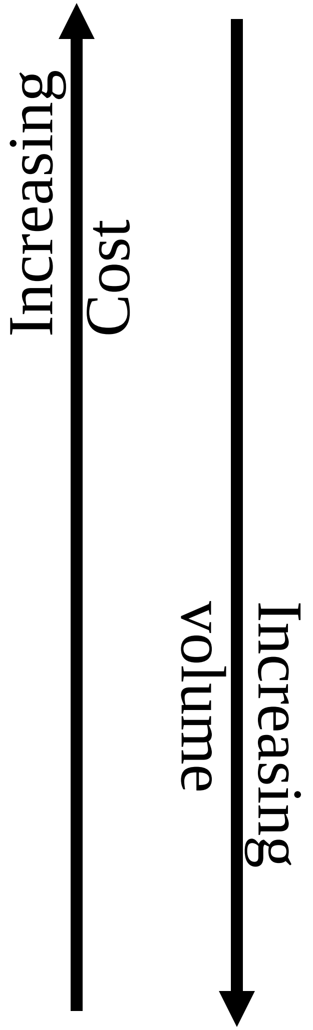
EE8591 DIGITAL SIGNAL PROCESSING

Digital Signal Processor (DSP) Architecture

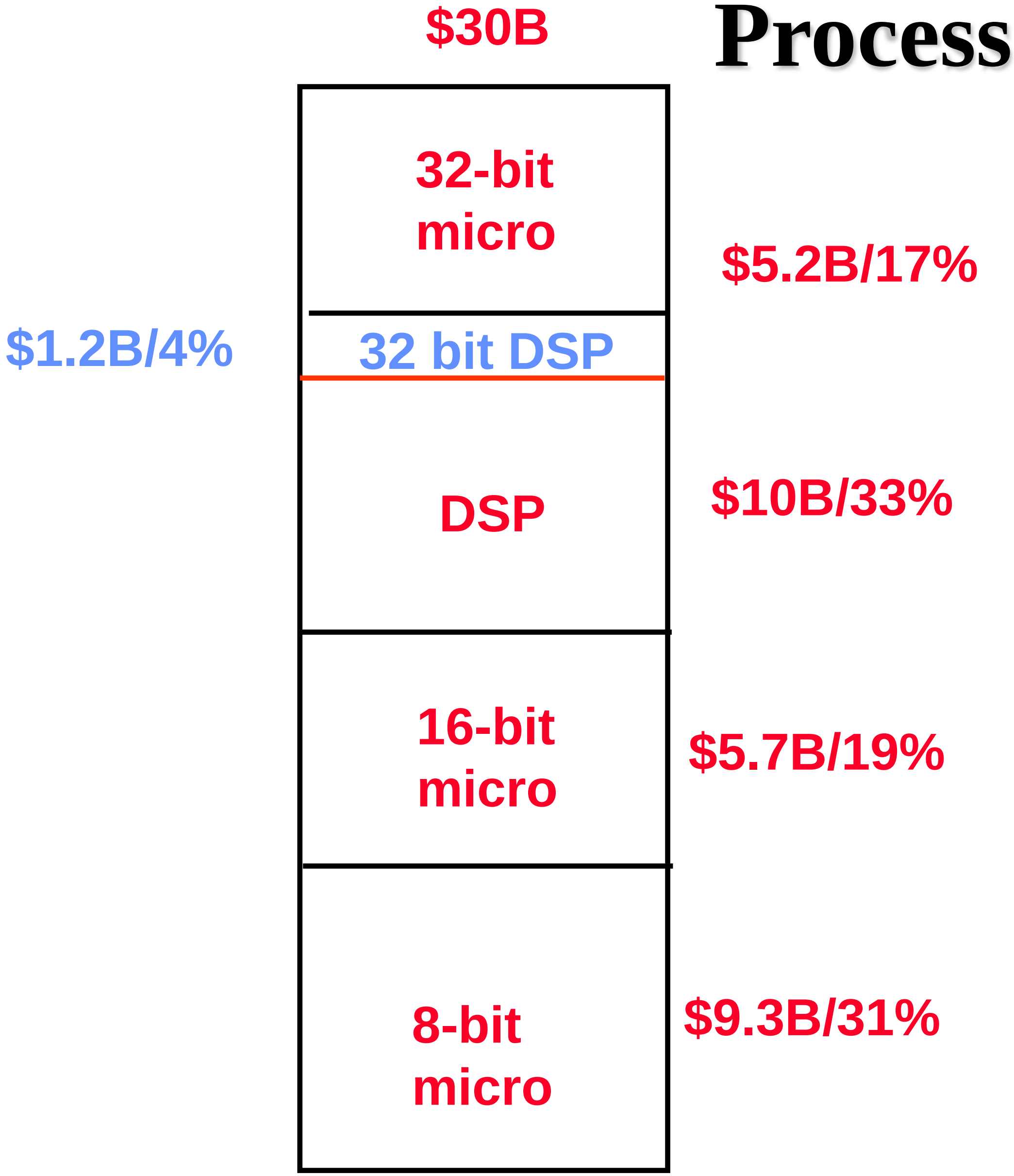
- Classification of Processor Applications
- Requirements of Embedded Processors
- DSP vs. General Purpose CPUs
- DSP Cores vs. Chips
- Classification of DSP Applications
- DSP Algorithm Format
- DSP Benchmarks
- Basic Architectural Features of DSPs
- DSP Software Development Considerations
- Classification 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 Processors (GPPs) - high performance.
  - Alpha’s, SPARC, MIPS ...
  - Used for general purpose software
  - Heavy weight OS - UNIX, Windows
  - Workstations, PC’s, Clusters
- 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
  - Control systems, Automobiles, toasters, thermostats, ...



Processor Markets



$30B

32-bit micro

$5.2B/17%

$1.2B/4%

32 bit DSP

DSP

$10B/33%

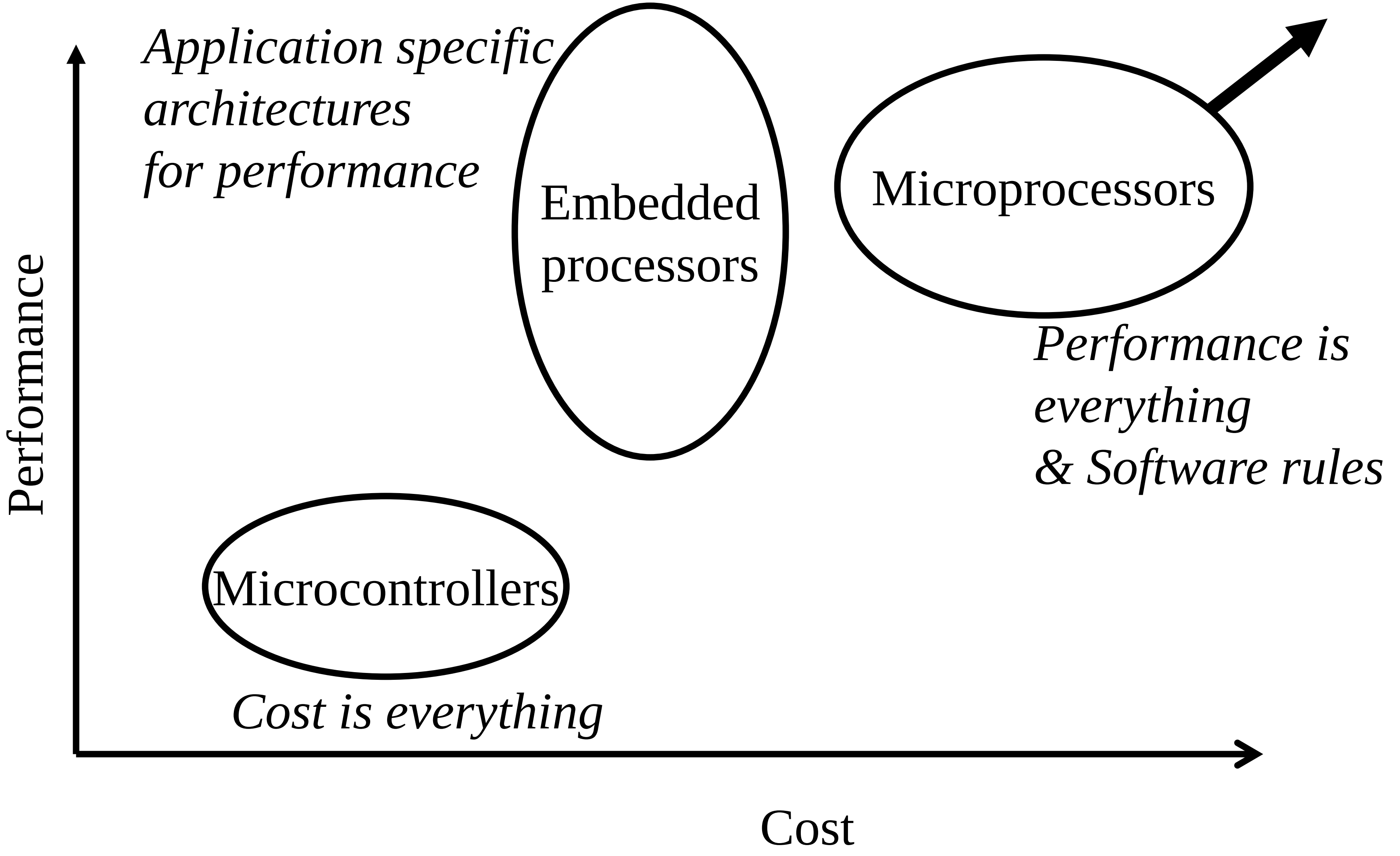
16-bit micro

$5.7B/19%

8-bit micro

$9.3B/31%

The Processor Design Space



Application specific architectures for performance

Embedded processors

Microprocessors

Performance is everything & Software rules

Microcontrollers

Cost is everything

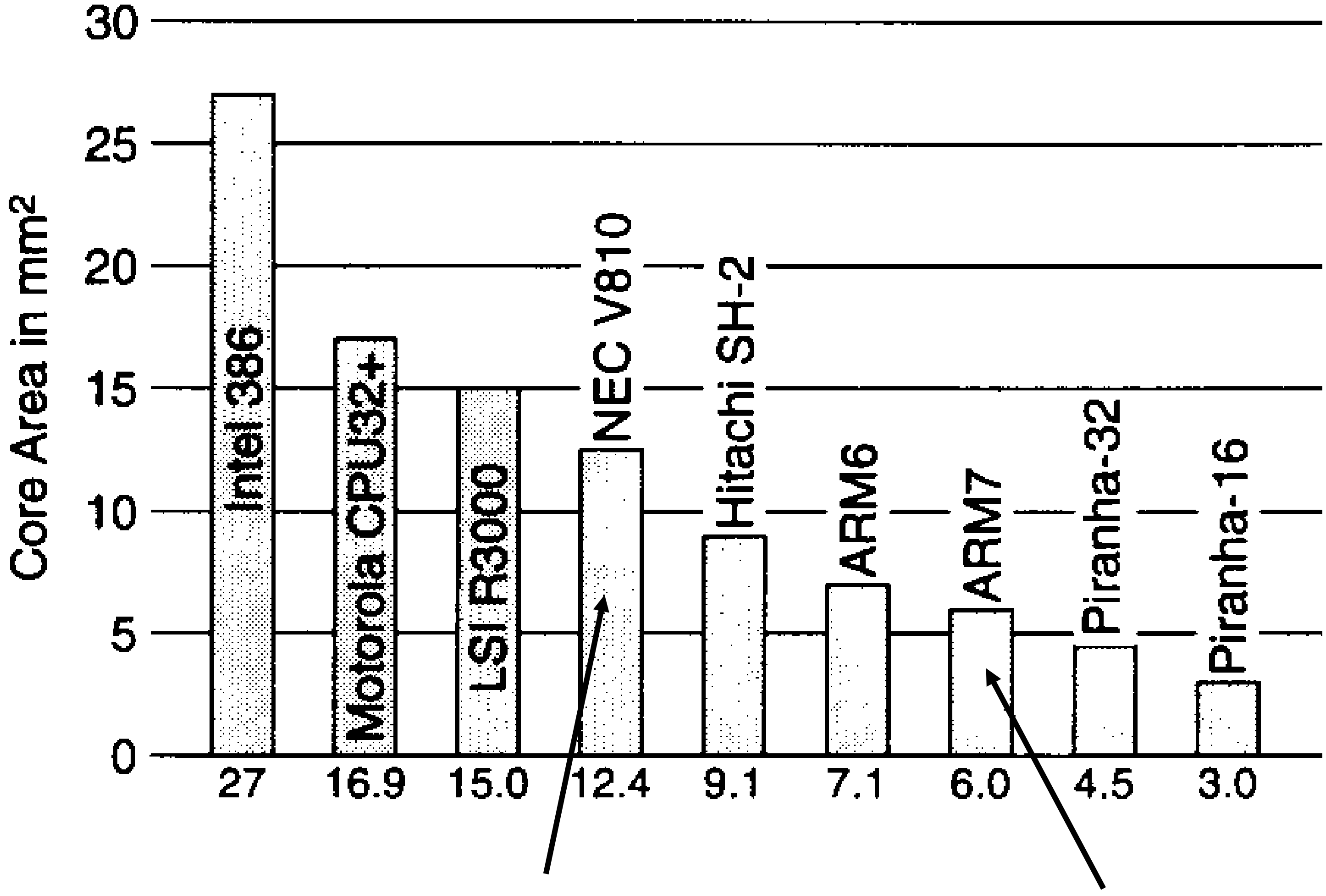
Performance

Cost

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 cores (System-on-Chip, SoC).
- Low power if application requires portability

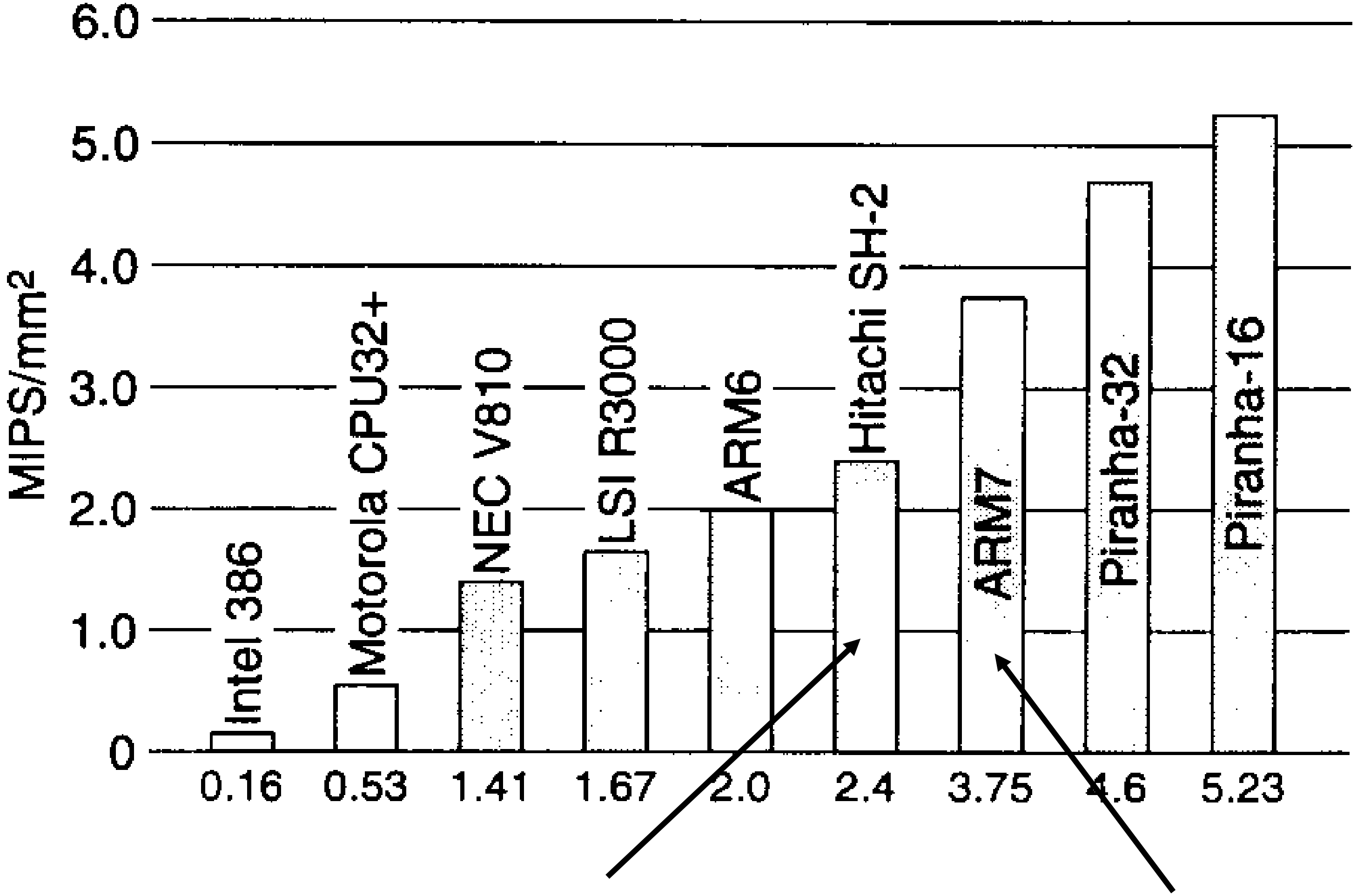
Area of processor cores = Cost



Nintendo processor

Cellular phones

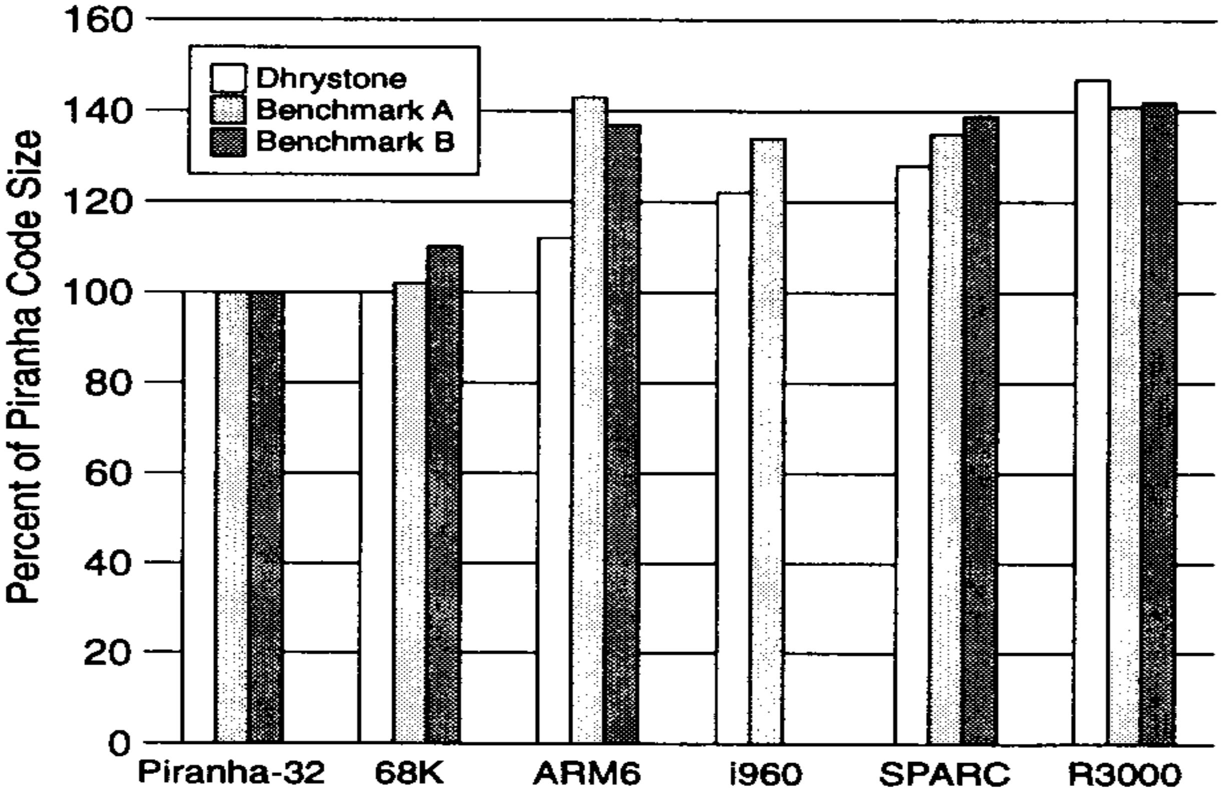
Another figure of merit: Computation per unit area



Nintendo processor

Cellular phones

Code size



- 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

Embedded System

- Runs a few applications often known at design time
- Not end-user programmable
- Operates in fixed run-time constraints that must be met, additional performance may not be useful/valuable
- Differentiating features:
  - Application-specific capability (e.g DSP).
  - 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)

Evolution of GPPs and DSPs

- **General Purpose Processors (GPPs) trace 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.**

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. GPP

- **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:**
  - ***Inifinite Impulse Response (IIR) filters***
  - ***Finite Impulse Response (FIR) filters***
  - ***FFT, and***
  - ***convolvers***
- **In DSPs, target algorithms are important:**
  - **Binary compatibility not a mojor 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.**

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 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. (SoC)**
  - **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**

DSP ARCHITECTURE
Enabling Technologies

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

Texas Instruments TMS320 Family
Multiple DSP μP Generations

	First Sample	Bit Size	Clock speed (MHz)	Instruction Throughput	MAC execution (ns)	MOPS	Device density (# of transistors)
Uniprocessor Based (Harvard Architecture)							
TMS32010	1982	16 integer	20	5 MIPS	400	5	58,000 (3μ)
TMS320C25	1985	16 integer	40	10 MIPS	100	20	160,000 (2μ)
TMS320C30	1988	32 flt.pt.	33	17 MIPS	60	33	695,000 (1μ)
TMS320C50	1991	16 integer	57	29 MIPS	35	60	1,000,000 (0.5μ)
TMS320C2XXX	1995	16 integer		40 MIPS	25	80	
Multiprocessor Based							
TMS320C80	1996	32 integer/flt.				2 GOPS 120 MFLOP	MIMD
TMS320C62XX	1997	16 integer		1600 MIPS	5	20 GOPS	VLIW
TMS310C67XX	1997	32 flt. pt.			5	1 GFLOP	VLIW

DSP Applications

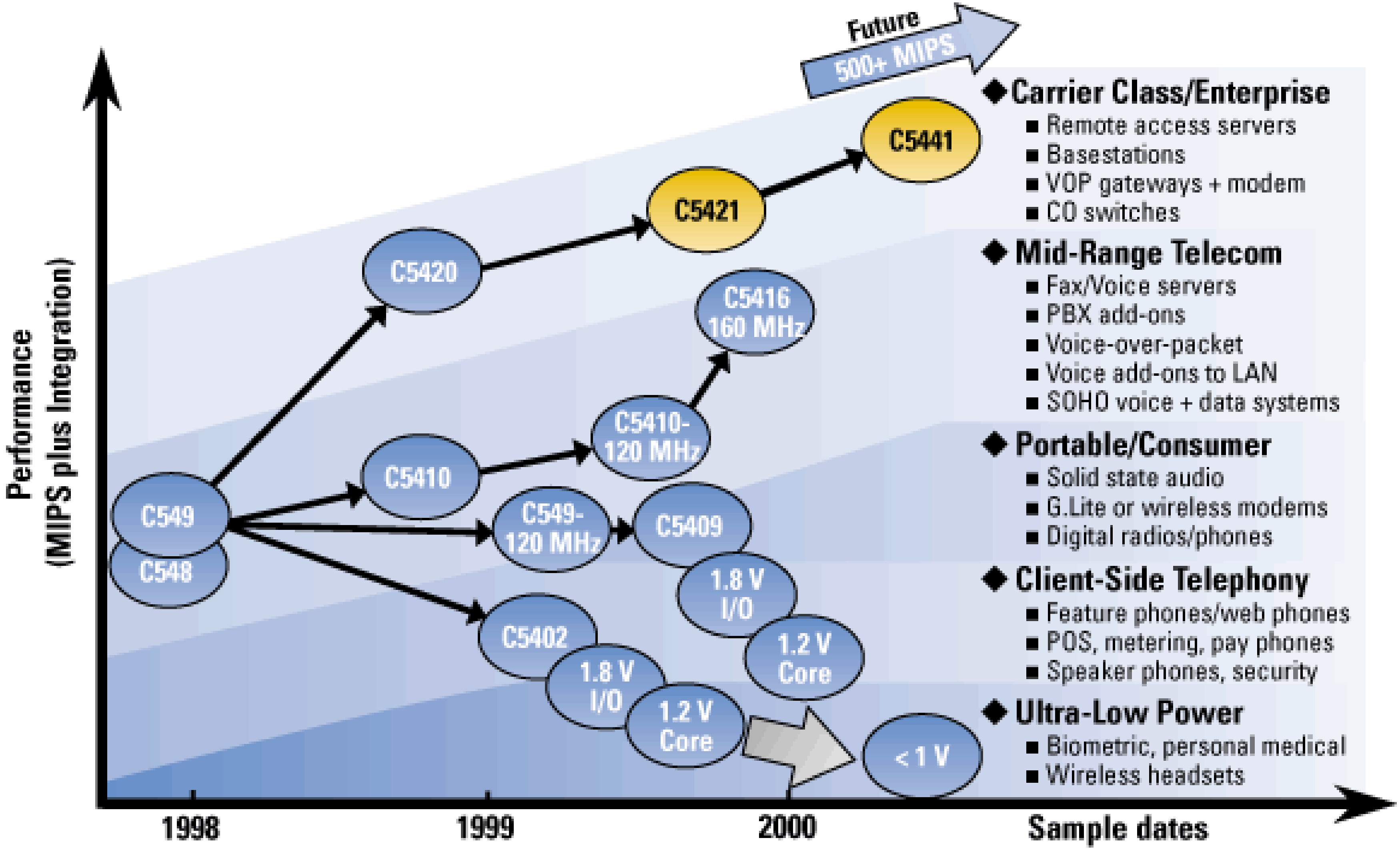
- Digital audio applications
  - MPEG Audio
  - Portable audio
- Digital cameras
- Cellular telephones
- Wearable medical appliances
- Storage products:
  - disk drive servo control
- Military applications:
  - radar
  - sonar

- Industrial control
- Seismic exploration
- Networking:
  - Wireless
  - Base station
  - Cable modems
  - ADSL
  - VDSL

DSP Applications

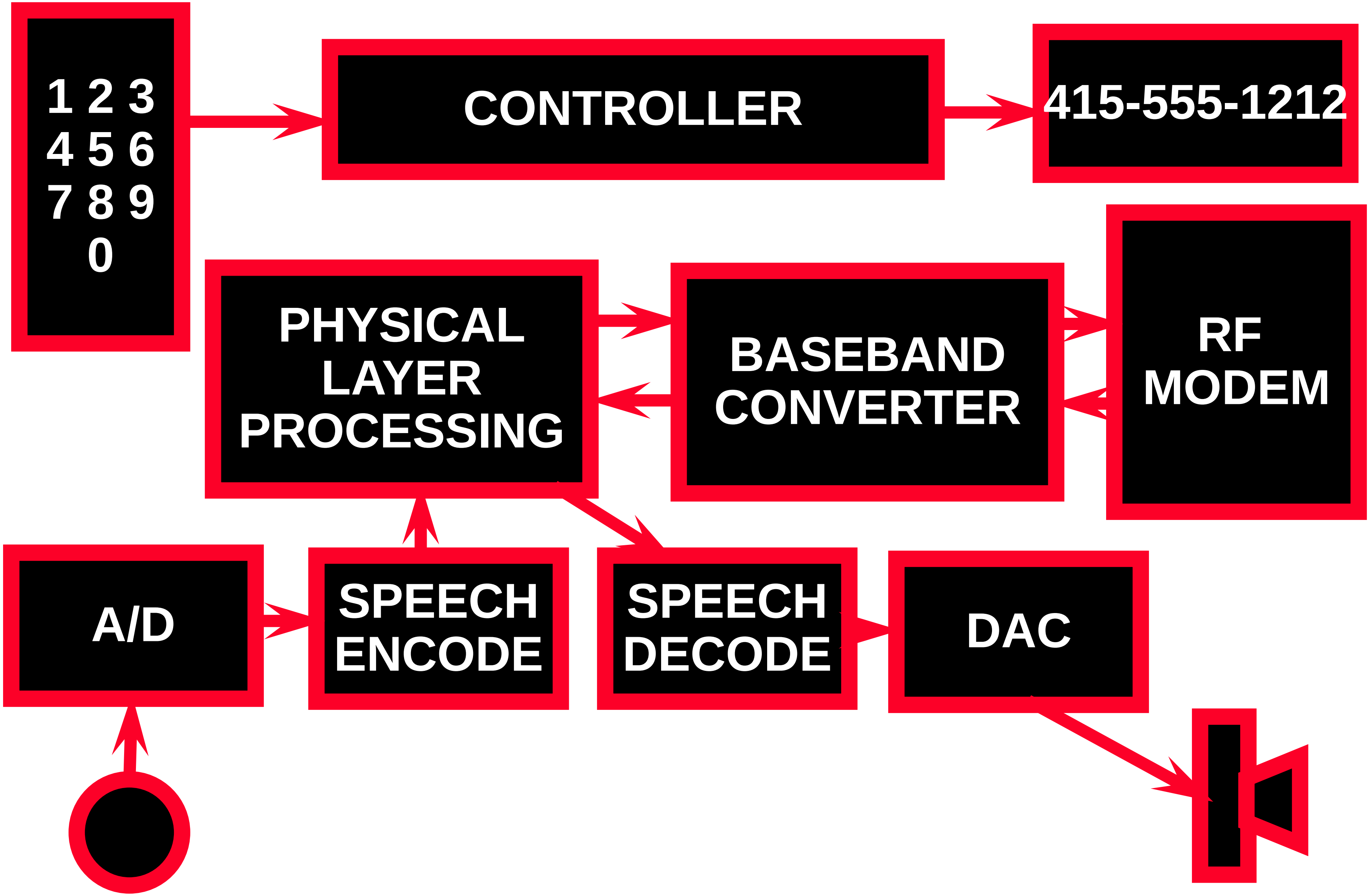
DSP Algorithm	System Application
Speech Coding	Digital cellular telephones, personal communications systems, digital cordless telephones, multimedia computers, secure communications.
Speech Encryption	Digital cellular telephones, personal communications systems, digital cordless telephones, secure communications.
Speech Recognition	Advanced user interfaces, multimedia workstations, robotics, automotive applications, cellular telephones, personal communications systems.
Speech Synthesis	Advanced user interfaces, robotics
Speaker Identification	Security, multimedia workstations, advanced user interfaces
High-fidelity Audio	Consumer audio, consumer video, digital audio broadcast, professional audio, multimedia computers
Modems	Digital cellular telephones, personal communications systems, digital cordless telephones, digital audio broadcast, digital signaling on cable TV, multimedia computers, wireless computing, navigation, data/fax
Noise cancellation	Professional audio, advanced vehicular audio, industrial applications
Audio Equalization	Consumer audio, professional audio, advanced vehicular audio, music
Ambient Acoustics Emulation	Consumer audio, professional audio, advanced vehicular audio, music
Audio Mixing/Editing	Professional audio, music, multimedia computers
Sound Synthesis	Professional audio, music, multimedia computers, advanced user interfaces
Vision	Security, multimedia computers, advanced user interfaces, instrumentation, robotics, navigation
Image Compression	Digital photography, digital video, multimedia computers, videoconferencing
Image Compositing	Multimedia computers, consumer video, advanced user interfaces, navigation
Beamforming	Navigation, medical imaging, radar/sonar, signals intelligence
Echo cancellation	Speakerphones, hands-free cellular telephones
Spectral Estimation	Signals intelligence, radar/sonar, professional audio, music

DSP range of applications

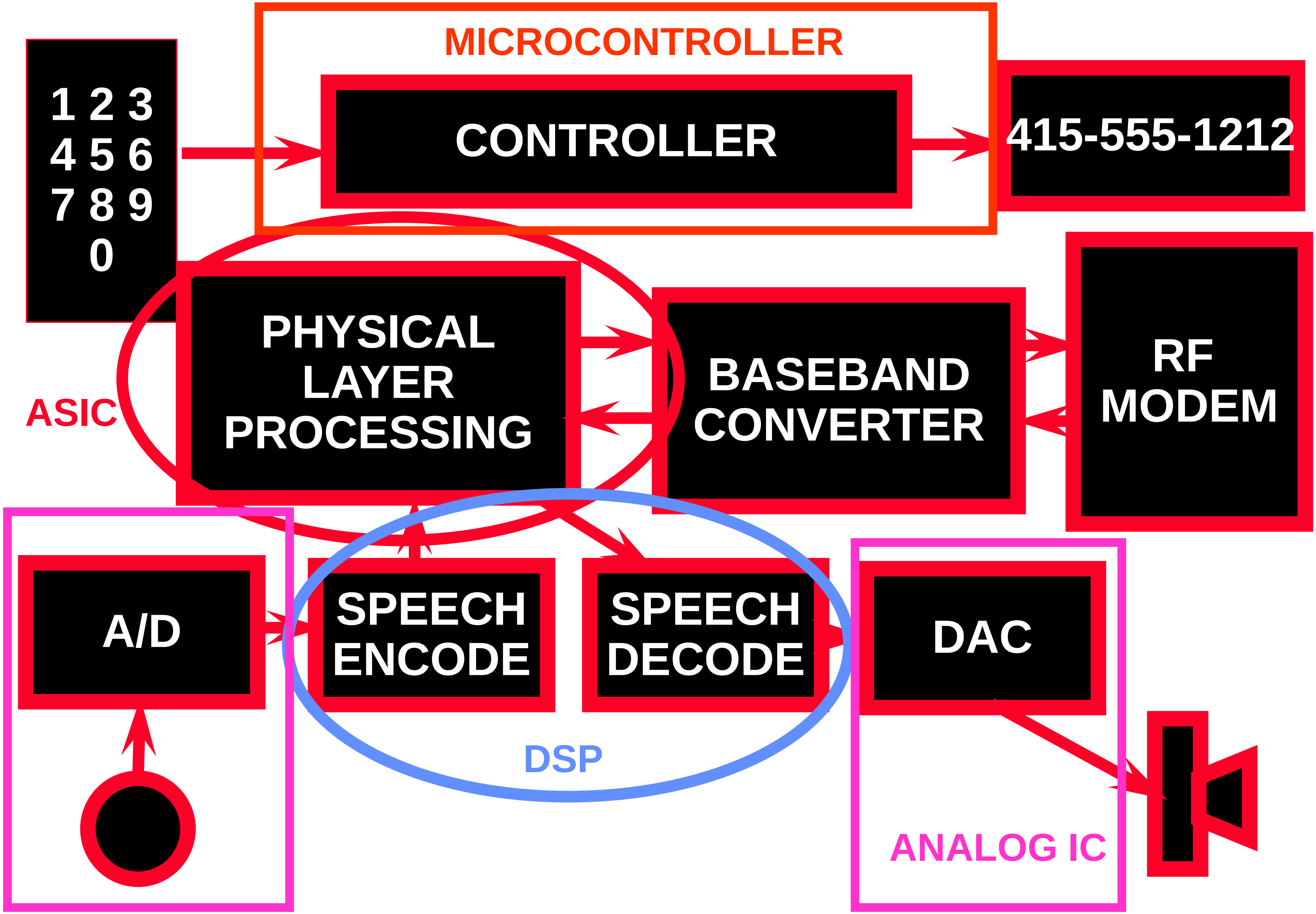


- Carrier Class/Enterprise
  - Remote access servers
  - Basestations
  - VOP gateways + modem
  - CO switches
- Mid-Range Telecom
  - Fax/Voice servers
  - PBX add-ons
  - Voice-over-packet
  - Voice add-ons to LAN
  - SOHO voice + data systems
- Portable/Consumer
  - Solid state audio
  - G.Lite or 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

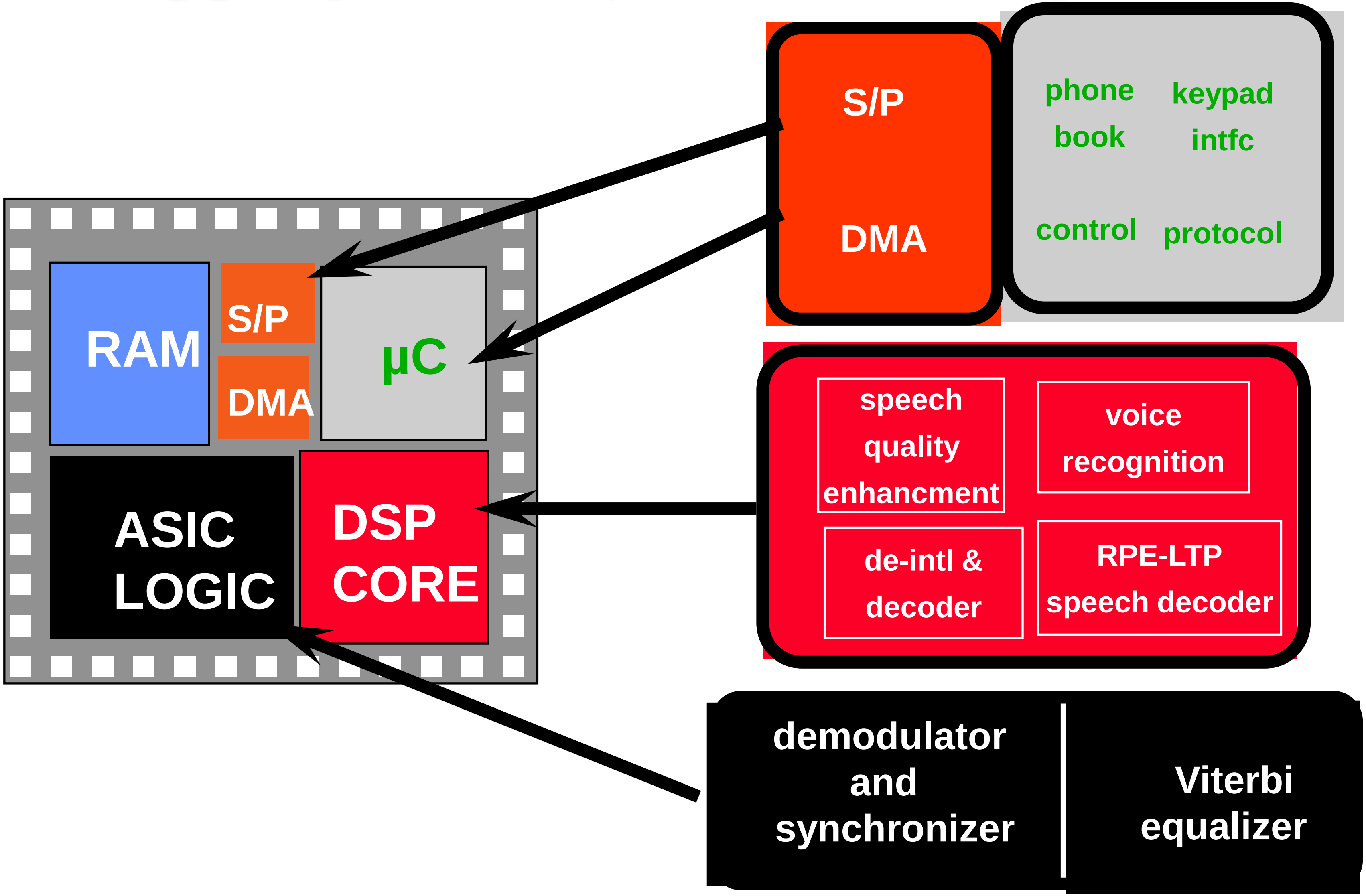
CELLULAR TELEPHONE SYSTEM



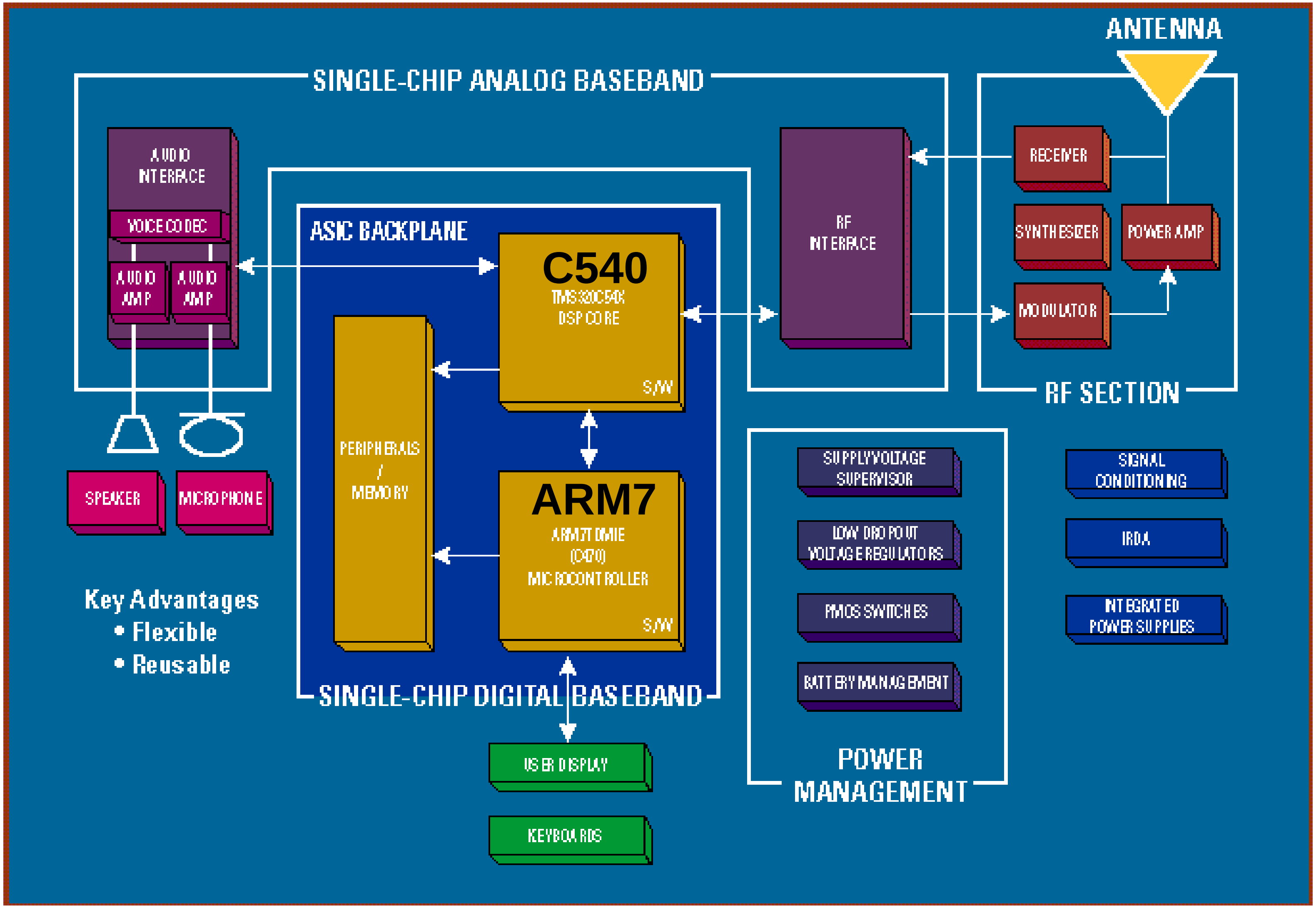
HW/SW/IC PARTITIONING



Mapping Onto System-on-Chip (SoC)



Example Wireless Phone Organization



ANTENNA

SINGLE-CHIP ANALOG BASEBAND

AUDIO INTERFACE

VOICE CODEC

AUDIO AMP

AUDIO AMP

SPEAKER

MICROPHONE

ASIC BACKPLANE

C540

TMS320C54X DSP CORE

S/W

PERIPHERALS / MEMORY

ARM7

ARM7TDMIE (C470) MICROCONTROLLER

S/W

SINGLE-CHIP DIGITAL BASEBAND

USER DISPLAY

KEYBOARDS

RF INTERFACE

RECEIVER

SYNTHESIZER

POWER AMP

MODULATOR

RF SECTION

SUPPLY VOLTAGE SUPERVISOR

LOW DROPOUT VOLTAGE REGULATORS

PMOS SWITCHES

BATTERY MANAGEMENT

POWER MANAGEMENT

SIGNAL CONDITIONING

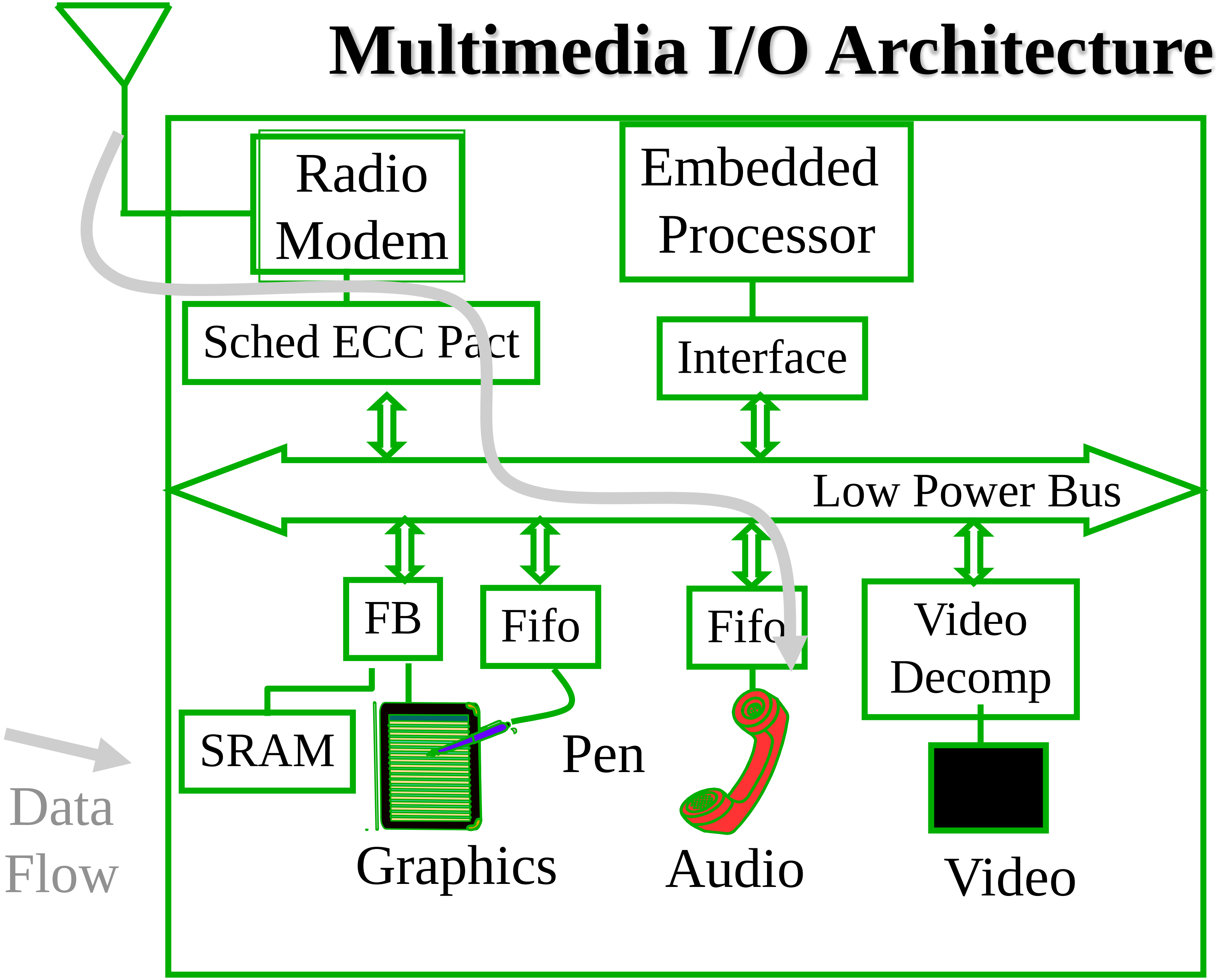
IRDA

INTEGRATED POWER SUPPLIES

Key Advantages

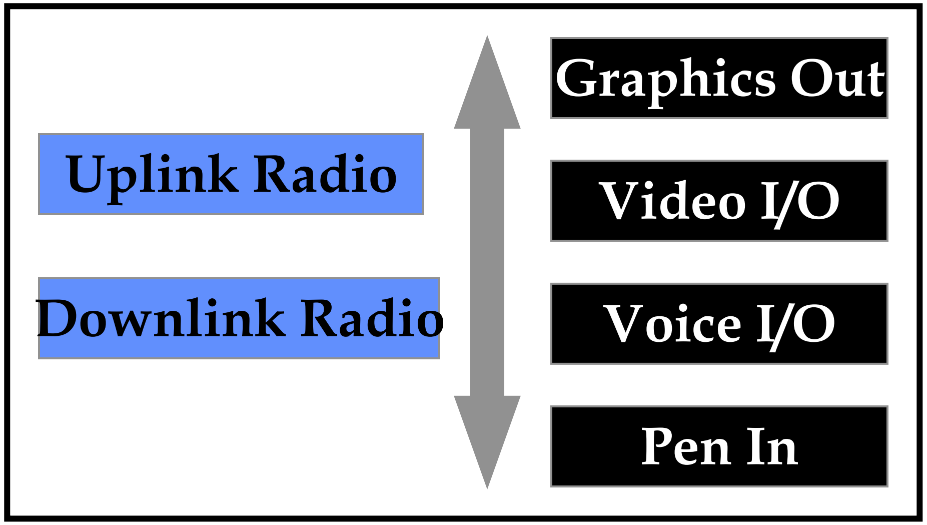
- Flexible
- Reusable

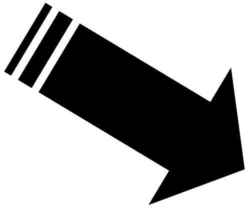
Multimedia I/O Architecture



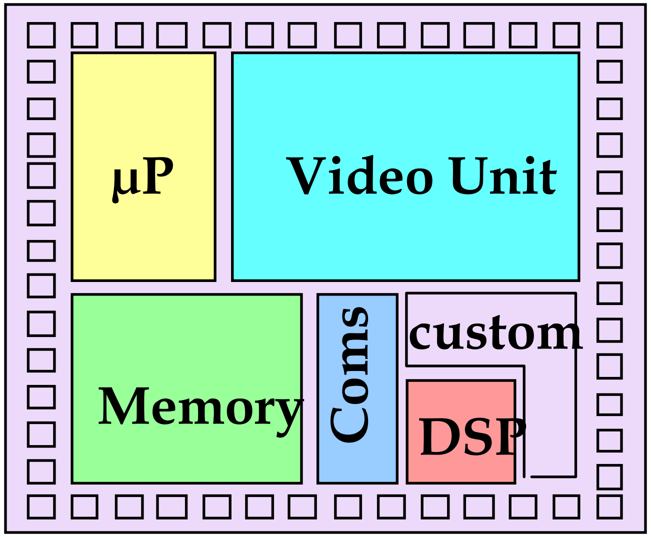
Multimedia System-on-Chip (SoC)

E.g. Multimedia terminal electronics





- **Future chips will be a mix of processors, memory and dedicated hardware for specific algorithms and I/O**



DSP Algorithm Format

- 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

DSP Algorithm Notation

- Uses “flowchart” notation instead of equations
- Multiply is X or ▷
- Add is + or Σ
- Delay/Storage is or or

Delay z^{-1} D

Typical DSP Algorithm:
Finite-Impulse Response (FIR) Filter

- **Filters reduce signal noise and enhance image or signal quality by removing unwanted frequencies.**
- **Finite Impulse Response (FIR) filters compute:**

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

where

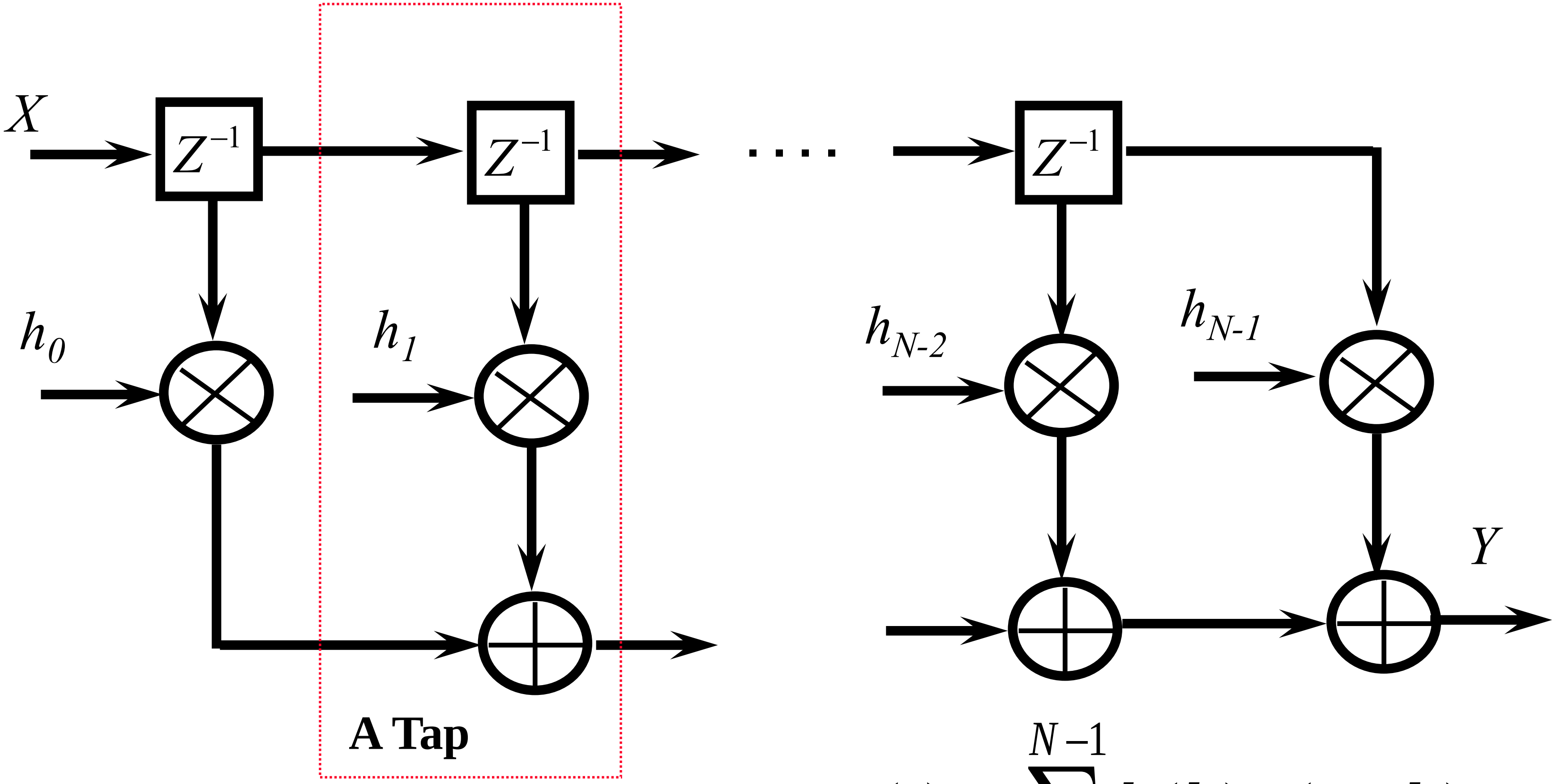
- – **x is the input sequence**
- – **y is the output sequence**
- – **h is the impulse response (filter coefficients)**
- – **N is the number of taps (coefficients) in the filter**

- **Output sequence depends only on input sequence and impulse response.**

Typical DSP Algorithm:
Finite-impulse Response (FIR) Filter

- **N most recent samples in the delay line (Xi)**
- **New sample moves data down delay line**
- **“Tap” is a multiply-add**
- **Each tap (N 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



$$y(i) = \sum_{k=0}^{N-1} h(k)\,x(i-k)$$

Goal: at least 1 FIR Tap / DSP instruction cycle

Sample Computational Rates for FIR Filtering

Signal type	Frequency	# taps	Performance
Speech	8 kHz	N =128	20 MOPs
Music	48 kHz	N =256	24 MOPs
Video phone	6.75 MHz	N*N = 81	1,090 MOPs
TV	27 MHz	N*N = 81	4,370 MOPs
HDTV	144 MHz	N*N = 81	23,300 MOPs

1-D FIR has $n_{op} = 2N$ and a 2-D FIR has $n_{op} = 2N^2$.

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

Typical DSP Algorithm:
Infinite-Impulse Response (IIR) Filter

- Infinite Impulse Response (IIR) filters compute:

$$y(i) = \sum_{k=1}^{M-1} a(k)y(i-k) + \sum_{k=0}^{N-1} b(k)x(i-k)$$

- Output sequence depends on input sequence, previous outputs, and impulse response.
- Both FIR and IIR filters
  - Require dot product (multiply-accumulate) operations
  - Use fixed coefficients
- Adaptive filters update their coefficients to minimize the distance between the filter output and the desired signal.

Typical DSP Algorithm:

Discrete Fourier Transform

- The **Discrete Fourier Transform** (DFT) allows for spectral analysis in the frequency domain.
- It is computed as

$$y(k) = \sum_{n=0}^{N-1} W_N{}^{nk} x(n) \qquad W_N = e^{\frac{-2j\pi}{N}} \qquad j = \sqrt{-1}$$

for k = 0, 1, … , N-1, where

  - x is the input sequence in the time domain
  - y is an output sequence in the frequency domain

- The **Inverse Discrete Fourier Transform** is computed as

$$x(n) = \sum_{k=0}^{N-1} W_N{}^{-nk} y(k), \quad \text{for } \ \mathrm{n} = 0, 1, \ldots, \mathrm{n} - 1$$

- The **Fast Fourier Transform** (FFT) provides an efficient method for computing the DFT.

Typical DSP Algorithm:
Discrete Cosine Transform (DCT)

- The **Discrete Cosine Transform** (DCT) is frequently used in video compression (e.g., MPEG-2).
- The DCT and **Inverse DCT** (IDCT) are computed as:

$$y(k) = e(k)\sum_{n=0}^{N-1}\cos[\frac{(2n+1)k\pi}{2N}]x(n), \quad \text{for} \;\; \mathrm{k} = 0, 1, \ldots \mathrm{N} - 1$$

$$x(n) = \frac{2}{N}\sum_{k=0}^{N-1}e(k)\cos[\frac{(2n+1)k\pi}{2N}]y(n), \quad \text{for} \;\; \mathrm{k} = 0, 1, \ldots \mathrm{N} - 1$$

where *e(k) = 1/sqrt(2)* if *k = 0*; otherwise *e(k) = 1*.

- A *N*-Point, 1D-DCT requires N^2 **MAC** operations.

DSP BENCHMARKS

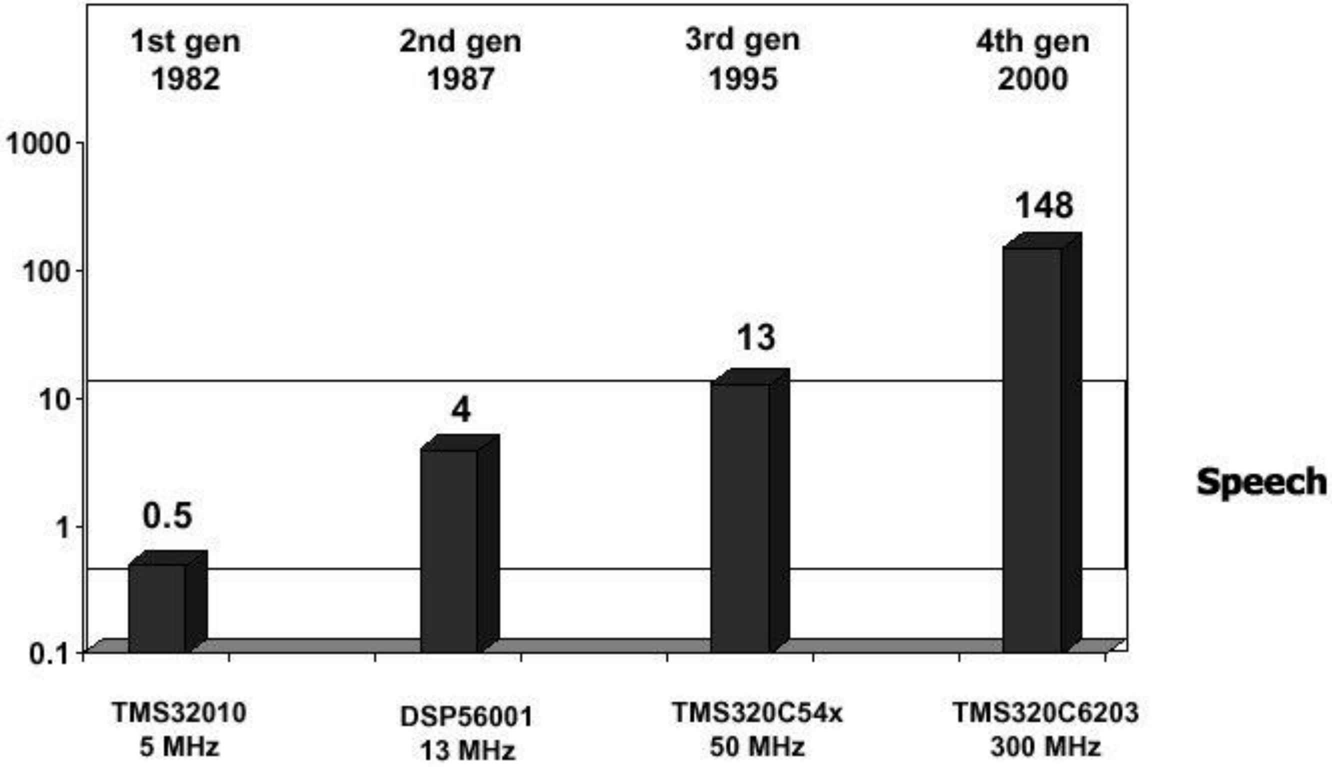
- **DSPstone: University of Aachen, application benchmarks**
  - ADPCM TRANSCODER - CCITT G.721, REAL_UPDATE, COMPLEX_UPDATES
  - DOT_PRODUCT, MATRIX_1X3, CONVOLUTION
  - FIR, FIR2DIM, HR_ONE_BIQUAD
  - LMS, FFT_INPUT_SCALED

- **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)

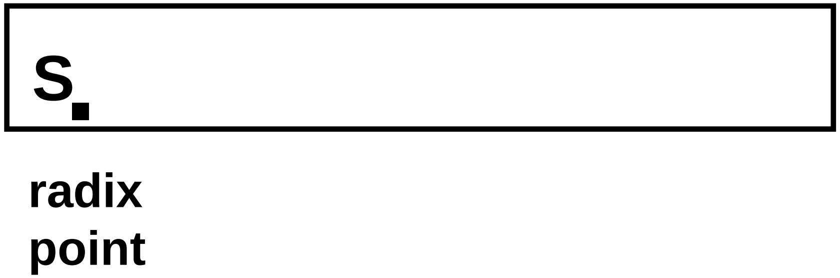


Basic Architectural Features of DSPs

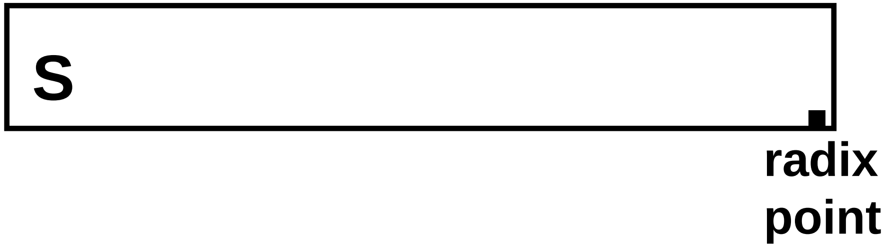
- Data path configured for DSP
  - Fixed-point arithmetic
  - MAC- Multiply-accumulate
- Multiple memory banks and buses -
  - Harvard Architecture
  - Multiple data memories
- Specialized addressing modes
  - Bit-reversed addressing
  - Circular buffers
- Specialized instruction set and execution control
  - Zero-overhead loops
  - Support for fast MAC
  - Fast Interrupt Handling
- 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 Š $x < 1$



$–2^{N–1}$ Š $x <$ $2^{N–1}$

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

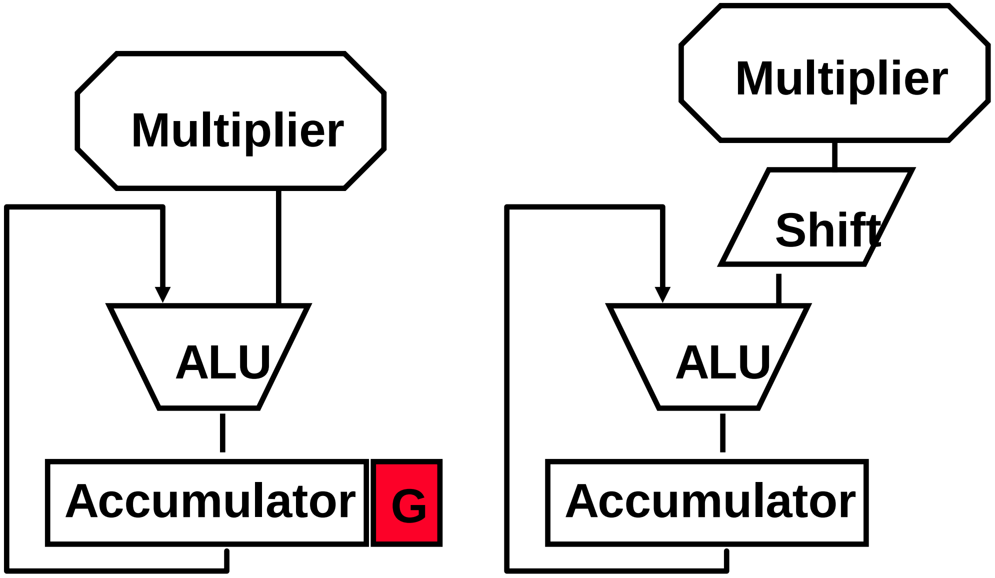
- **DSP are descended from analog :**
  - **Modulo Arithmetic.**
- **Set to most positive ($2^{N-1}-1$) or most negative value(-2^{N-1}) : "saturation"**
- **Many DSP algorithms were developed in this model.**

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: accumalator 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

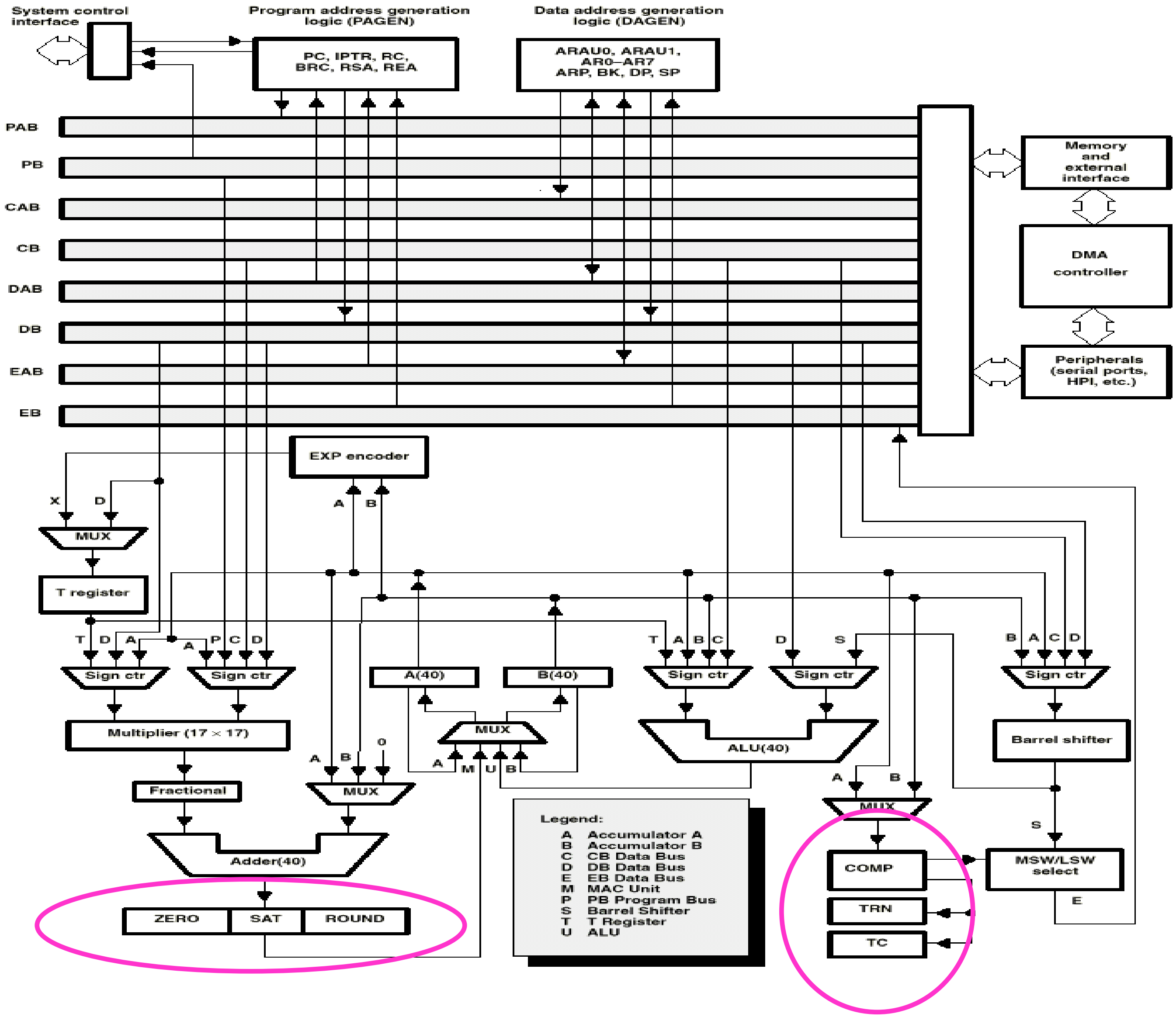
DSP Processor

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

General-Purpose Processor

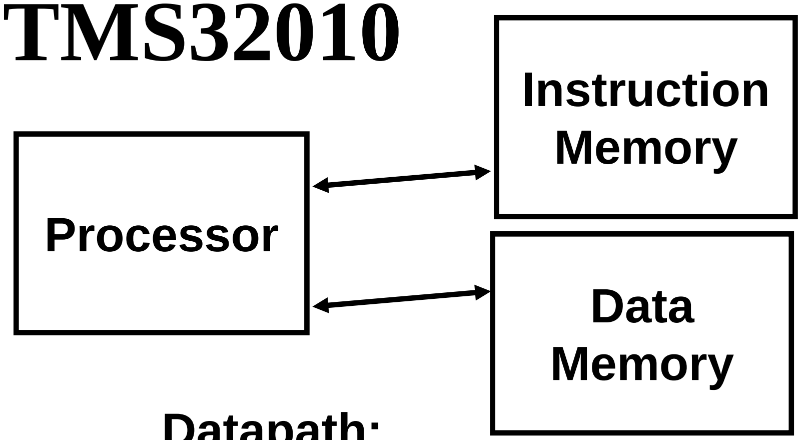
- Multiplies often take>1 cycle
- Shifts often take >1 cycle
- Other operations (e.g., saturation, rounding) typically take multiple cycles.

TI 320C54x DSP (1995) Functional Block Diagram

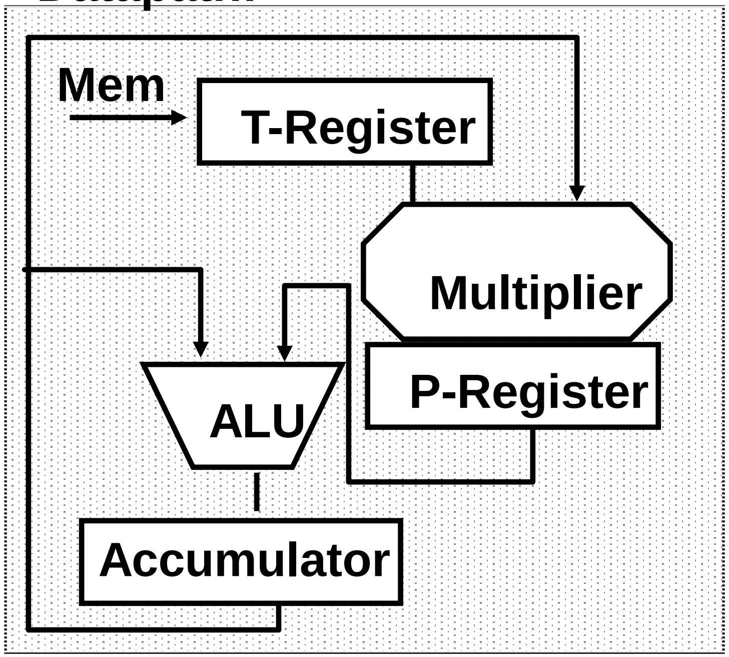


First Commercial DSP (1982): Texas Instruments TMS32010

- 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.



Datapath:



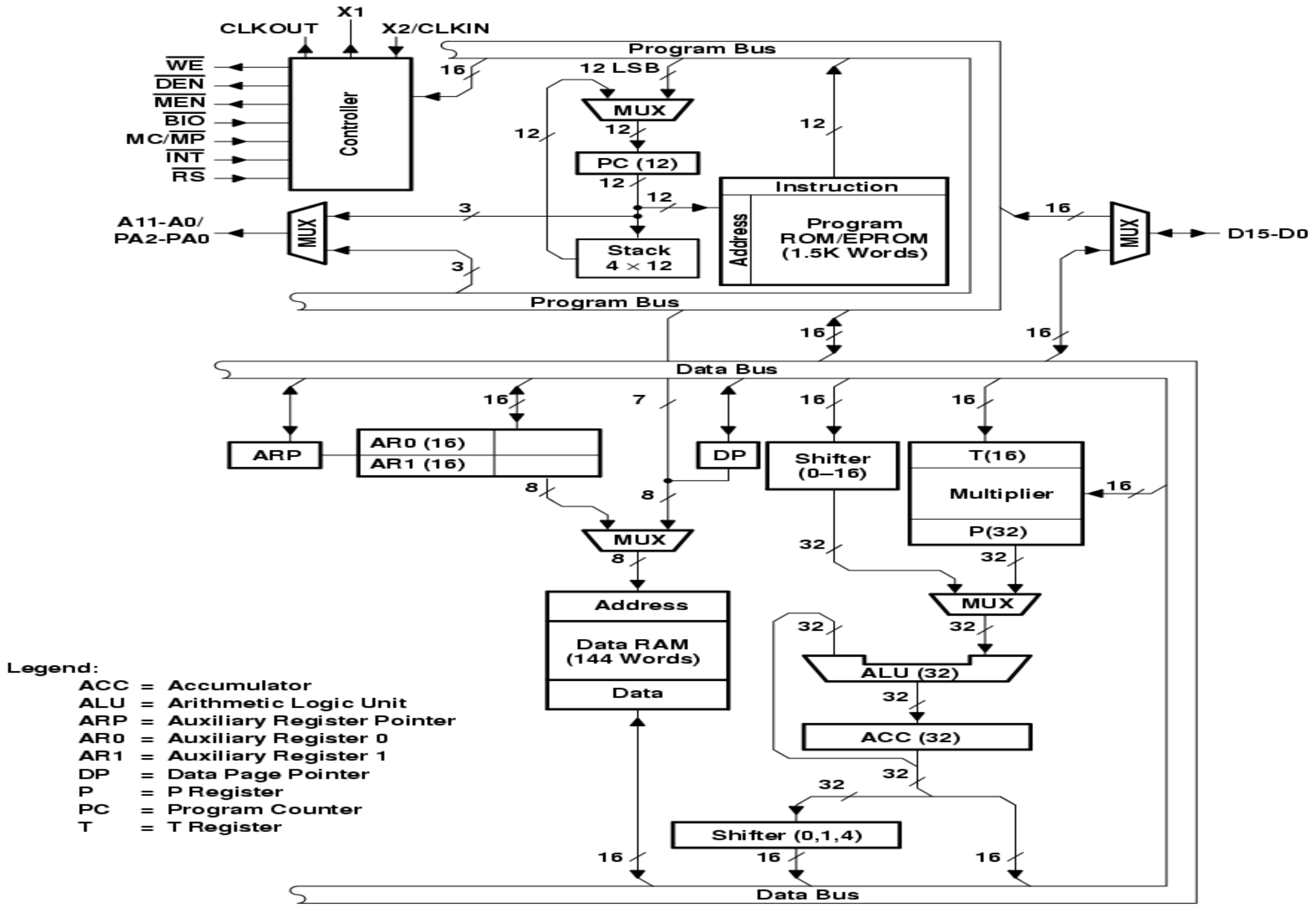
First Generation DSP μP

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 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

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
```

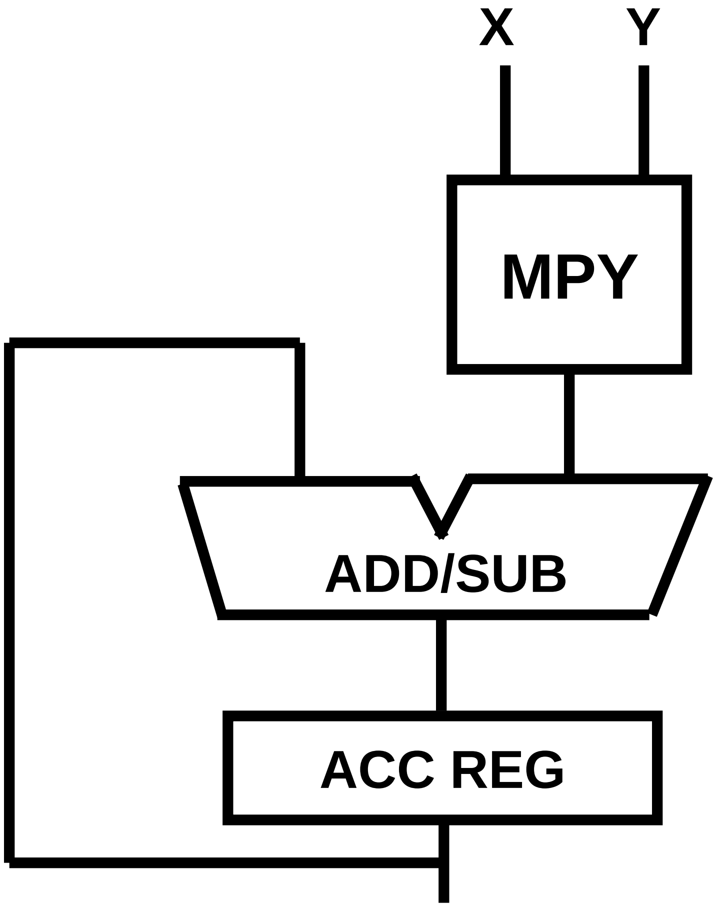
...

- **Two instructions per tap, but requires unrolling**

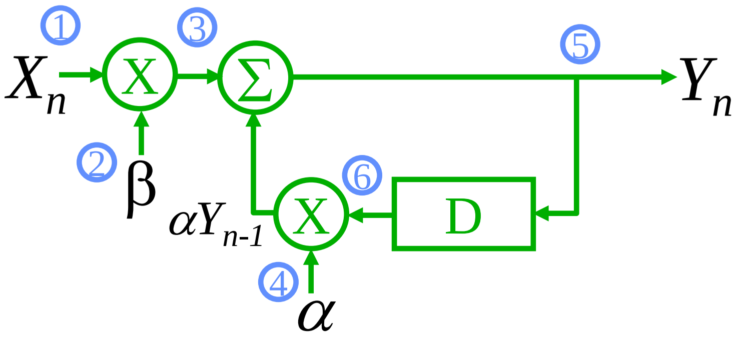
Micro-architectural impact - MAC

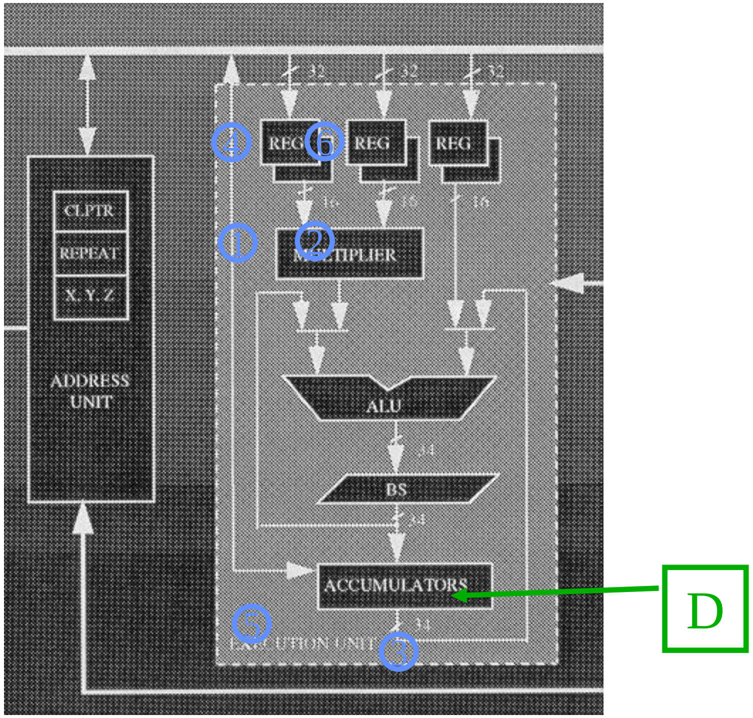
$$y(n) = \sum_{0}^{N-1} h(m)x(n-m)$$

element of finite-impulse response filter computation



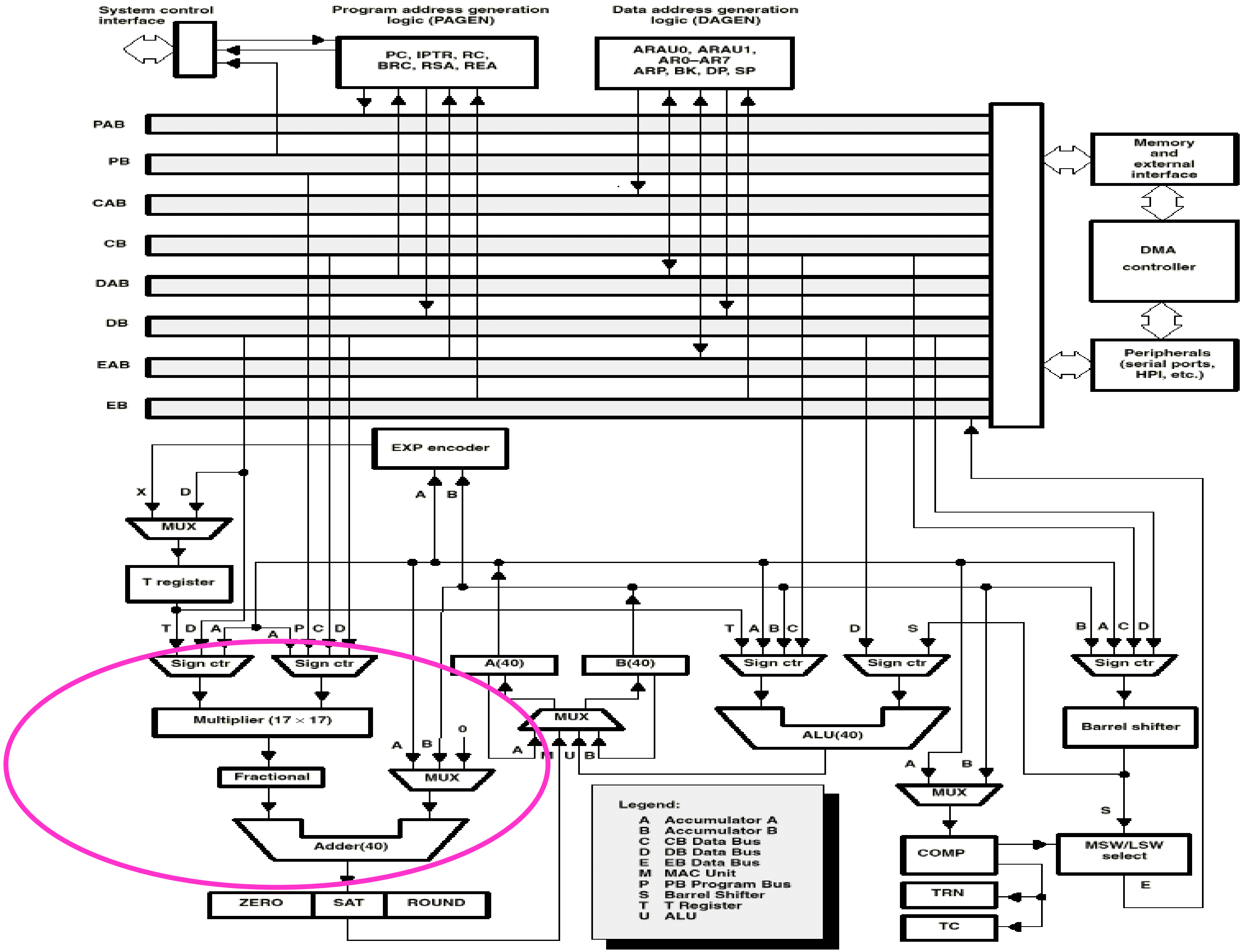
Mapping of the filter onto a DSP execution unit





- 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



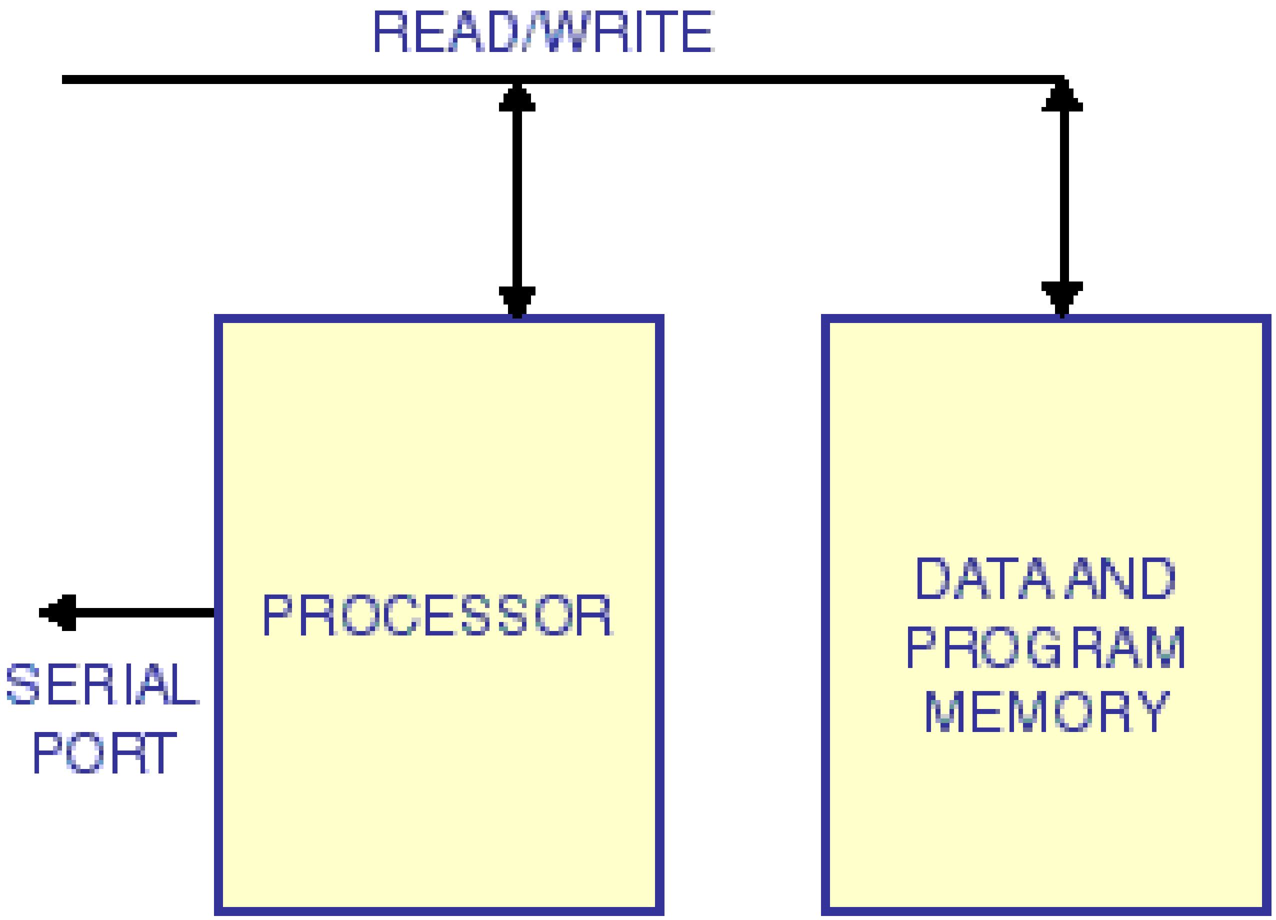
Legend:

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

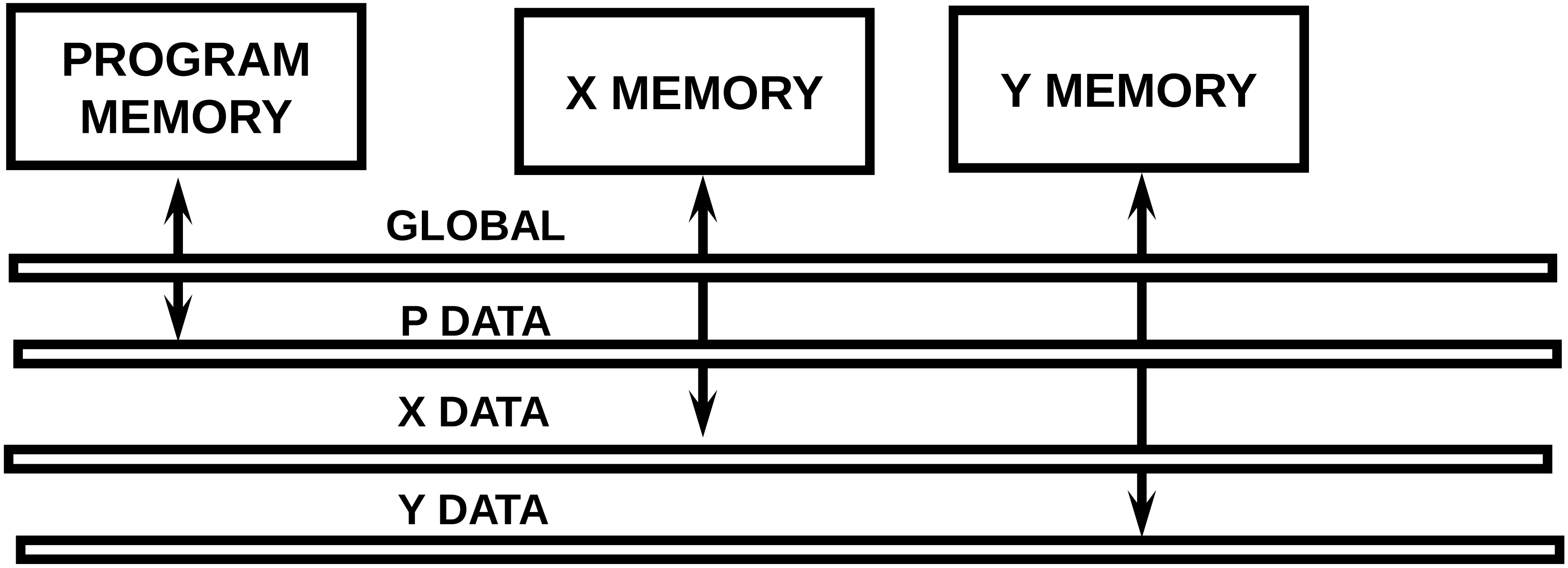
DSP Memory

- FIR Tap implies multiple memory accesses
- DSPs require 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 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



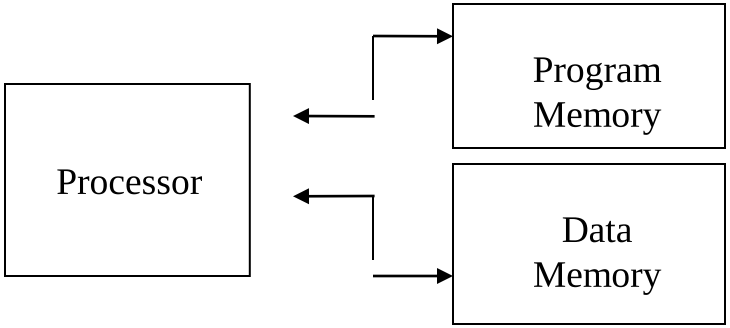
Memory Architecture Comparison

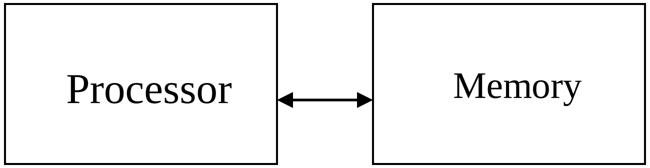
DSP Processor

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

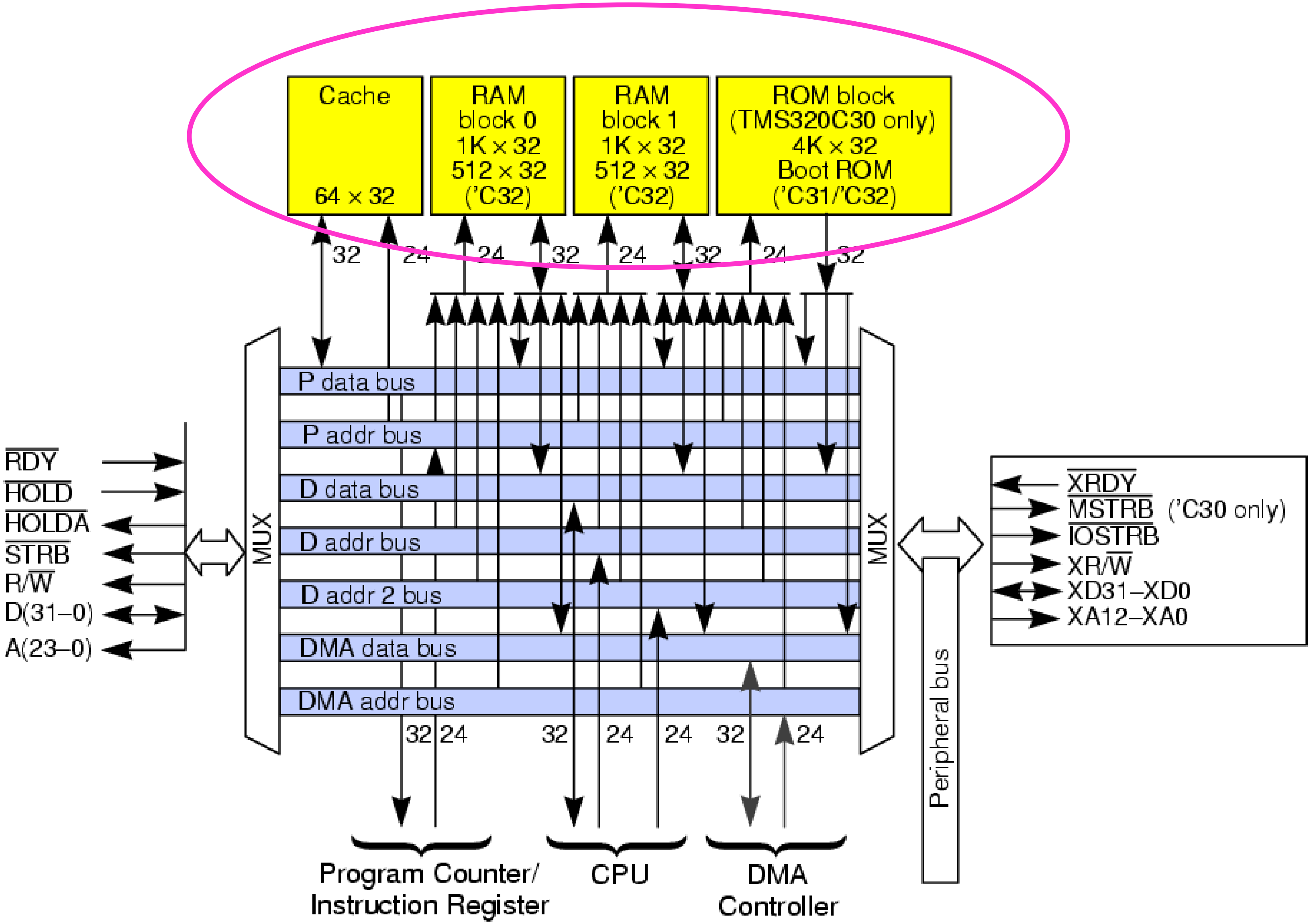
General-Purpose Processor

- **Von Neumann architecture**
- **Typically 1 access/cycle**
- **Use caches**

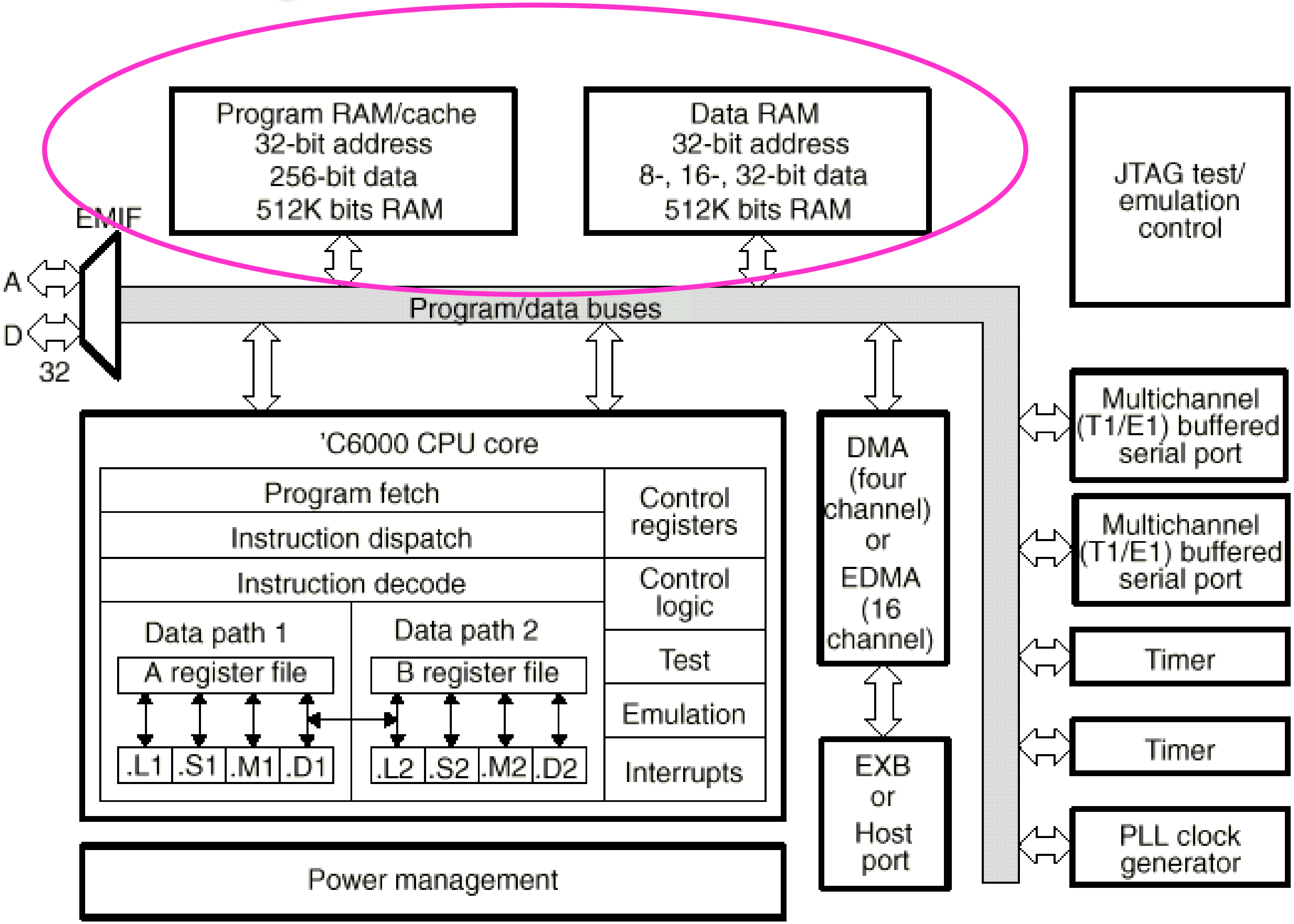




Eg. TMS320C3x MEMORY BLOCK DIAGRAM - Harvard Architecture



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

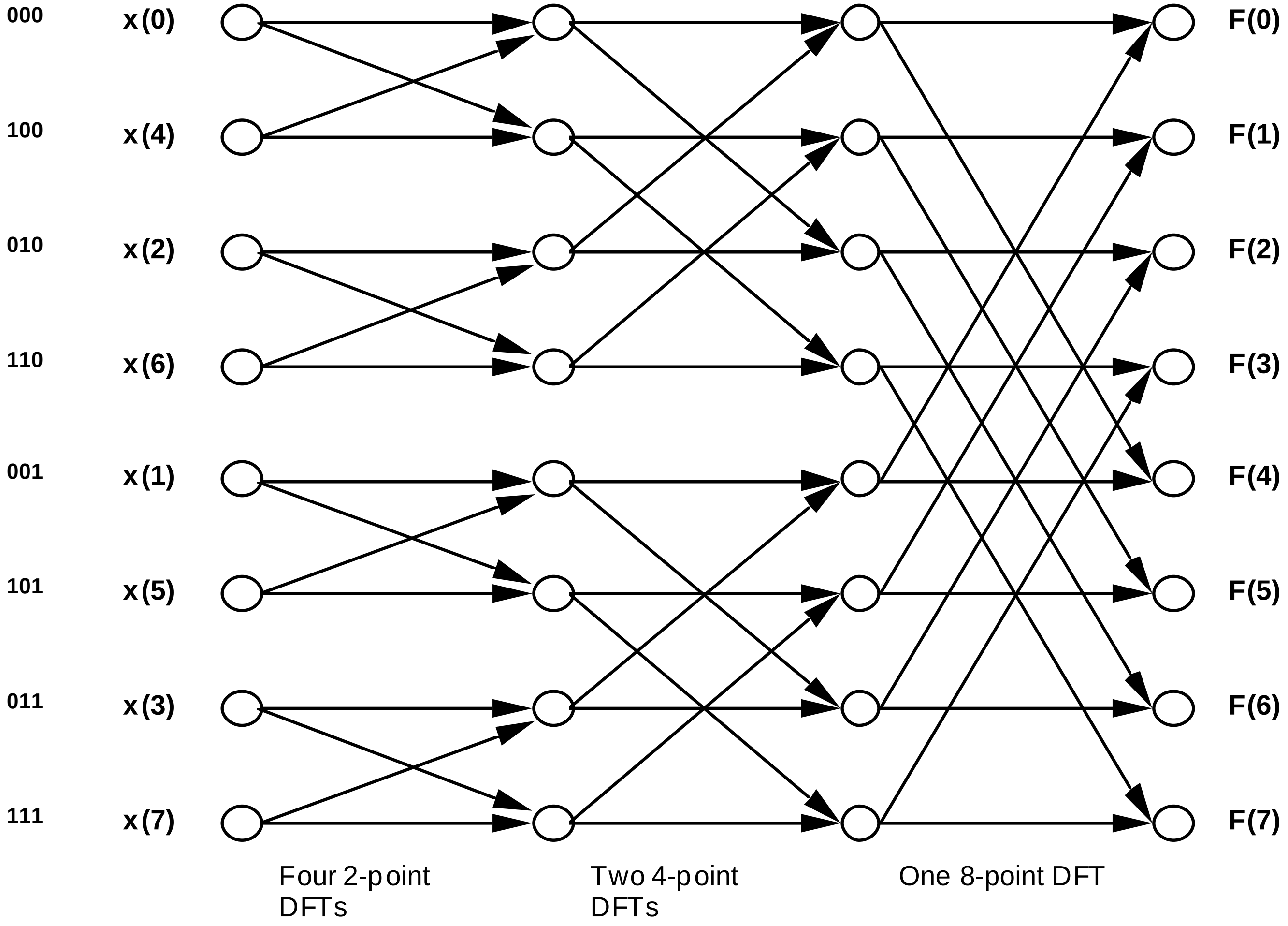
DSP Addressing: FFT

- **FFTs start or end with data in 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 "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 "<u>modulo</u>" or "<u>circular</u>" addressing

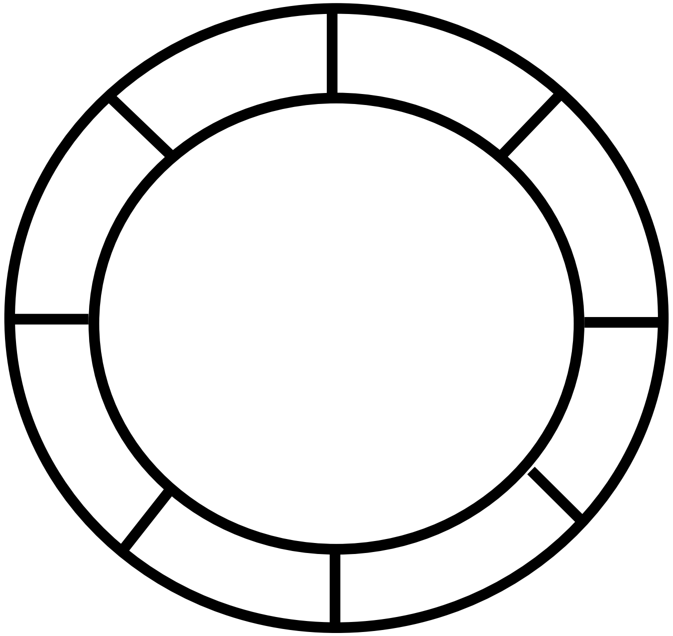
CIRCULAR BUFFERS

Instructions accomodate three elements:

- buffer address
- buffer size
- increment

Allows for cycling through:

- delay elements
- coefficients in data memory



Addressing Comparison

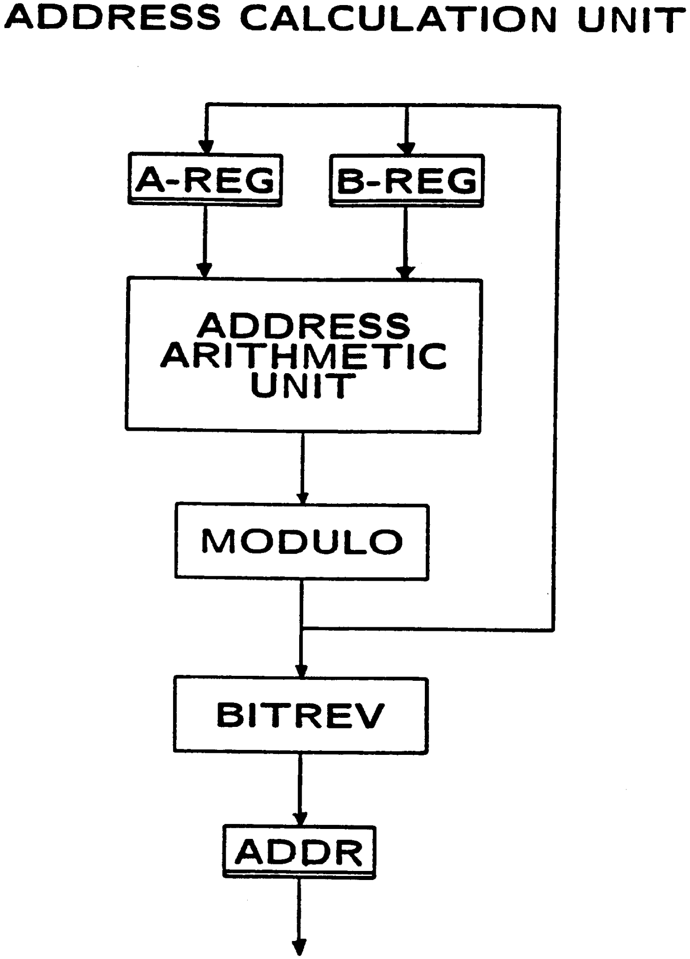
DSP Processor

- 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

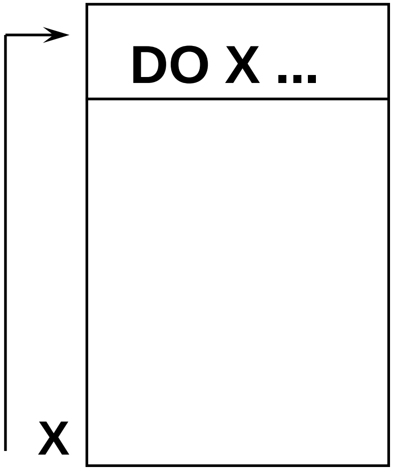
Often duplicated to calculate multiple addresses per cycle

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
```

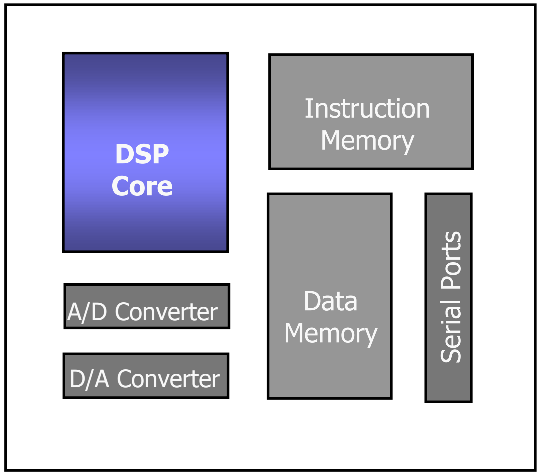
General-Purpose Processor

- **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

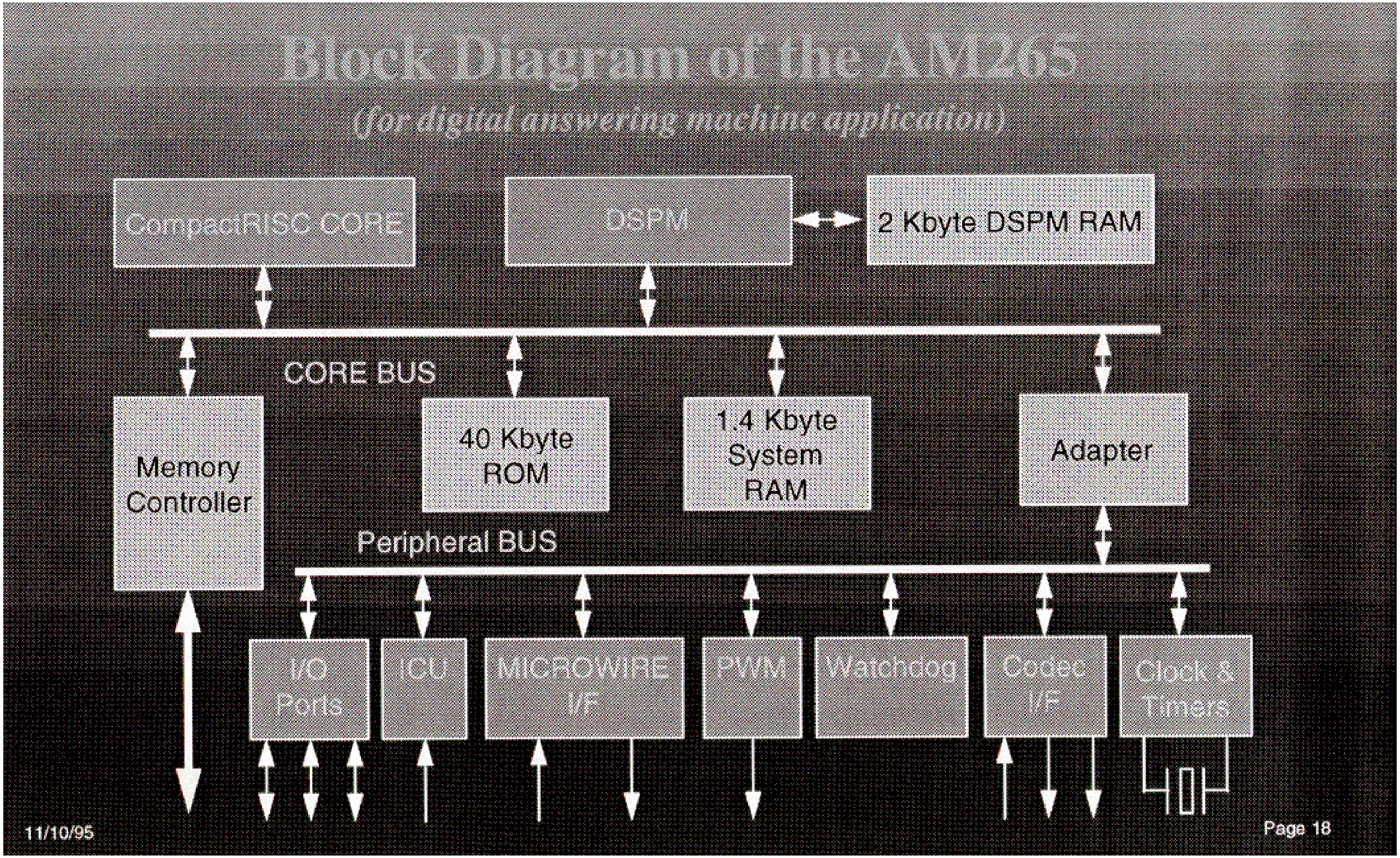
- Synchronous serial ports
- Parallel ports
- Timers
- On-chip A/D, D/A converters



- 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



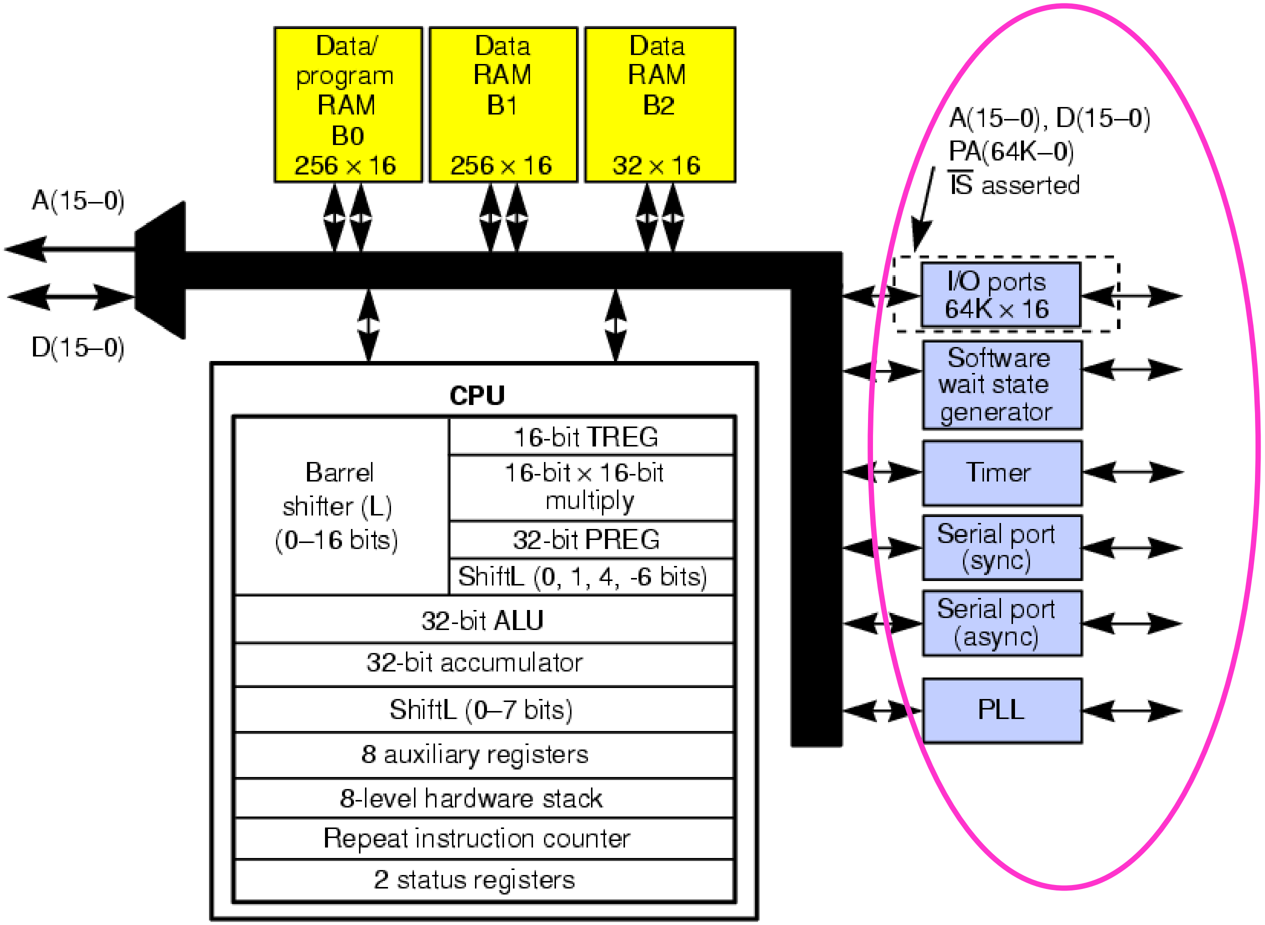
Block Diagram of the AM265

(for digital answering machine application)

11/10/95

Page 18

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.**

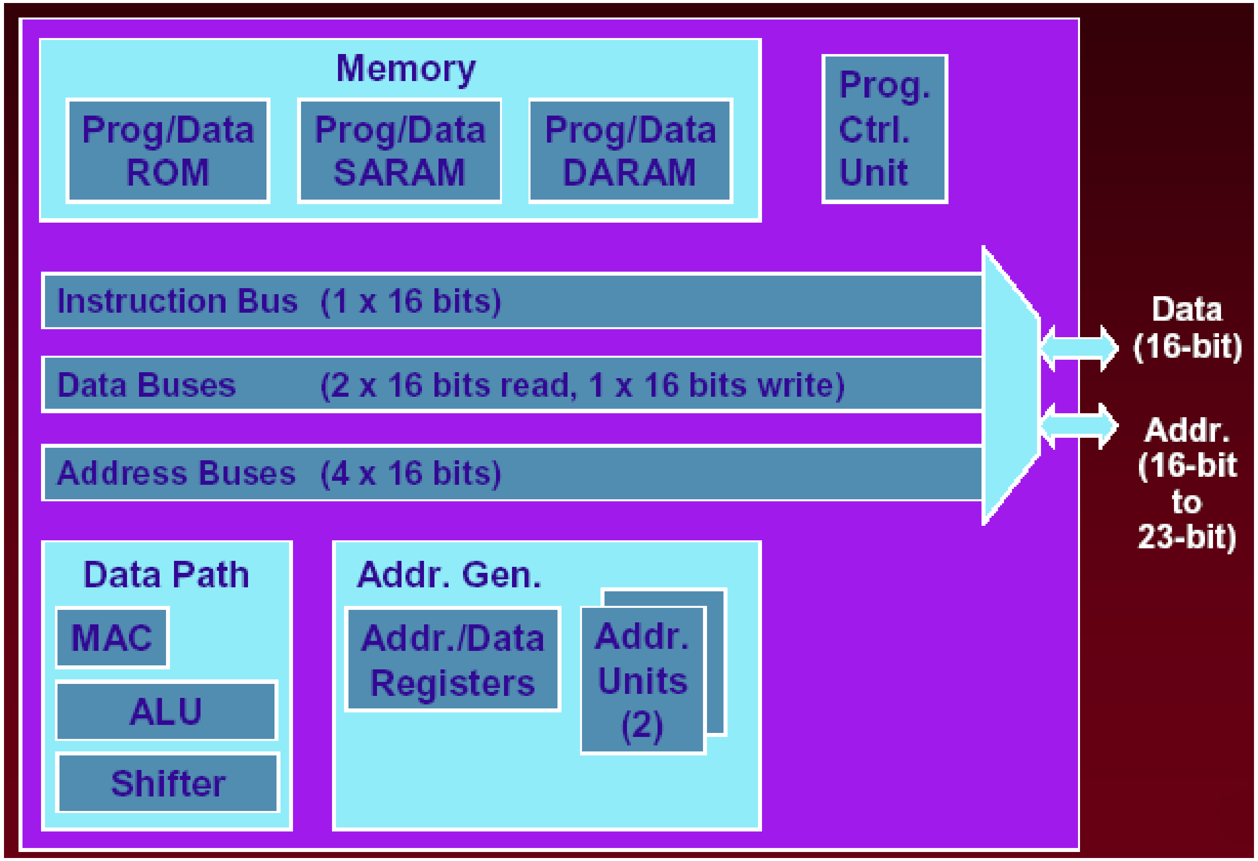
Classification of Current DSP Architectures

- **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).

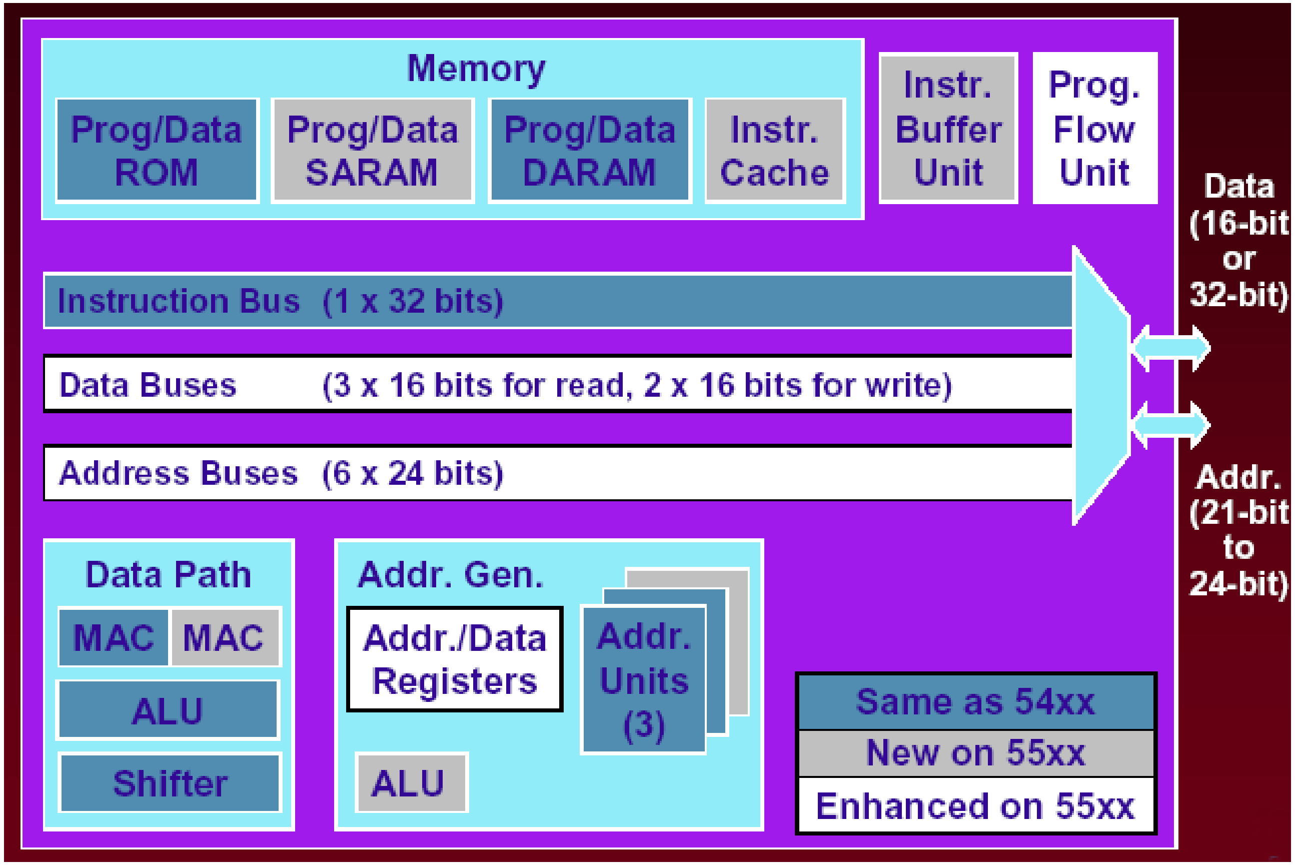
A Current Conventional DSP: TI TMSC54xx



An Enhanced Conventional DSP: TI TMSC55xx

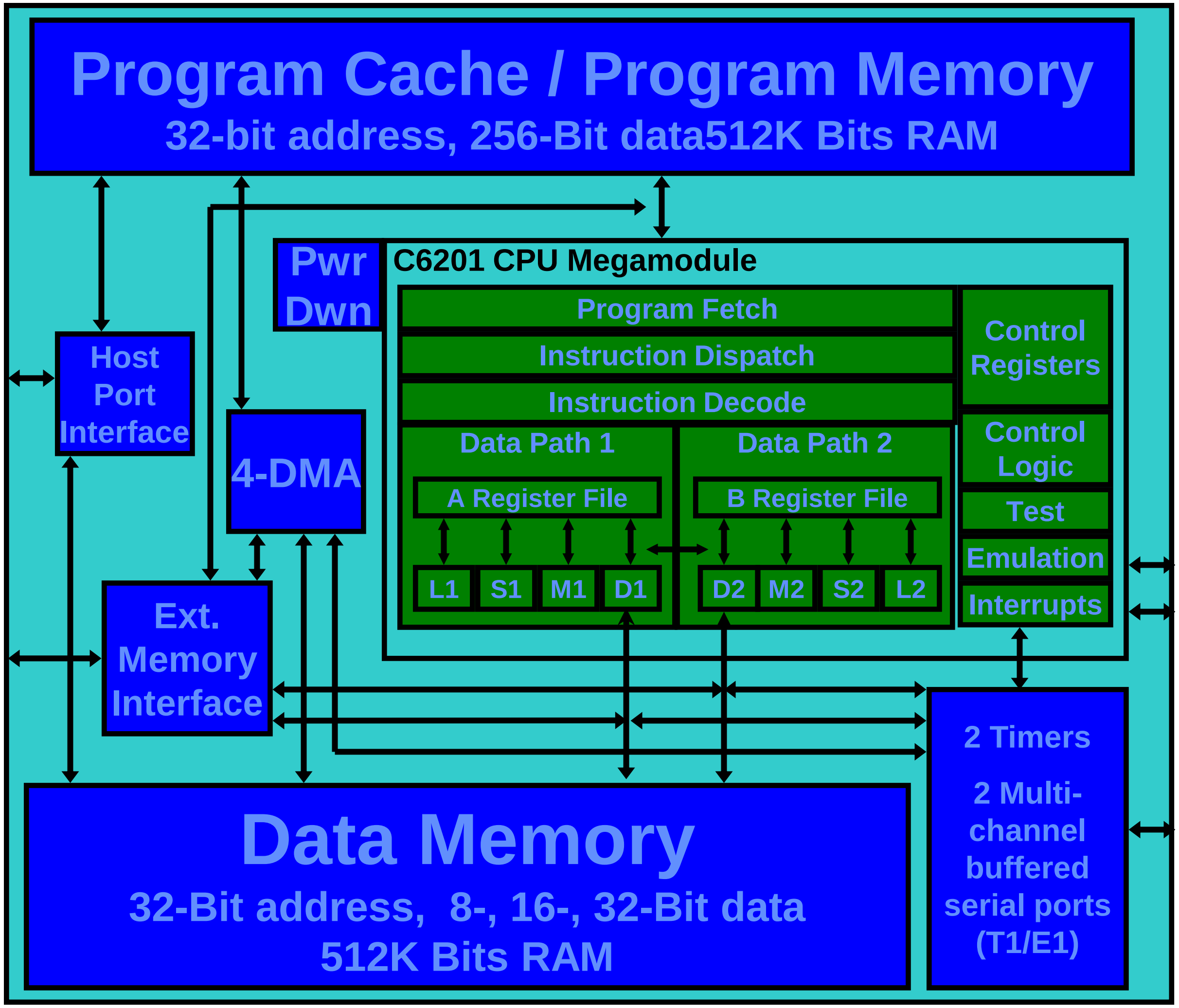
- **The TMS320C55xx is based on Texas Instruments' earlier TMS320C54xx family, but adds significant enhancements to the architecture and instruction set, including:**
  - **Two instructions/cycle**
    - Instructions are scheduled for parallel execution by the assembly programmer or compiler.
  - **Two 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



16-bit Fixed-Point VLIW DSP:
TI TMS320C6201 Revision 2 (1997)

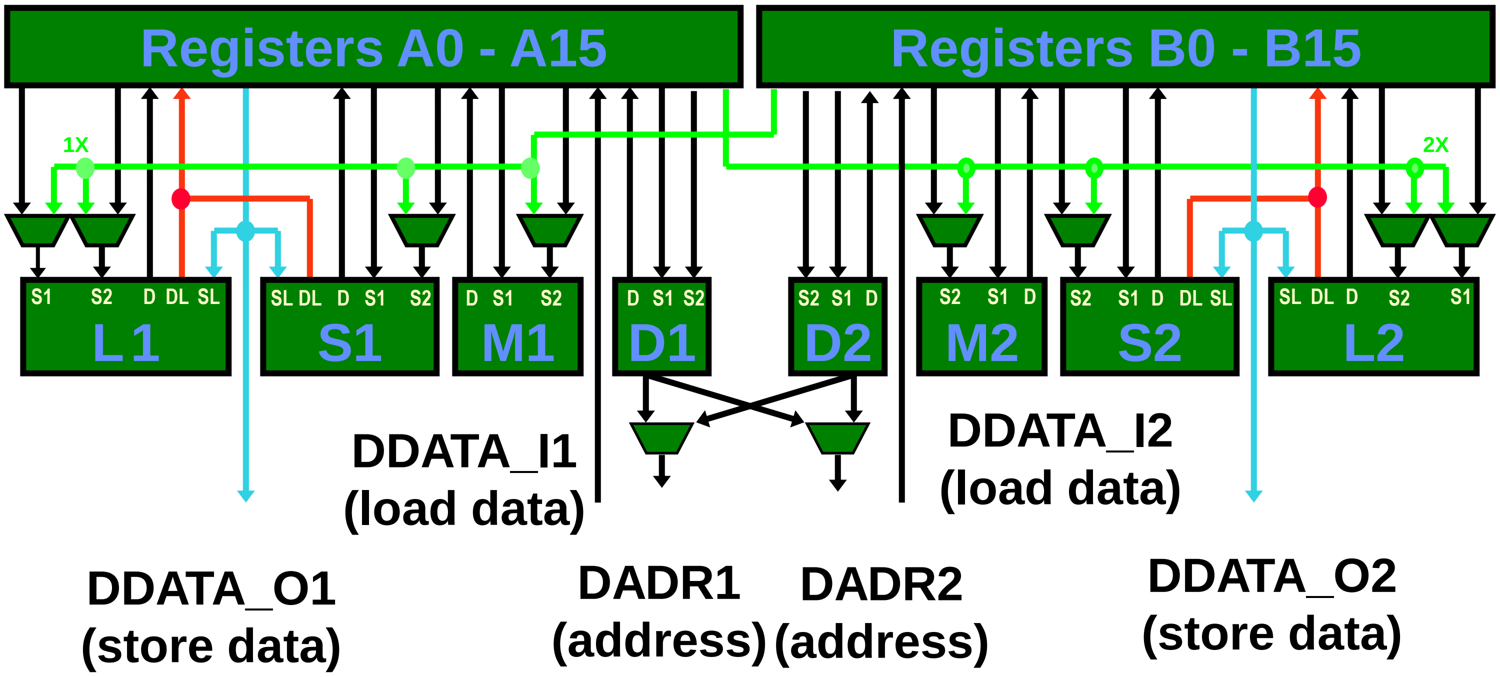
The TMS320C62xx is the first fixed-point DSP processor from Texas Instruments that is based on a VLIW-like architecture which allows it to execute up to eight 32-bit RISC-like instructions per clock cycle.



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



DDATA_I1
(load data)

DDATA_I2
(load data)

DDATA_O1
(store data)

DADR1
(address)

DADR2
(address)

DDATA_O2
(store data)

— Cross Paths

— 40-bit Write Paths (8 MSBs)

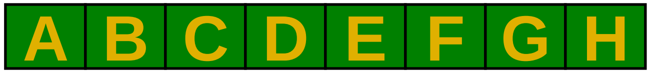
— 40-bit Read Paths/Store Paths

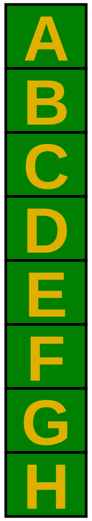
C62x Functional Units

- 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

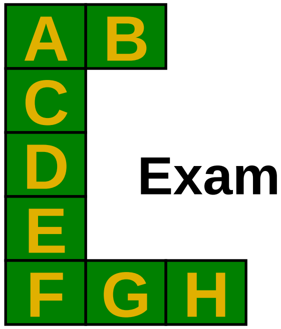
C62x Instruction Packing
Instruction Packing Advanced VLIW

Example 1





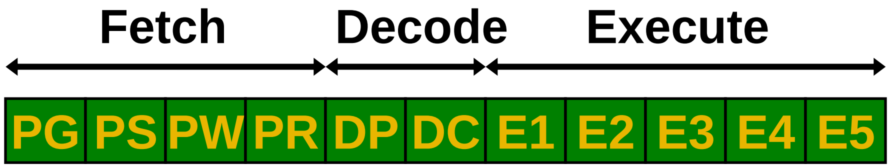
Example 2



Example 3

- 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

C62x Pipeline Operation
Pipeline Phases



Fetch

Decode

Execute

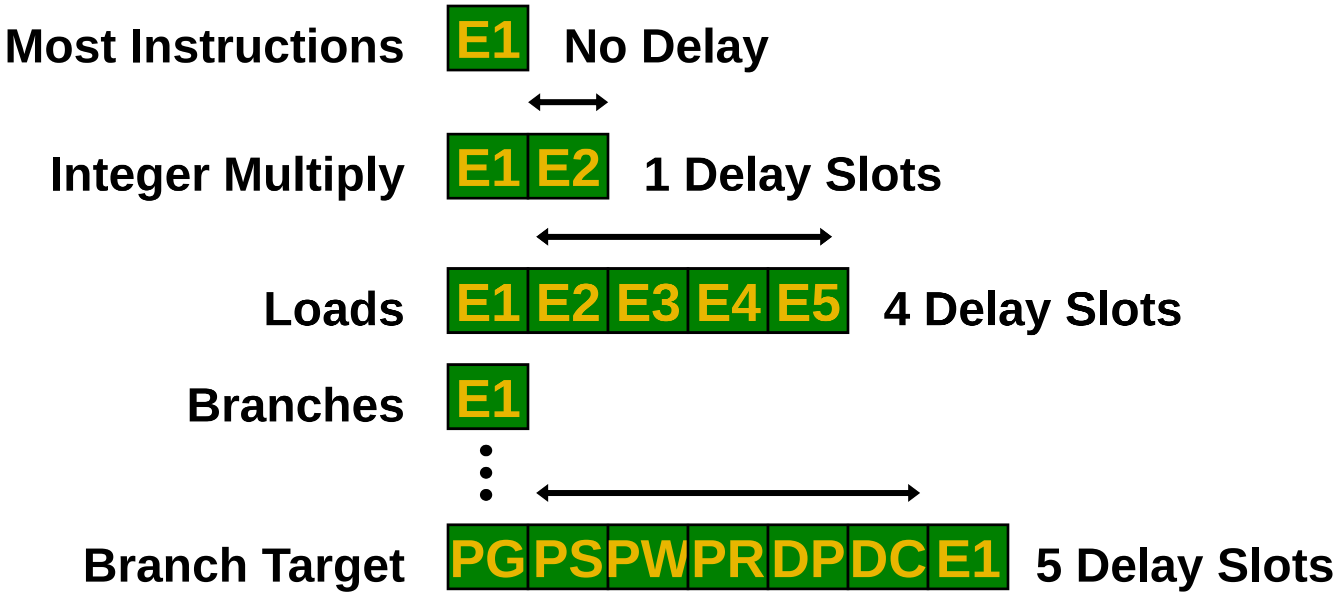
PG PS PW PR DP DC E1 E2 E3 E4 E5

- 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

	PG	PS	PW	PR	DP	DC	E1	E2	E3	E4	E5						
Execute Packet 2		PG	PS	PW	PR	DP	DC	E1	E2	E3	E4	E5					
Execute Packet 3			PG	PS	PW	PR	DP	DC	E1	E2	E3	E4	E5				
Execute Packet 4				PG	PS	PW	PR	DP	DC	E1	E2	E3	E4	E5			
Execute Packet 5					PG	PS	PW	PR	DP	DC	E1	E2	E3	E4	E5		
Execute Packet 6						PG	PS	PW	PR	DP	DC	E1	E2	E3	E4	E5	
Execute Packet 7							PG	PS	PW	PR	DP	DC	E1	E2	E3	E4	E5

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



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

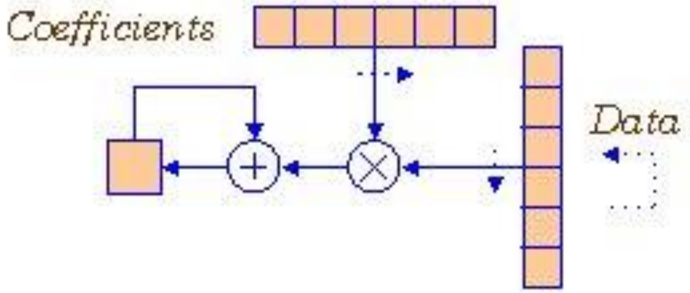
C6000 Instruction Set Addressing Features

- 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**

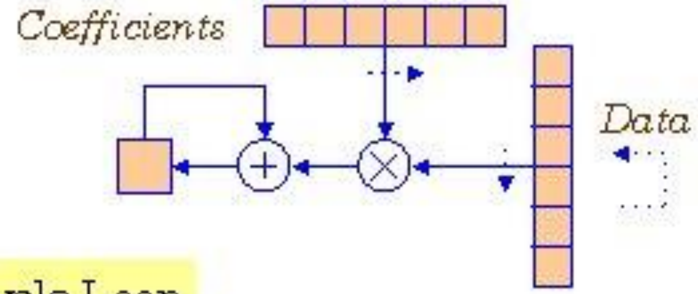
Application: FIR Filter on a TMS320C5x



```
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 .M1x 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.

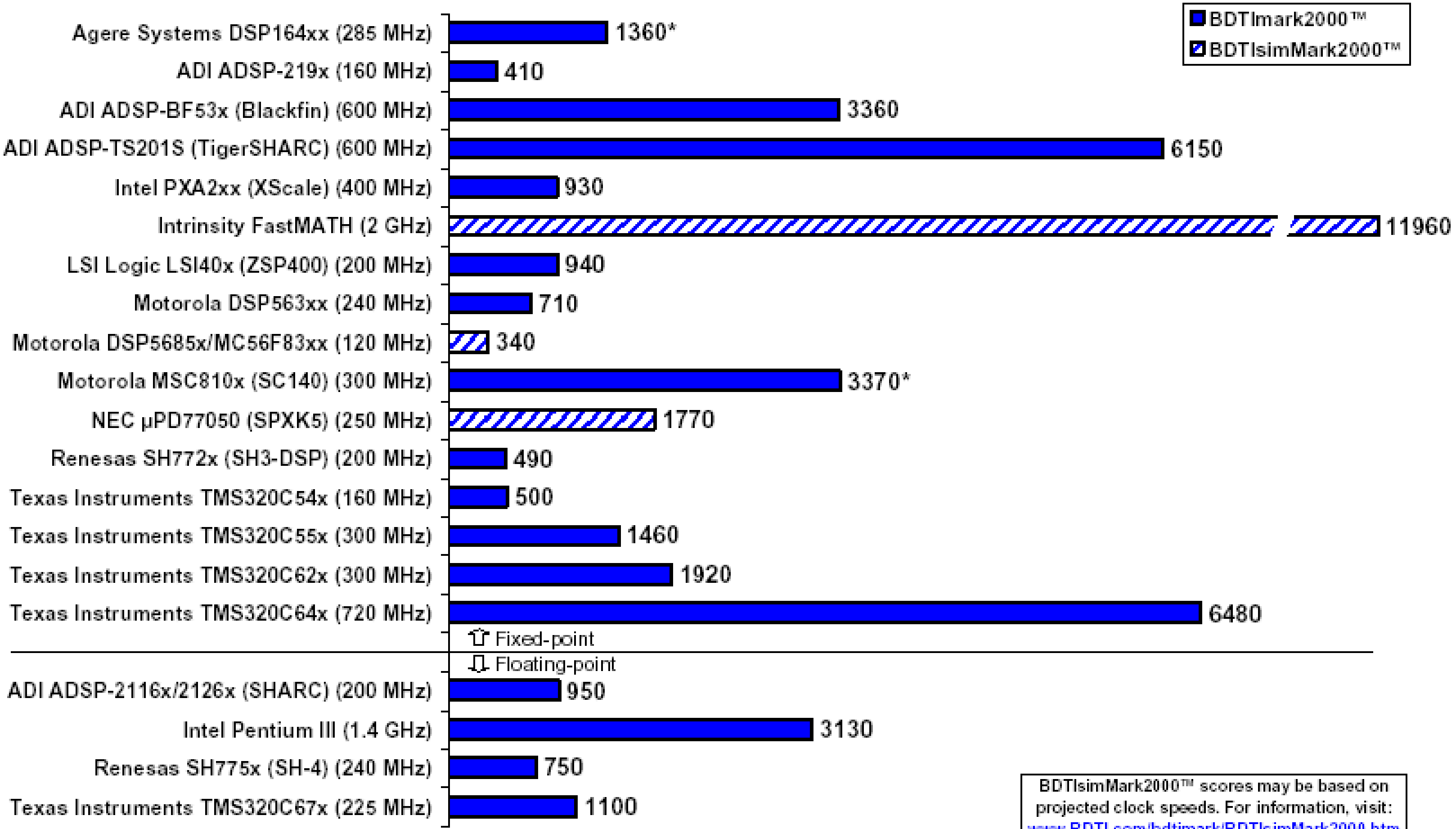
Superscalar DSP: LSI Logic ZSP400

- **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.**

BDTImark2000™/BDTIsimMark2000™ Scores for Packaged Processors

Updated October 2003

Contact BDTI for authorization to publish BDTImark2000™/BDTIsimMark2000™ scores.



* for one core

BDTIsimMark2000™ scores may be based on projected clock speeds. For information, visit: www.BDTI.com/bdtimark/BDTIsimMark2000.htm

Thank You