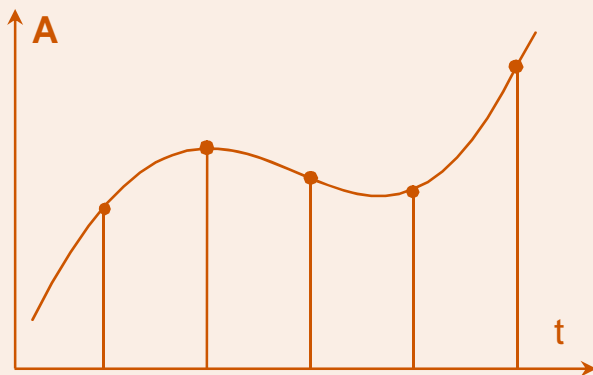# Digital Signal Processors

Mark McDermott

Agenda

- **What is Digital Signal Processing**

- **Key Algorithms**

- **HW Assist for Algorithms**

- **Interesting DSPs**

What is Digital Signal Processing?
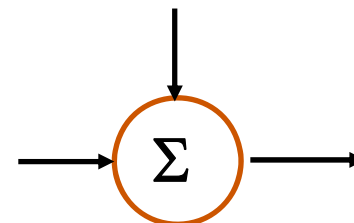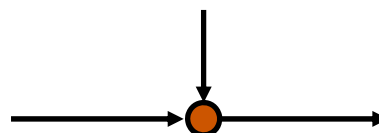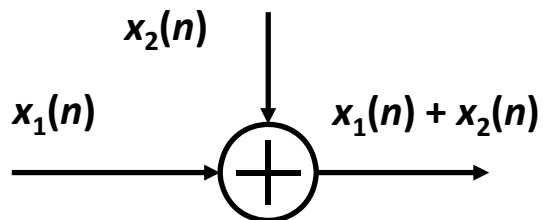

Digital sampling of an analog signal:

Most DSP algorithms can be expressed with a Multiply-Accumulate (MAC):

$$Y = \sum_{i=1}^{count} coeff_i * x_i$$

```
for (i = 0; i < count; i++){
   sum += c[i] * x[i]; }
```

Decoding DSP Lingo
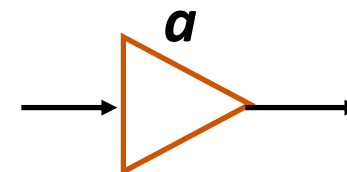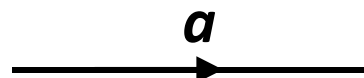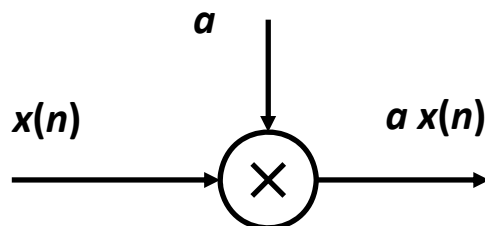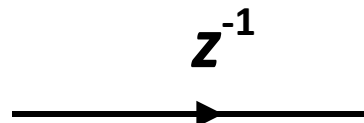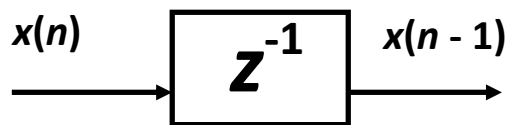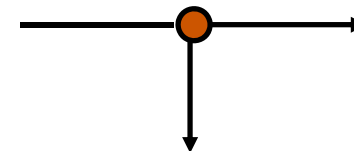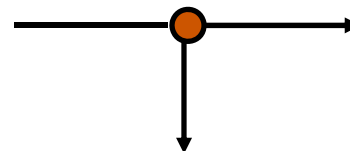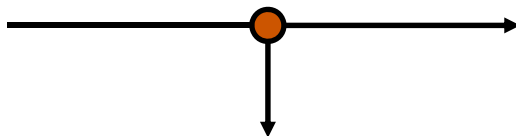
More Lingo

- **MIPS: million instructions per second**

- **MOPS: million (mathematical) operation per second**

- **MFLOPS: million floating-point operation per second**

- **MMACS: million MACs per second**

- **GMACS: gazillion MACs per second**

What is Digital Signal Processing? (cont)

- **Digital Signal Processing is the application of mathematical operations to digitally represented signals**

- **Signals represented digitally as sequences of samples**

- **Digital signals are obtained from physical signals via transducers (e.g., microphones) and analog-to-digital converters (ADC)**

- **Digital signals are converted back to physical signals via digital-to-analog converters (DAC)**

- **DSPs generally have an "infinite" continuous data stream**

- **Most DSP tasks require:**
  - **Repetitive numeric computations**
  - **Attention to numeric fidelity**
  - **High memory bandwidth**
  - **Real-time processing**

Gene's Law Drives DSP Development

Gene's Law

DSP Power

Gene's Law will have it's challenges to hold the line!
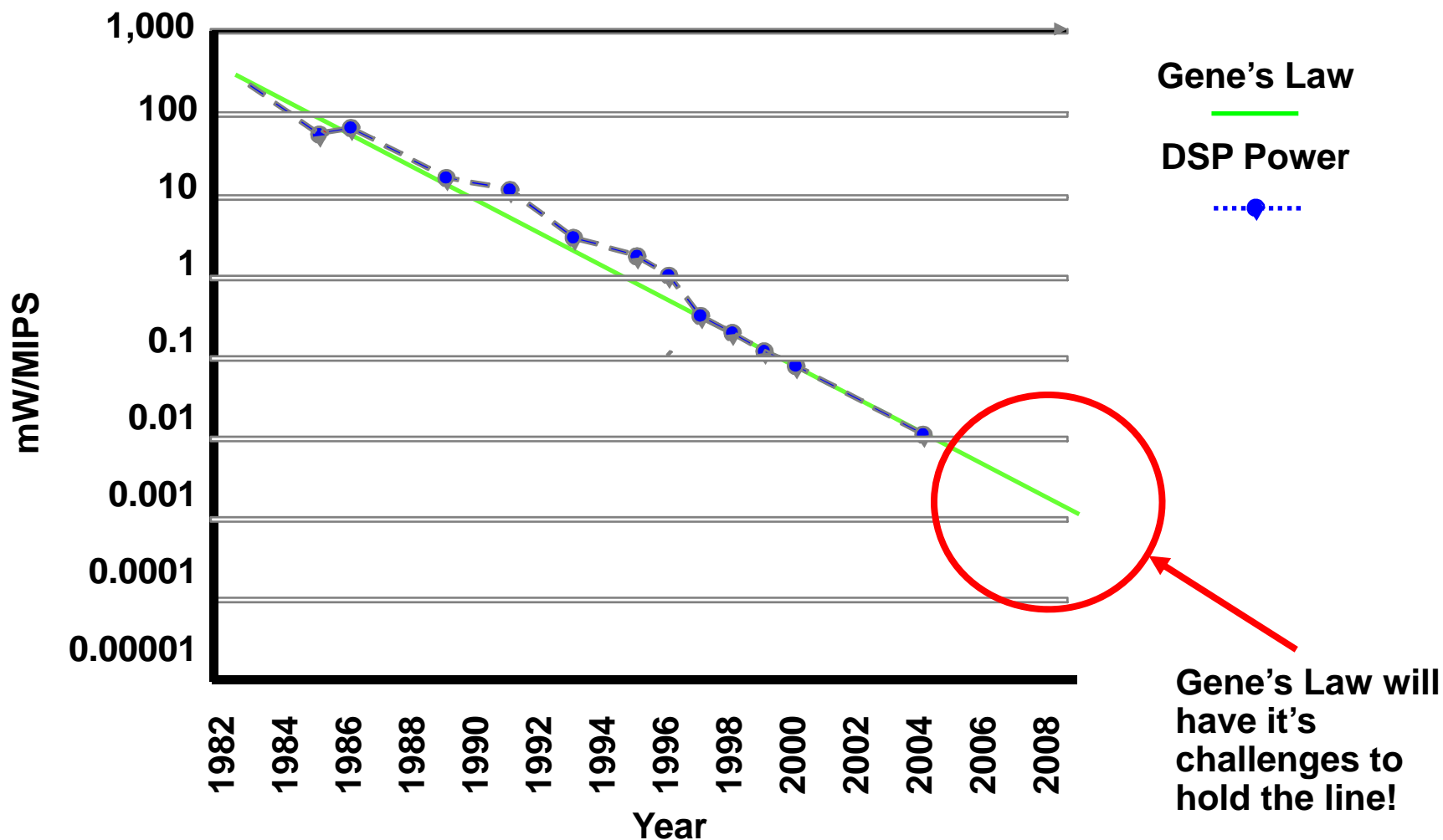
What's driving Gene's Law

Digital Audio

- MP3
- Real Audio

Streaming Video

- MPEG 4
- H.263

Connectivity

- Internet
- Bluetooth

Modem Standards

- UMTS
- GMS

What Makes a DSP a DSP?

- **Hard Real-Time**
  - **Single-Cycle MAC**
  - **Multiple Execution Units**
  - **Custom Data Path**
  - **High Bandwidth (Flat) Memory Sub-Systems**
  - **Dual Access Memory**
  - **Efficient Zero-Overhead Looping**
  - **Short Pipeline**
  - **High Bandwidth I/O**
  - **Specialized Instruction Sets**
  - **Low Latency Interrupts**
  - **Sophisticated DMA**
  - **No Speculation**
  - **RTOS**

- **Soft Real-Time (Application Processor)**
  - **Single-Cycle MAC**
  - **Multiple Execution Units**
  - **Custom Data Path**
  - **L1D$, L1I$, L2$ with MMU**
  - **Speculative Fetching and Branching**
  - **Virtual Memory**
  - **Protected Memory**
  - **Virtual Machines**
  - **Semaphores**
  - **Context Save and Restore**
  - **Threading: SMT, IMT**
  - **Efficient Zero-Overhead Looping**
  - **Short Pipeline**
  - **High Bandwidth I/O**
  - **Specialized Instruction Sets**
  - **Low Latency Interrupts**
  - **Sophisticated DMA**
  - **O/S**

DSP vs Processor Power Distribution

C54x
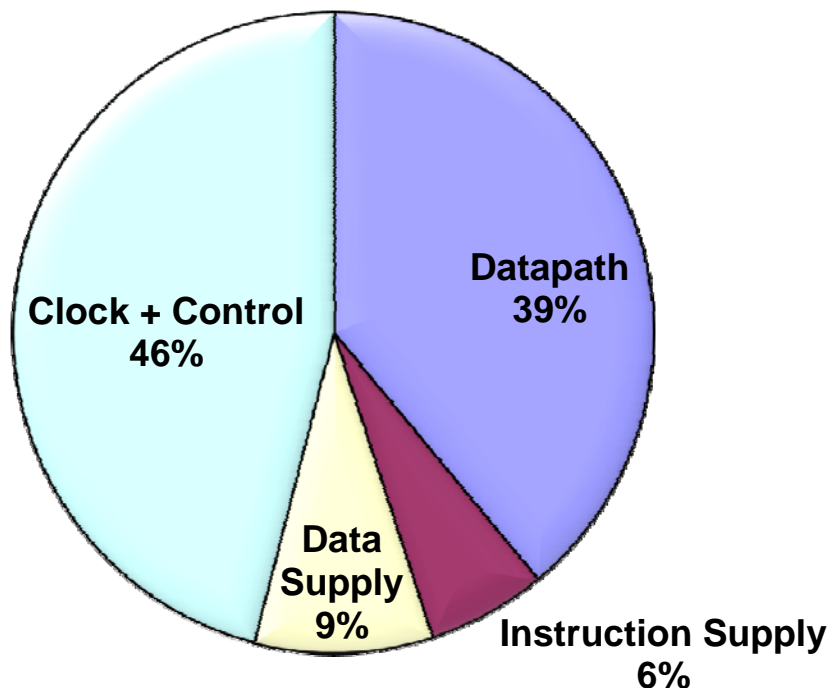
RISC GPP
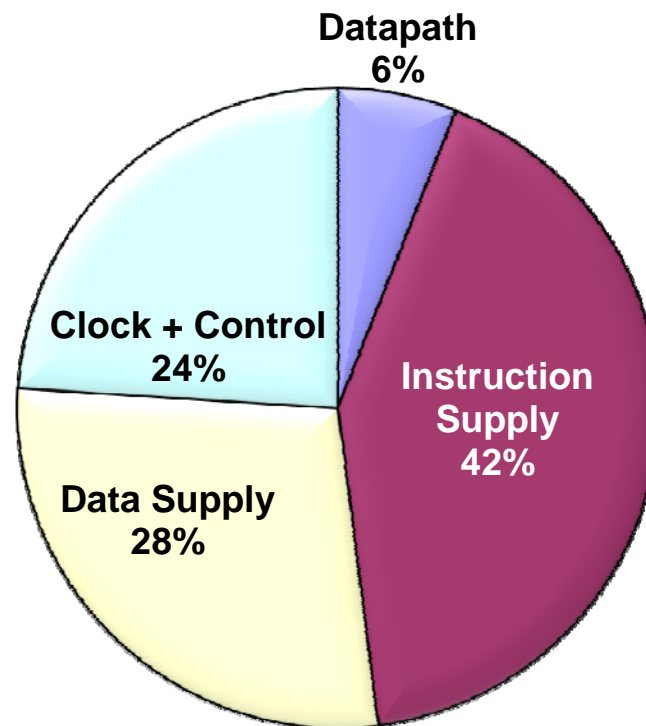
Techniques for optimizing energy

- **Optimizing Energy*Delay While Increasing ILP**
  - Memory Sub-system - Accessing (Flat) On Chip Memory At Speed Within 2-3 cycles
  - Multi-port Register File - Feeding Multiple VLIW Functional Units From a Single Register File
  - Pipeline - Running 1Ghz+ with a 7-9 Stage Pipeline
  - Datapath Control - Linking Multiple Functional Units with Result Forwarding
  - Branching – Supporting zero overhead loops
  - ISA – Balancing pipeline for both RISC and CISC instructions
  - Packet Headers - Achieving ARM Like Code Density

Key Algorithms

FIR Filter

- **Difference equation (vector dot product)**
  - y(n) = 2 x(n) + 3 x(n - 1) + 4 x(n - 2) + 5 x(n - 3)

- **Dot product of inputs vector and coefficient vector**

- **Store input in circular buffer, coefficients in array**

- **Multiply/Addition intensive**

- **Sum operation with high precision -- overflow considerations**

- **Long simple loop**

- **Online operation -- "infinite" amount of data**

- **Store coefficients on-chip for fast access**

$$y(n) = \sum_{i=0}^{N-1} a(i)\ x(n-i)$$

© smithmr@ucalgary.ca
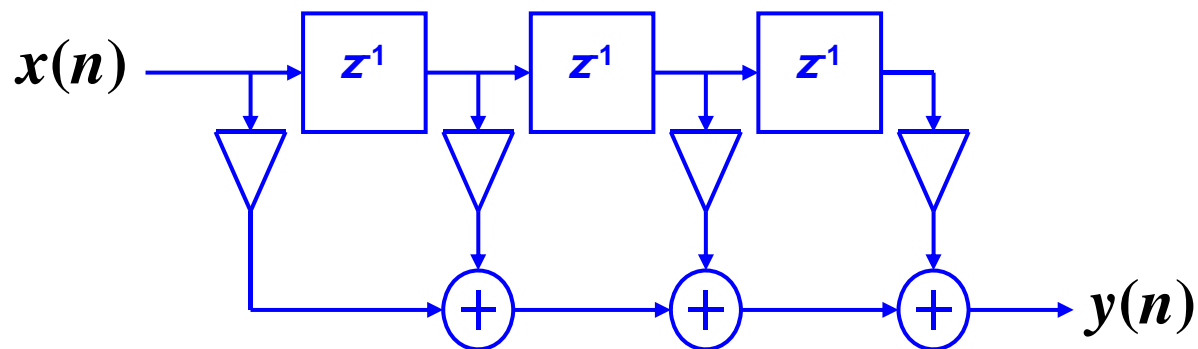
FIR Filter

■ **Each tap requires**

- **Fetching data sample**
- **Fetching coefficient**
- **Fetching operand**
- **Multiplying two numbers**
- **Accumulating multiplication result**
- **Possibly updating delay line (see below)**

One tap
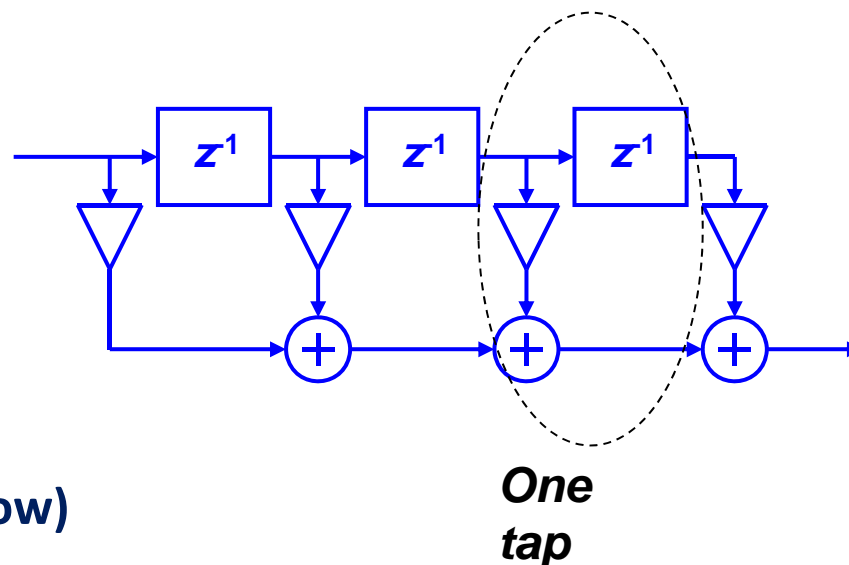
■ **Computing an FIR tap in one instruction cycle**

‣ **Two data memory and one program memory accesses**

‣ **Auto-increment or auto-decrement addressing modes**

‣ **Modulo addressing to implement delay line as circular buffer**

http://signal.ece.utexas.edu/

Symmetric FIR Filters

- **Impulse response often symmetric about midpoint**
  - **Phase of frequency response is linear**
  - **Example: three-tap FIR filter (M = 3) with h[0] = h[2]**

$$y[k] = h[0]\, x[k] + h[1]\, x[k-1] + h[2]\, x[k-2]$$
$$= h[0]\big(x[k] + x[k-2]\big) + h[1]\, x[k-1]$$

- **Implementation savings**
  - **Reduce number of multiplications from M to M/2 for even-length and to (M+1)/2 for odd-length impulse responses**
  - **Reduce storage of impulse response by same amount**
  - **TI TMS320C54 DSP has an accelerator instruction 'FIRS' to compute h[0] (x[k] + x[k-2]) in one instruction cycle**
  - **On most DSPs, no accelerator instruction is available**

http://signal.ece.utexas.edu/

Basic 2nd Order IIR Structures
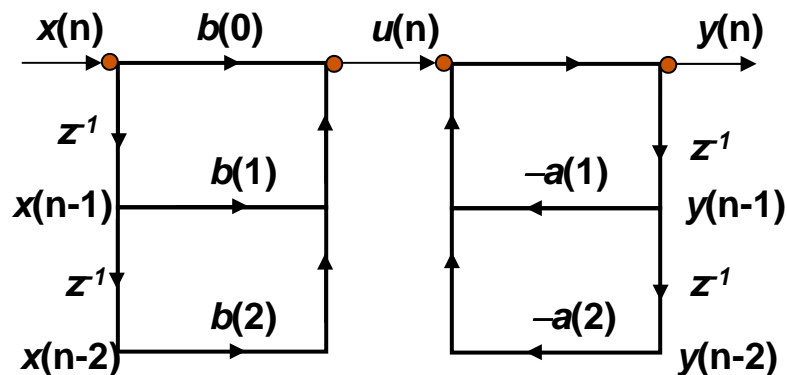
Direct form I realization
5 Multiply 4 Additions per y(n)
4 registers storing x(n-1), x(n-2), y(n-1), y(n-2)
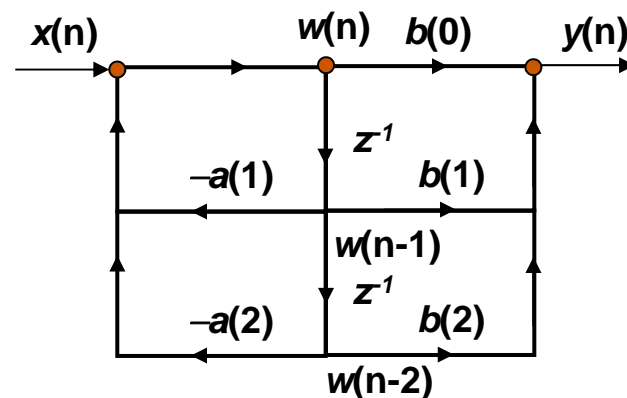
Direct form II realization a.k.a BiQuad
5 multiply, 4 additions per y(n)
2 registers storing w(n-1), w(n-2)

- **Interrelated and order dependent multiplications and additions**
- **Small number of delays via register moves?**
- **Short loop -- low number of instructions in loop which makes it difficult to optimize**
- **Precision -- very important because of feedback**
- **Multiple stages --** *I.e.* **IIR follows IIR etc**

Fast Fourier Transform

- **Complex variables (A and B) and fixed coefficients (W)**
- **Complex address calculations and memory accesses**
- **Multiplication and additions**
- **Need for fast access to many registers, address pointers, constants, variables**
- **Very hard to pipeline**
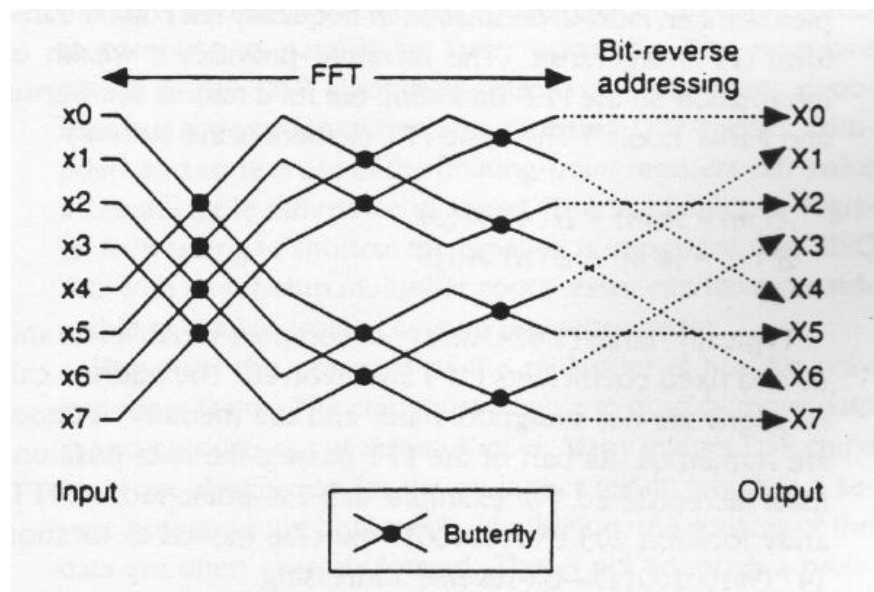
Vector Dot Product

■ **A vector dot product is common in filtering**

$$Y = \sum_{n=1}^{N} a(n)\ x(n)$$

− **Store *a*(*n*) and *x*(*n*) into an array of *N* elements**
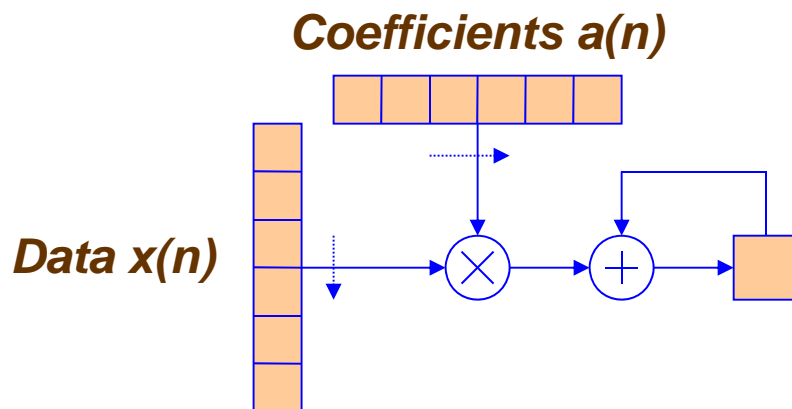
Adaptive Filtering

- **Self-learning: Filter coefficients adapt in response to training signal.**
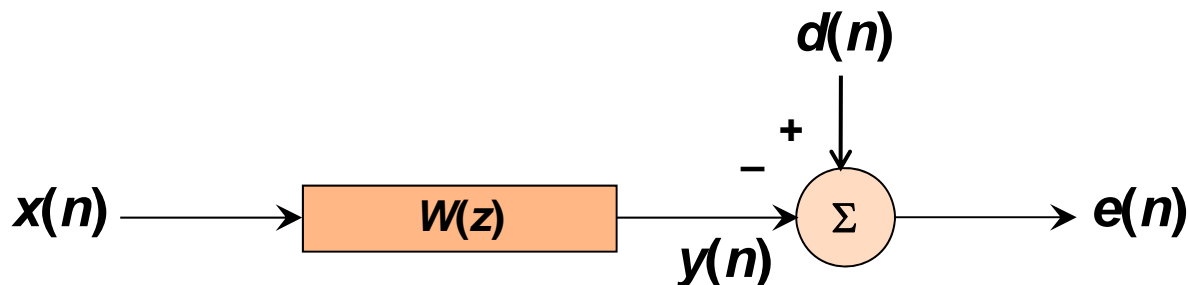
- **Filter update: Least Mean Squares (LMS) algorithm**

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n)$$

http://signal.ece.utexas.edu/

Discrete Cosine Transform

- **Arrows represent multiplication by -1**
- **a1=0.707, a2=0.541, a3=0.707, a4=1.307, a5=0.383**

DCT coefficients in bit-reversed order

[Arai, Agui & Nakajima]
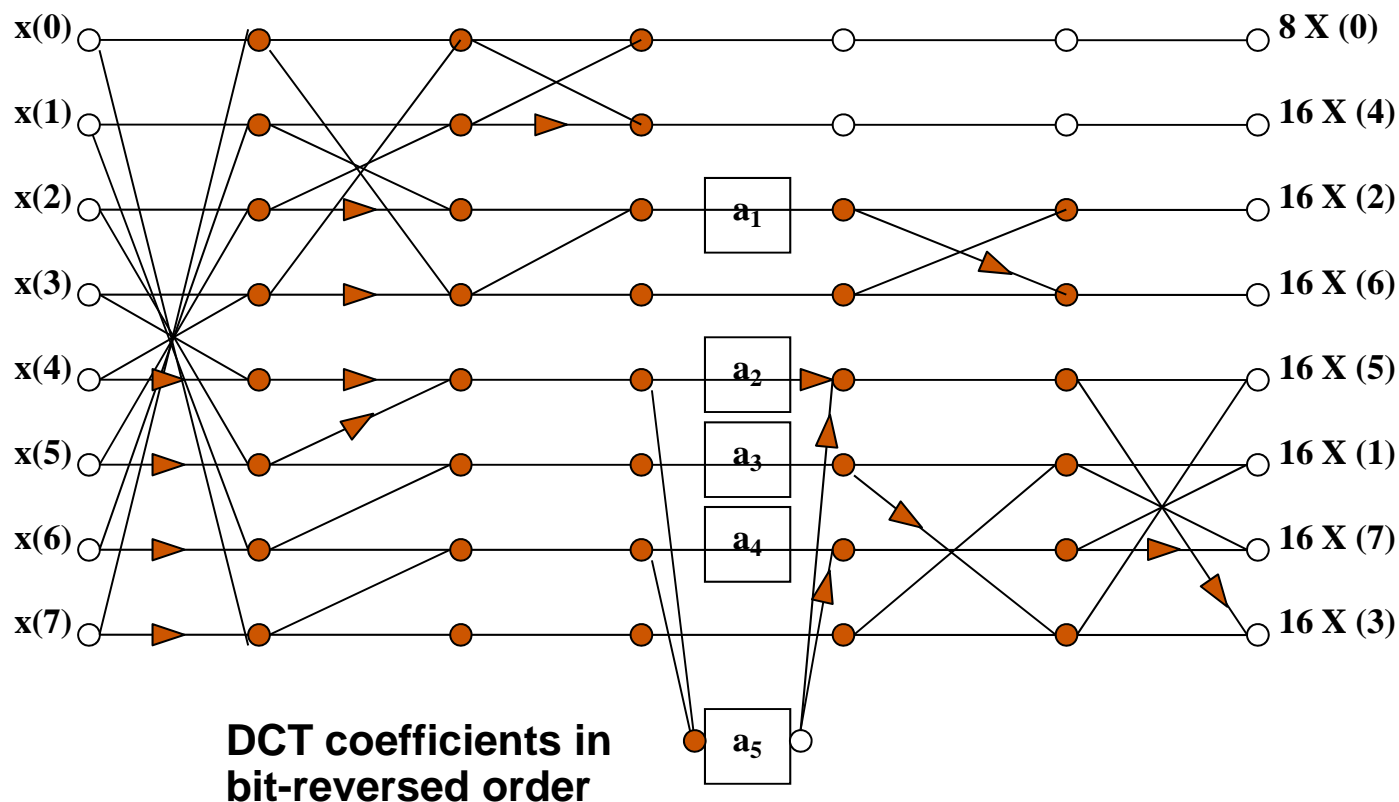
Hardware Assist for Algorithms

Hardware Assist for Algorithms

- **There are four key areas of hardware "assist" which are essential for DSP applications**
  - **Single cycle execution**
    - **Ability to fetch instructions and operands in a single cycle using multiple busses and on-chip memory**
    - **Ability to branch with zero cycle penalty**
    - **Pipelining**
  - **Operand addressing**
    - **Ability to generate addresses for specific algorithm**
  - **Optimized datapath(s)**
    - **Ability to support single cycle arithmetic operations**
    - **Multiple datapaths for parallel operations**
    - **Multiple local temp registers**
  - **Highly tuned instruction set architecture**
    - **Sophisticated instructions that execute in fewer cycles, with less code and low power demands**

Single cycle performance

■ **Harvard Architecture**
  – **Data memory/buses separate from program memory/bus**
  – **One read from program memory per instruction cycle**
  – **Two reads/writes from/to data memory per instruction cycle**
  – **Single cycle access to filter coefficients.**
    • **Multiport register files**
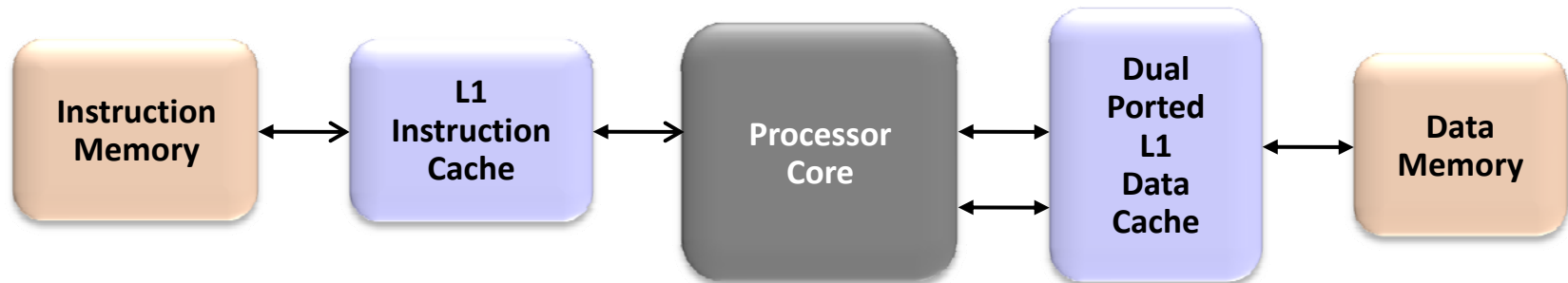
■ **Delayed Branching**
  – **Similar to MIPS style**

Single cycle performance (cont)

▪ Zero overhead (hardware) looping

- With (zero-overhead looping), specialized hardware is used to decrement the loop counter, test if it is zero, and branch.

Software looping	Hardware looping
MOVE #16, B	RPT #16
LOOP: MAC (RO)+, (R4)+, A	MAC (RO)+, (R4)+, A
DEC B	
BNE LOOP	

- Most DSPs support both single-instruction loops (like above) and multiple instruction loops (may use a special loop instruction buffer)
- Instruction(s) in loop may need to be fetched only once, thereby saving memory bandwidth.
- Less likely to be found in "compiler-friendly" DSPs

Pipelining

Sequential *(Motorola 56000)*

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

Pipelined *(Most conventional DSP processors)*

Fetch   **Decode**   **Read**   **Execute**
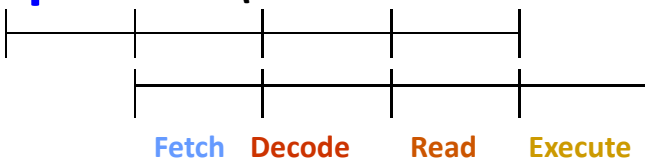
Superscalar *(Pentium, MIPS)*

Fetch   **Decode**   **Read**   **Execute**
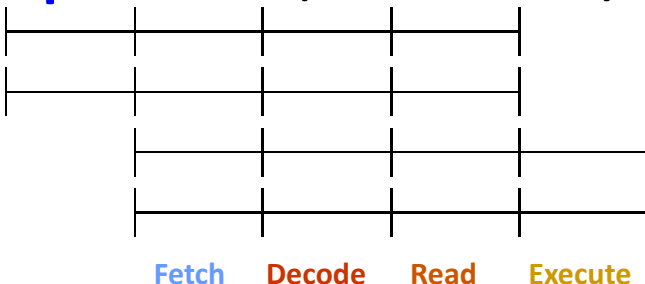
Superpipelined *(TMS320C6000)*

Fetch   **Decode**   **Execute**
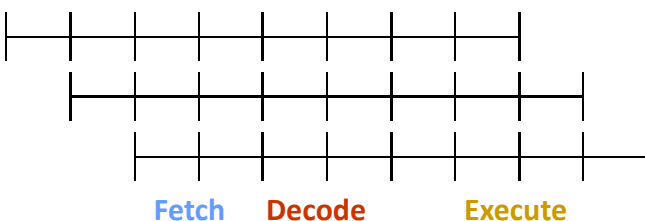
Managing Pipelines

- **compiler or programmer (TMS320C6000)**

- **pipeline interlocking in processor (TMS320C30)**

- **hardware instruction scheduling**

http://signal.ece.utexas.edu/

Operand addressing

- ## Register indirect addressing with post increment
  - Increment address pointer where repetitive computations are performed on a series of data.

- ## Linear buffer
  - Order by time index
  - Data shifting update: discard oldest data, copy old data left, insert new data

Data Shifting

Time	Buffer contents					Next sample
n=N	x_{N-K+1}	x_{N-K+2}	$---$	x_{N-1}	x_N	$\leftarrow$ x_{N+1}
n=N+1	x_{N-K+2}	x_{N-K+3}	$---$	x_N	x_{N+1}	$\leftarrow$ x_{N+2}
n=N+2	x_{N-K+3}	x_{N-K+4}	$---$	x_{N+1}	x_{N+2}	$\leftarrow$ x_{N+3}
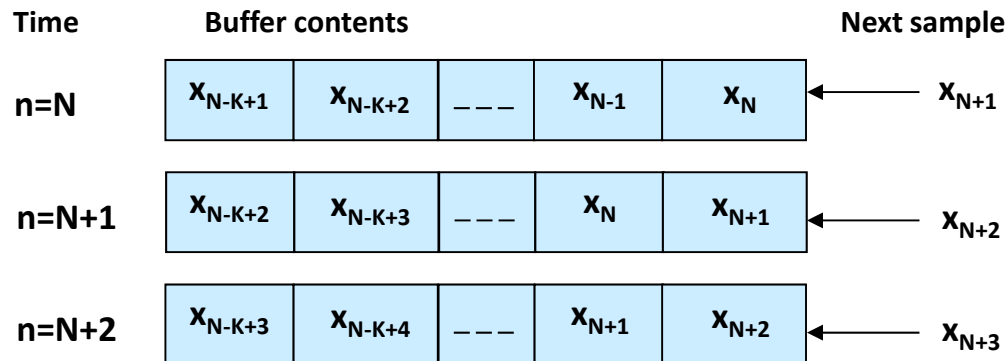
Operand addressing (cont)

■ Circular (Modulo) Addressing

- DSPs deal with streaming I/O
- Often interact with delay lines
- To save memory, buffer is often organized as a circular buffer
- To avoid overhead of address checking we keep a start register and end register per address register for use with auto-increment addressing, reset to start when reach end of buffer

Modulo Addressing

Time	Buffer contents						Next sample
n=N	x_{N-2}	x_{N-1}	x_N	x_{N-K+1}	x_{N-K+2}	....	x_{N+1}
n=N+1	x_{N-2}	x_{N-1}	x_N	x_{N+1}	x_{N-K+2}	x_{N-K+3}	x_{N+2}
n=N+2	x_{N-2}	x_{N-1}	x_N	x_{N+1}	x_{N+2}	x_{N-K+3} x_{N-K+4}	x_{N+3}
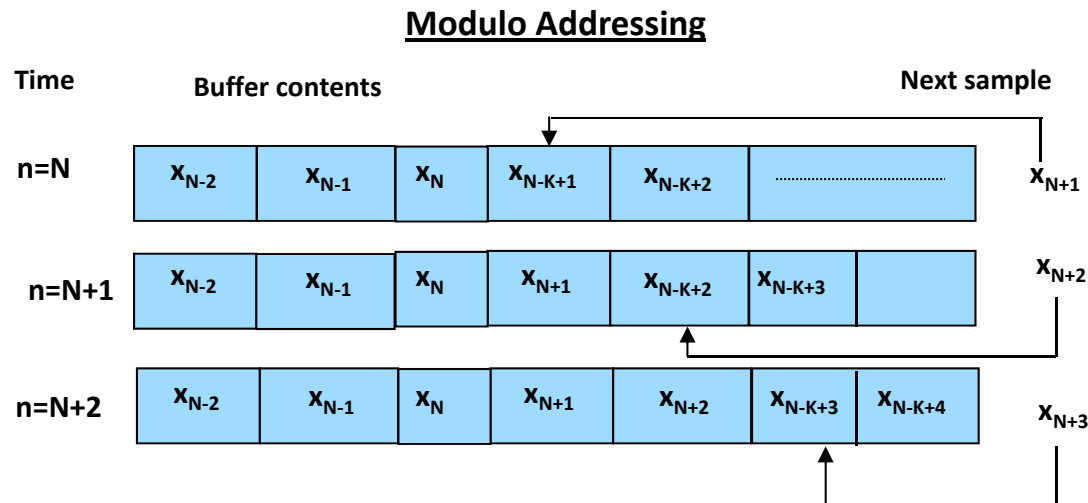
Operand addressing (cont)

- ## Bit reverse addressing
  - ### FFTs start or end with data in weird 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)

  - ### To avoid overhead of address checking instructions for FFT we us a "<u>bit reverse</u>" address addressing mode for use with auto-increment addressing
  - ### Use for radix-2 FFT

- ## Direct Memory Access Controller (DMAC)
  - ### Streaming data from I/O channels, etc.

Optimized Datapath Configurations

- **Support for Fixed Point (FXP) and Floating Point (FLP) data**

- **Fused Operations**
  - **Multiply-Accumulate**

- **Multiple Wide accumulators**
  - **Wider than datapath**
  - **Guard bits for precision and simplified scaling requirements**

- **Parallel Operations**
  - **VLIW - Multiple instructions execute in parallel**
  - **SIMD – Single instruction multiple data**

- **Special purpose shifters**
  - **Bit extraction**
  - **Scaling**

DSP Data Path: Arithmetic Operations

- **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.

$$-1 \leq x < 1$$

**radix
point**

S   .

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

**radix
point**

DSP Data Path: Precision

- **Word size affects precision of fixed point numbers**

- **DSPs have 16-bit, 20-bit, 24-bit, 32 bit data words**

- **DSP programmers will scale values inside code**
  - **SW Libraries**
  - **Separate explicit exponent**

- **Floating point support simplifies development**

- **Floating Point DSPs cost 2X - 4X vs. fixed point, slower than fixed point**

DSP Data Path: Overflow?

- **DSP are descended from analog computers: what should happen to output when you "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})**
  - Called "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**
  - **Requires a single cycle latency multiplier**
  - **Can be pipelined**

- **Need to perform multiply-accumulate (MAC)**
  - **n-bit multiplier => 2n-bit product**
  - **Accumulator is generally 1.5n wide**

DSP Data Path: Accumulator

- **Don't want overflow or have to scale accumulator**

- **Option 1: accumulator wider than product:**
  "guard bits"
  – **24b x 24b => 48b product, 56b Accumulator**

- **Option 2: shift right and round product before adder**
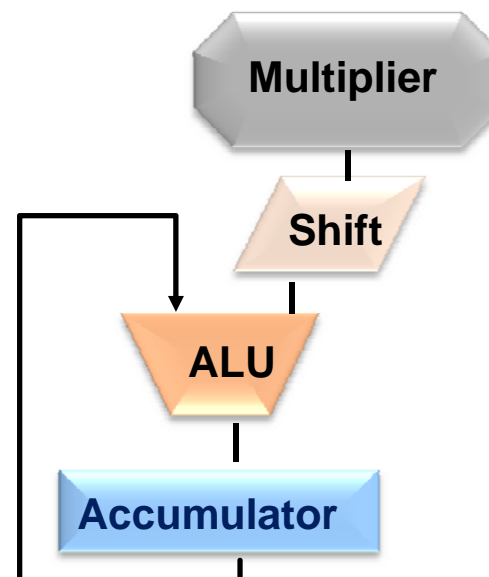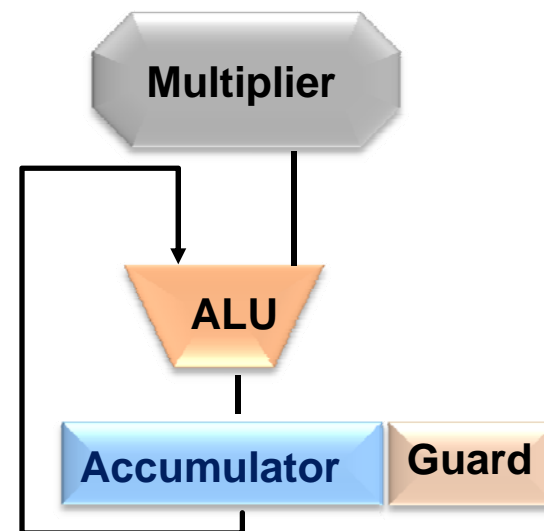
DSP Data Path: Rounding

▪ **Even with guard bits, will need to round when store accumulator into memory**

▪ **3 DSP standard options:**
  – <u>**Truncation**</u>**: chop results**
    => biases results up
  – <u>**Round to nearest**</u>**:**
    < 1/2 round down, >= 1/2 round up (more positive)
    => smaller bias
  – <u>**Convergent**</u>**:**
    < 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 <u>**round to nearest even**</u>

DSP Data Path: Multiple Instruction Units

- **VLIW Architectures Driving ILP**
- **Typical Instruction Units**
  - **M-Unit - MAC**
  - **S-Unit - Shift**
  - **L-Unit - ALU**
  - **D-Unit – Load/Store**
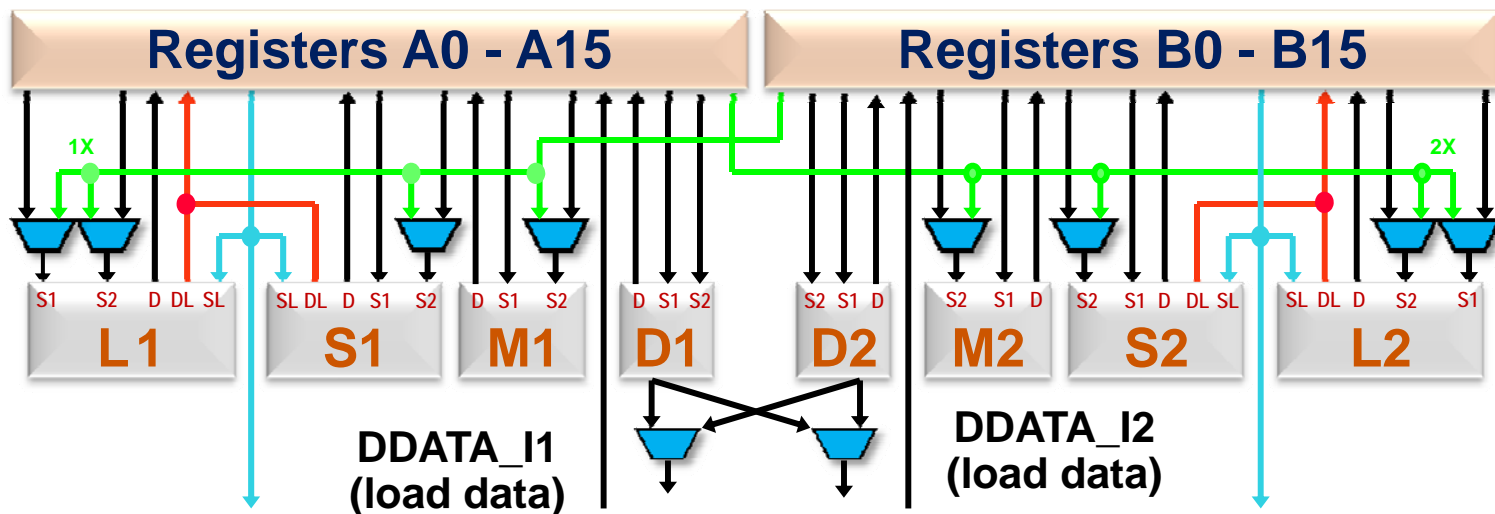
Specialized Instruction Sets

- **Base RISC ISA Plus CISC ISA Driven by End Application**
  - **MAC: Multiply Accumulate**
  - **SAD: Saturating Addition**
  - **LMS: Least Mean Squares**
  - **FIRS: Symmetrical FIR**
  - **Viterbi**

- **Support For Both Scalar and Vector Instructions**

- **Instructions can be highly orthogonal or variable width**

- **Implemented with FSM's or Microcode**

Interesting DSP Architectures

TI C64x DSP

Functional units:

6 ALUs

(L1, L2, S1, S2, D1, D2)

2 multipliers (M1, M2)

Program cache/program memory

32-bit addresses

256-bit data

TMS320C64x CPU

Program fetch

Instruction decode

Instruction dispatch

Register file A

Register file B

.L1 | .S1 | .M1 | .D1 | .D2 | .M2 | .S2 | .L2

Data cache/data memory

32-bit address

8-, 16-, 32-, 64- bit data
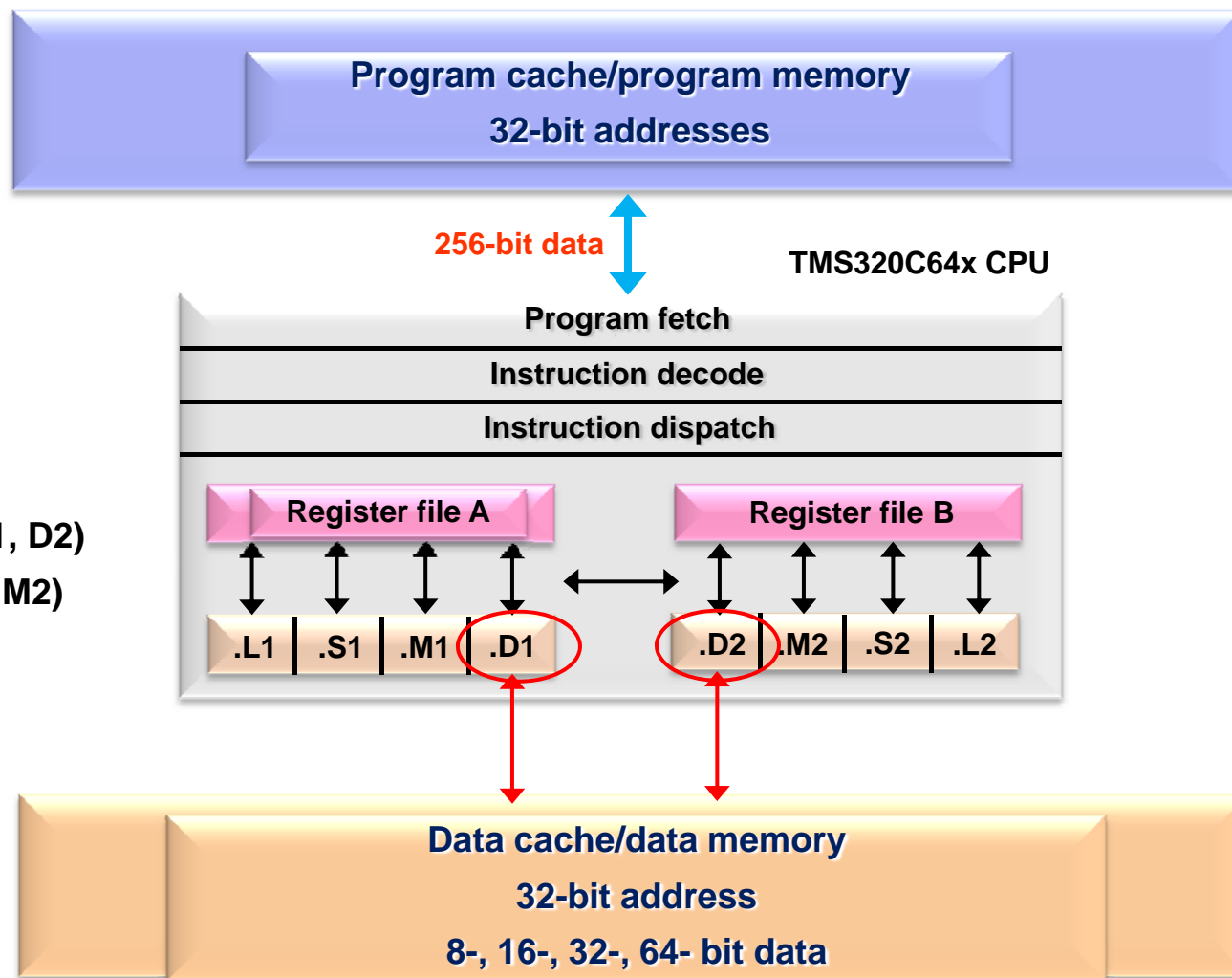
TI C64x DSP System-on-Chip

1064 MB/s — EMIF 64

266 MB/s — EMIF 16

12.5 MB/s — McBSP 0

12.5 MB/s — McBSP 1

12.5 MB/s — McBSP 2

133 MB/s — HPI32

100 MB/s — Utopia 2

Enhanced DMA Controller

L2 Cache/Memory

2.4 GB/s

9.6 GB/s

9.6 GB/s

L1P

19.2 GB/s

C64x™ DSP Core

2400 MMACS

9.6 GB/s

L1D

JTAG RTDX | Power Down Logic | PLL | Timer 0 | Timer 1 | Timer 2
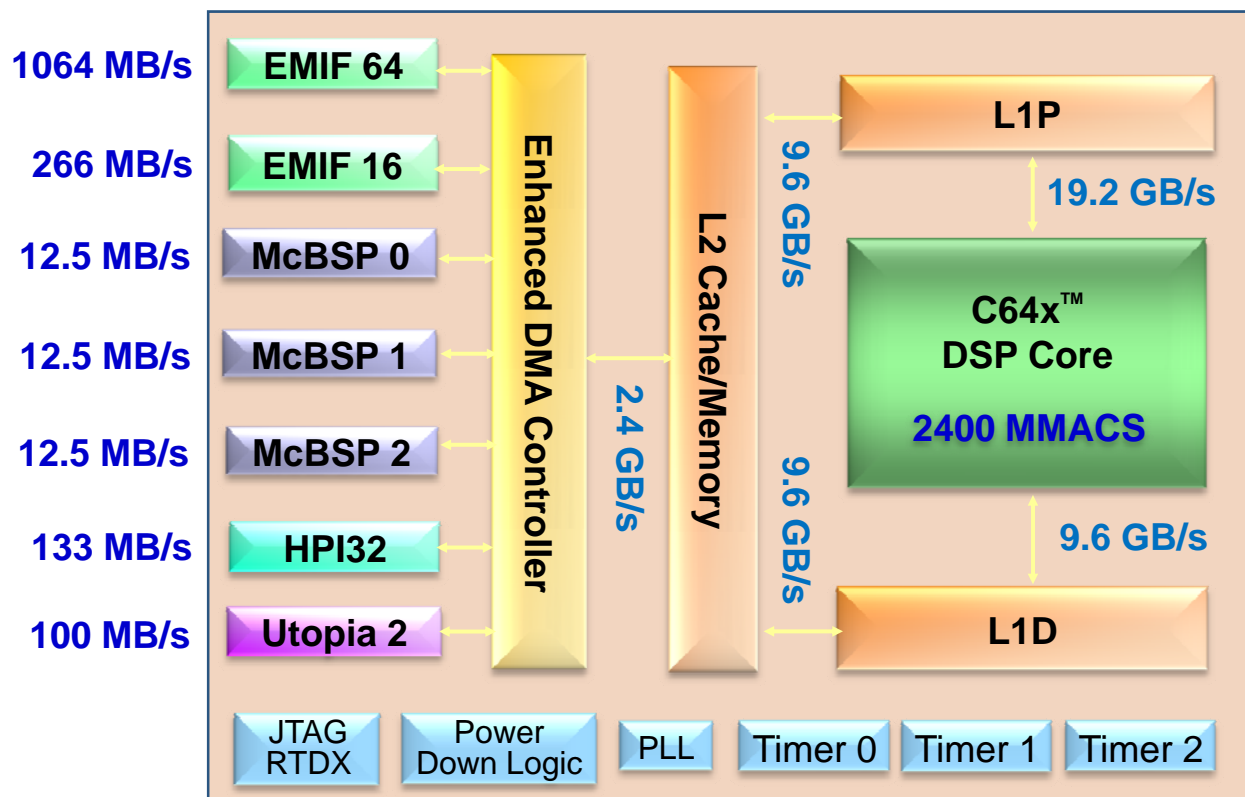
Performance:
2400 MMACS

Real-time multi-level memory architecture:
28.8 GB/s CPU Bandwidth

Concurrent, multi-threaded EDMA:
2.4 GB/s DMA bandwidth

High-speed I/O:
1.6 GB/s I/O bandwidth

C6X Main Features

- Performance of up to 4 billion instruction per second;
- Clock rate 500 MHZ;
- 2 register banks of 32 32-bit registers each;
- Program fetch, instruction dispatch (advanced instruction packing) and instruction decode units, which can supply 8 32-bits instructions to the functional units per cycle;
- Instructions are executed in 2 data path (A and B), each with four functional units (a multiplier and 3 ALUs) and a register bank

- The C64x register file contains 32 32-bit registers (A0-A31 for file A and B0-B31 for file B);
- GPRs can be used for data, pointers or conditions;
- Values larger than 32 bits (40-bit long and 64-bit float quantities) are stored in register pairs. Least significant bits are placed in an even-numbered register and the remaining bits (8 for 40-bit value and 32 for 32-bit value) are the next upper register;
- Packed data types are: four 8-bit values or two 16-bit values in a single 32-bit register, four 16-bit values in a 64-bit register pair.

Analog Devices SHARC ADSP-21061
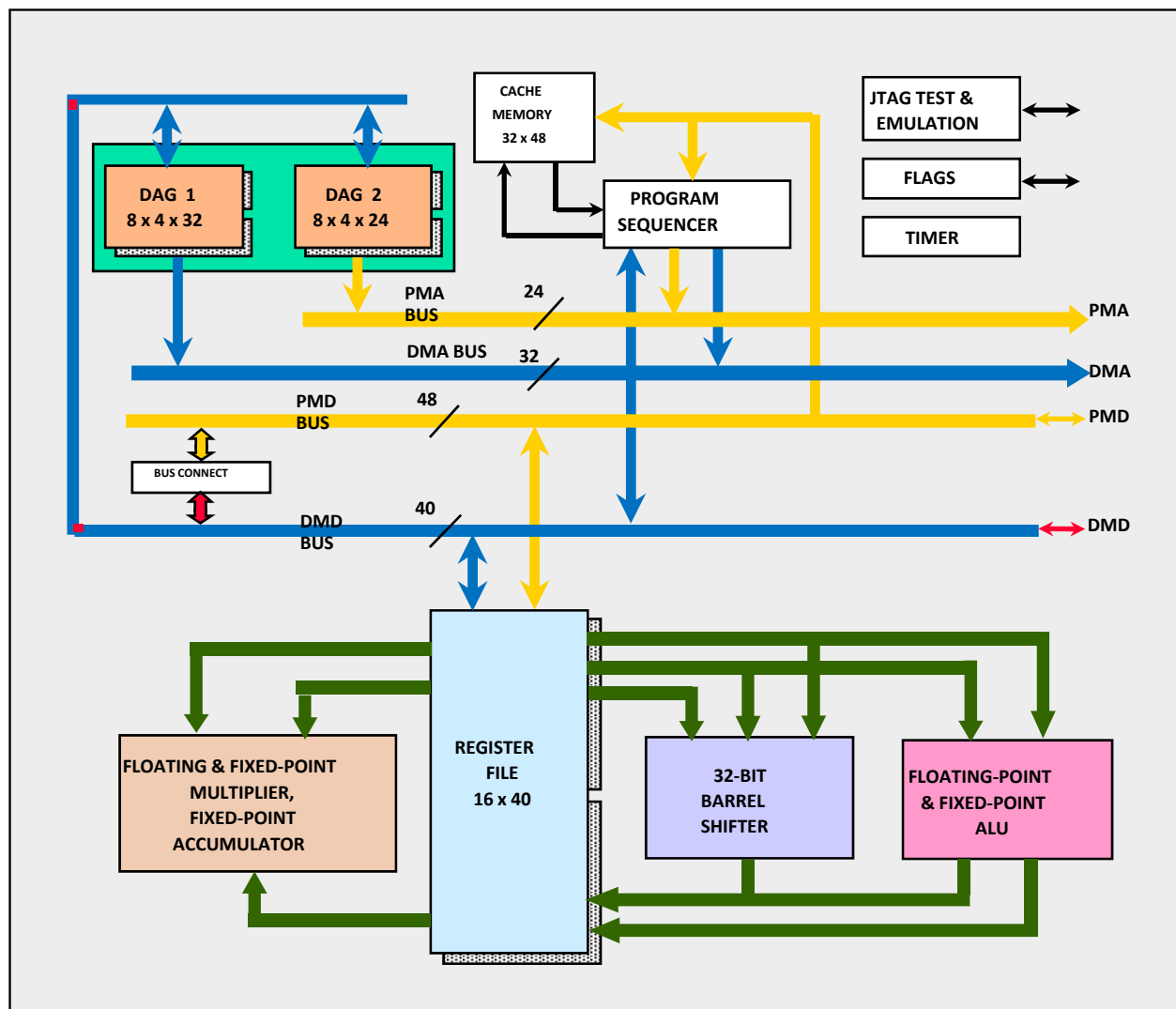
SHARC's Main Features

- **32/40-bit IEEE floating-point math**
- **32-bit fixed-point MACs with 64-bit product and 80-bit accumulation**
- **No arithmetic pipeline; Thus all computations are single-cycle**
- **Circular Buffer Addressing supported in hardware**
- **32 address pointers support 32 circular buffers**
- **16 48-bit Data Registers**

- **Six nested levels of zero-overhead looping in hardware**
- **Four busses to memory (2 DM + 2 PM)**
- **1 Mbit on-chip Dual Ported SRAM**
- **Maximum processing of 50 MIPS**
- **Possibility of four parallel operations processed in one clock cycle**
  - **+/-, *, DM, PM**
  - **Assuming Pipeline is full**
  - **PM clashing – utilize Instruction Cache**

Blackfin ADSP-215xx
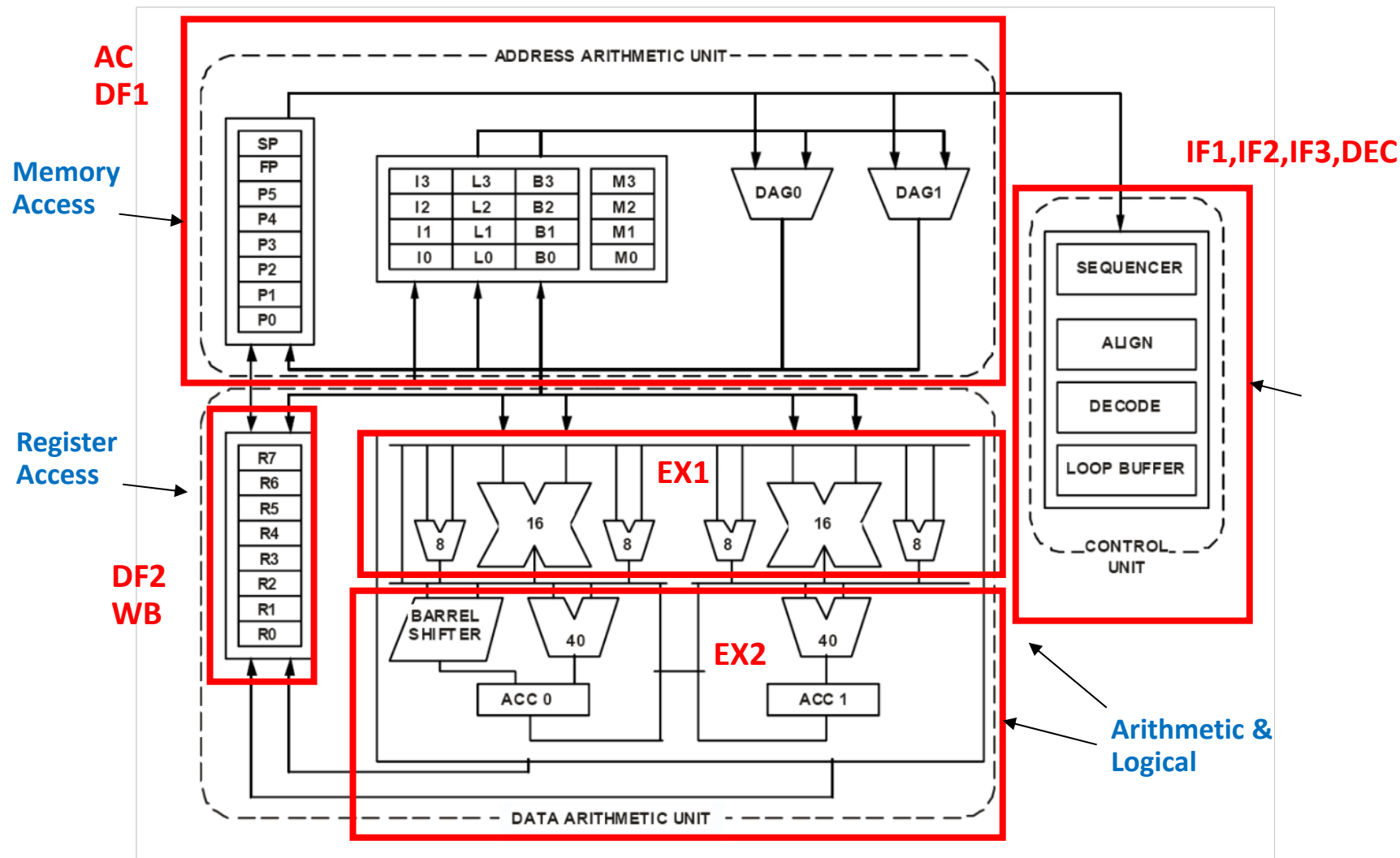
Blackfin's Main Features

- **Two 16-bit MACs, two 40-bit ALUs, and four 8-bit Video ALUs**
- **Support for 8/16/32-bit integer and 16/32-bit fractional data types**
- **Concurrent fetch of one instruction and two unique data elements**
- **Two loop counters that allow for nested zero-overhead looping**
- **Two DAG units with circular and bit-reversed addressing**
- **600 MHz core clock performing 600 MMACs**

- **Possibility of the following parallel operations processed in one clock cycle**
  - **Execution of a single instruction operating on both MACs or ALUs and**
  - **Execution of two 32-bit Data Moves (either 2 Reads or 1 Read/1 Write) and**
  - **Execution of two pointer updates and**
  - **Execution of hardware loop update**

Conventional DSP Processors Summary

	Fixed-Point	Floating-Point
Cost/Unit	$3 - $79	$3 - $381
Architecture	Accumulator	load-store or memory-register
Registers	2-4 data 8 address	8 or 16 data 8 or 16 address
Data Words	16 or 24 bit integer and fixed-point	32 bit integer and fixed/floating-point
On-Chip Memory	2-64 kwords data 2-64 kwords program	8-64 kwords data 8-64 kwords program
Address Space	16-128 kw data 16-64 kw program	16 Mw – 4Gw data 16 Mw – 4 Gw program
Compilers	C, C++ compilers; poor code generation	C, C++ compilers; better code generation
Examples	TI TMS320C5000; Motorola 56000	TI TMS320C30; Analog Devices SHARC