Lecture 8: Digital Signal Processors

Professor David A. Patterson
Computer Science 252
Spring 1998

Vector Summary

- Vector is alternative model for exploiting ILP
- If code is vectorizable, then simpler hardware, more energy efficient, and better real-time model than Out-of-order machines
- Design issues include number of lanes, number of functional units, number of vector registers, length of vector registers, exception handling, conditional operations
- Will multimedia popularity revive vector architectures?

Review: Processor Classes

- General Purpose - high performance
  - Pentiums, Alpha's, SPARC
  - 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, ...

Increasing Cost ↑

Increasing Volume ↓

DSP Outline

- Intro
- Sampled Data Processing and Filters
- Evolution of DSP
- DSP vs. GP Processor
- Lecture material based "Introduction to Architectures for Digital Signal Processing" lecture by Bob Brodersen
  www.cs.berkeley.edu/~pattrsn/152F97/slides/CS152_dsp.pdf
  - Will refer to page from his lecture as "RB: i"
- and Dr. Jeff Bier "Evolution of Digital Signal Processing"
  www.cs.berkeley.edu/~pattrsn/152F97/slides/ slides.evolution.pdf
  - Will refer to page from his lecture as "JB: i"

DSP Introduction

- <u>Digital Signal Processing</u>: application of mathematical operations to digitally represented signals
- Signals represented digitally as <u>sequences of samples</u>
- Digital signals obtained from physical signals via <u>tranducers</u> (e.g., microphones) and <u>analog-to-digital converters (ADC)</u>
- Digital signals converted back to physical signals via <u>digital-to-analog converters (DAC)</u>
- <u>Digital Signal Processor (DSP)</u>: electronic system that processes digital signals

Common DSP algorithms and applications

- Applications – Instrumentation and measurement
  - Communications
  - Audio and video processing
  - Graphics, image enhancement, 3- D rendering
  - Navigation, radar, GPS
  - Control - robotics, machine vision, guidance
- Algorithms
  - Frequency domain filtering - FIR and IIR
  - Frequency- time transformations - FFT
  - Correlation

What Do DSPs Need to Do Well?

- Most DSP tasks require:
  - Repetitive numeric computations
  - Attention to numeric fidelity
  - High memory bandwidth, mostly via array accesses
  - Real-time processing
- DSPs must perform these tasks efficiently while minimizing:
  - Cost
  - Power
  - Memory use
  - Development time

DSP Application - equalization

- See RB slide 20

www.cs.berkeley.edu/~pattrsn/152F97/slides/CS152_dsp.pdf

- The audio data streams from the source (computer) through the digital analysis and synthesis
- Hard realtime requirement - the processing must be done at the sample rate

Who Cares?

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

A Tale of Two Cultures

- General Purpose Microprocessor traces roots back to Eckert, Mauchly, Von Neumann (ENIAC)
- DSP evolved from Analog Signal Processors, using analog hardware to transform phyical 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
- Increasing markets leading to cultural warfare

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

Today's DSP "Killer Apps"

- In terms of dollar volume, the biggest markets for DSP processors today include:
  - Digital cellular telephony
  - Pagers and other wireless systems
  - Modems
  - Disk drive servo control
- Most demand good performance
- All demand low cost
- Many demand high energy efficiency
- Trends are towards better support for these (and similar) major applications.

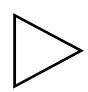
Digital Signal Processing in General Purpose Microprocessors

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

Decoding DSP Lingo

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

Decoding DSP Lingo

- Uses "flowchart" notation instead of equations
- Multiply is X or 
- Add is + or Σ
- Delay/Storage is **Delay** or z^{-1} or **D**

designed to keep computer architects without the secret decoder ring out of the DSP field?

CS 252 Administrivia

- Selected projects last week
- Upcoming events in CS 252

20-Feb DSP/Multimedia Processors #2 (Fri)
25-Feb Memory Hierachy: Caches; <u>Meeting signup</u>
25-Feb <u>Project Survey due (Wed)</u>
<u>26-Feb</u> <u>HW #2 due by 5:00 PM (Thu)</u>
27-Feb Memory Hierarchy Example; <u>6 minute Proj. Meetings 3:40–5:40</u>
4-Mar Quiz 1 (5:30PM – 8:30PM, 306 Soda) (Wed)
Pizza at LaVal's 8:30 – 10PM

Sampled data processing

- **RB Slides 22-30**

www.cs.berkeley.edu/~pattrsn/152F97/slides/CS152_dsp.pdf

FIR Filtering: A Motivating Problem

- **JB Slide 8**
  www.cs.berkeley.edu/~pattrsn/152F97/slides/ slides.evolution.pdf
- **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**

DSP Assumptions of the World

- **Machines issue/execute/complete in order**
- **Machines issue 1 instruction per clock**
- **Each line of assembly code = 1 instruction**
- **Clocks per Instruction = 1.000**
- **Floating Point is slow, expensive**

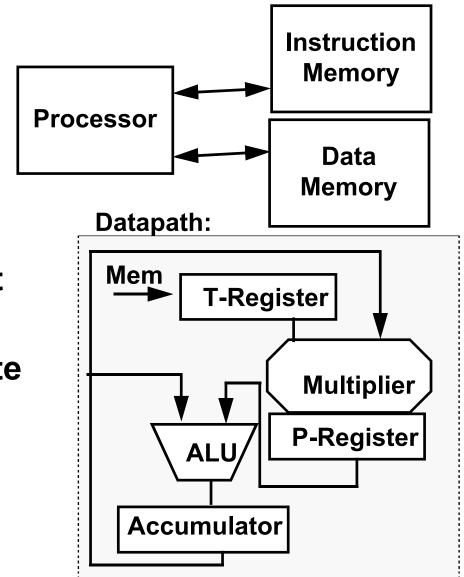
FIR filter on (simple) General Purpose Processor

```
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
...
```

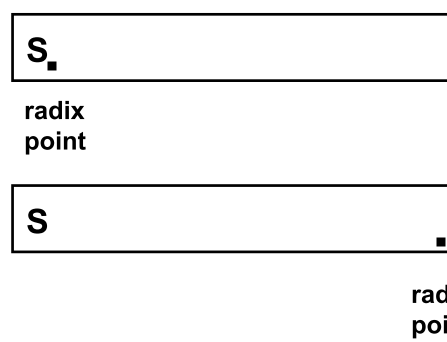
- Two instructions per tap, but requires unrolling

Features Common to Most DSP Processors

- Data path configured for DSP
- Specialized instruction set
- Multiple memory banks and buses
- Specialized addressing modes
- Specialized execution control
- 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 "<u>fixed point</u>" as well as integers



$-1 \leq x < 1$

$-2^{N-1} \leq x < 2^{N-1}$

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**
  - **Seperate explicit exponent**
- **"<u>Blocked Floating Point</u>" single exponent for a group of fractions**
- **Floating point support simplify development**

DSP Data Path: Overflow?

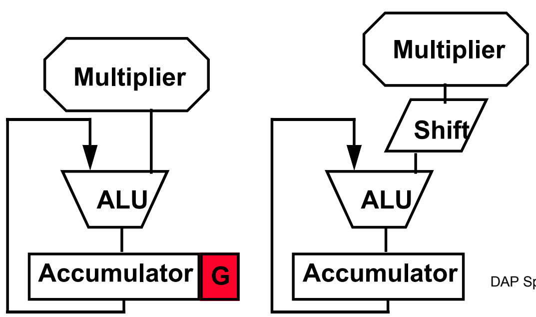
- **DSP are descended from analog : what should happen to output when "peg" an input? (e.g., turn up volume control knob on stereo)**
  - **Modulo Arithmetic???**
- **Set to most positive ($2^{N-1}-1$) or most negative value(-2^{N-1}) : "<u>saturation</u>"**
- **Many algorithms were developed in this 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: accumalator wider than product: "<u>guard bits</u>"**
  - **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 bais
- 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 bais
  IEEE 754 calls this round to nearest even

DSP Memory

- FIR Tap implies multiple memory accesses
- DSPs want multiple data ports
- Some DSPs have ad hoc techniques to reduce memory bandwdith demand
  - Instruction repeat buffer: do 1 instruction 256 times
  - Often disables interrupts, thereby increasing interrupt responce 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

DSP Addressing

- Have standard addressing modes: immediate, displacement, register indirect
- Want to keep MAC datapth 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: Buffers

- DSPs dealing with continuous I/O
- Often interact with an I/O buffer (delay lines)
- To save memory, buffer often organized as circular buffer
- 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

DSP Addressing: FFT

- FFTs start or end with data in wierd bufferfly 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 "<u>bit reverse</u>" address addressing mode for use with autoincrement addressing
- Many DSPs have "<u>bit reverse</u>" addressing for radix-2 FFT

DSP Instructions

- 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 reigster 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

DSP vs. General Purpose MPU

- DSPs are like embedded MPUs, very concerned about energy and cost.
  - So concerned about cost is that they might even us a 4.0 micron (not 0.40) to try to shrink the the wafer costs by using fab line with no overhead costs.
- DSPs that fail are often claimed to be good for something other than the highest volume application, but that's just designers fooling themselves.
- Very recently convention wisdom has changed so that you try to do everything you can digitally at low voltage so as to save energy.
  - 3 years ago people thought doing everything in analog reduced power, but advances inlower power digital design flipped that bit.

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:
  - Inifinite Impule Response (IIR) filters
  - Finite Impule Response (FIR) filters
  - FFT, and
  - convolvers
- In DSPs, algorithms are king!
  - Binary compatability not an issue
- Software is not (yet) king in DSPs.
  - People still write in assembly language for a product to minimize the die area for ROM in the DSP chip.

Generations of DSPs

- JB Slides 19, 21, 25, 29, 31, 32, 33
  www.cs.berkeley.edu/~pattrsn/152F97/slides/ slides.evolution.pdf
- (If time permits; otherwise do next time)

Summary: How are DSPs different?

- Essentially infinite streams of data which need to be processed in real time
- Relatively small programs and data storage requirements
- Intensive arithmetic processing with low amount of control and branching (in the critical loops)
- High amount of I/ O with analog interface
- Loosely coupled multiprocessor operation

Summary: How are DSPs different?

- Single cycle multiply accumulate (multiple busses and array multipliers)
- Complex instructions for standard DSP functions (IIR and FIR filters, convolvers)
- Specialized memory addressing
  - Modular arithmetic for circular buffers (delay lines)
  - Bit reversal (FFT)
- Zero overhead loops and repeat instructions
- I/ O support – Serial and parallel ports

Summary: Unique Features in DSP architectures

- Continuous I/O stream, real time requirements
- Multiple memory accesses
- Autoinc/autodec addressing
- Datapath
  - Multiply width
  - Wide accumulator
  - Guard bits/shiting rounding
  - Saturation
- Weird things
  - Circular addressing
  - Reverse addressing
- Special instructions
  - shift left and saturate (arithmetic left-shift)

Conclusions

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

For More Information

- `http://www.bdti.com`
  Collection of BDTI's papers on DSP processors, tools, and benchmarking.
- `http://www.eg3.com/dsp`
  Links to other good DSP sites.
- *Microprocessor Report*
  For info on newer DSP processors.
- *DSP Processor Fundamentals*,
  Textbook on DSP Processors, BDTI
- *IEEE Spectrum*, July, 1996
  Article on DSP Benchmarks
- *Embedded Systems Prog.,* October, 1996
  Article on Choosing a DSP Processor