td>D-Unit 1 Data Load/Store</td>
<td>D-Unit 1 Data Load/Store</td>
</tr>
<tr>
<td>S-Unit 1 Auxiliary Logic Unit</td>
<td>S-Unit 1 Auxiliary Logic Unit with Floating Point</td>
</tr>
<tr>
<td>L-Unit 1 Arithmetic Logic Unit</td>
<td>L-Unit 1 Arithmetic Logic Unit with Floating Point</td>
</tr>
<tr>
<td>Register file</td>
<td>Register file</td>
</tr>
<tr>
<td>Decode</td>
<td>Decode</td>
</tr>
<tr>
<td>Program Fetch &amp; Dispatch</td>
<td>Program Fetch &amp; Dispatch</td>
</tr>
</tbody>
</table>

‘C67x and ‘C62x Commonality

- Driving commonality between ’C67x & ’C62x shortens ’C67x design time.
- Maintaining symmetry between datapaths shortens the ’C67x design time.

<table>
<thead>
<tr>
<th>’C62x CPU</th>
<th>’C67x CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>M-Unit 1 Multiplier Unit</td>
<td>M-Unit 1 Multiplier Unit with Floating Point</td>
</tr>
<tr>
<td>D-Unit 1 Data Load/Store</td>
<td>D-Unit 1 Data Load/Store</td>
</tr>
<tr>
<td>S-Unit 1 Auxiliary Logic Unit</td>
<td>S-Unit 1 Auxiliary Logic Unit with Floating Point</td>
</tr>
<tr>
<td>L-Unit 1 Arithmetic Logic Unit</td>
<td>L-Unit 1 Arithmetic Logic Unit with Floating Point</td>
</tr>
<tr>
<td>Register file</td>
<td>Register file</td>
</tr>
<tr>
<td>Decode</td>
<td>Decode</td>
</tr>
<tr>
<td>Program Fetch &amp; Dispatch</td>
<td>Program Fetch &amp; Dispatch</td>
</tr>
</tbody>
</table>