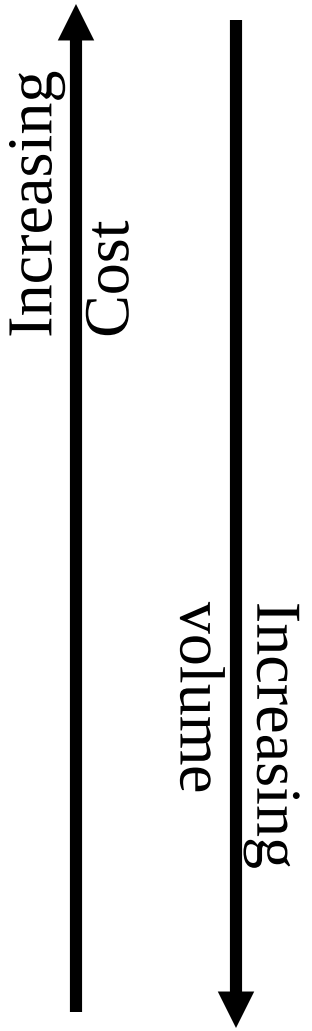
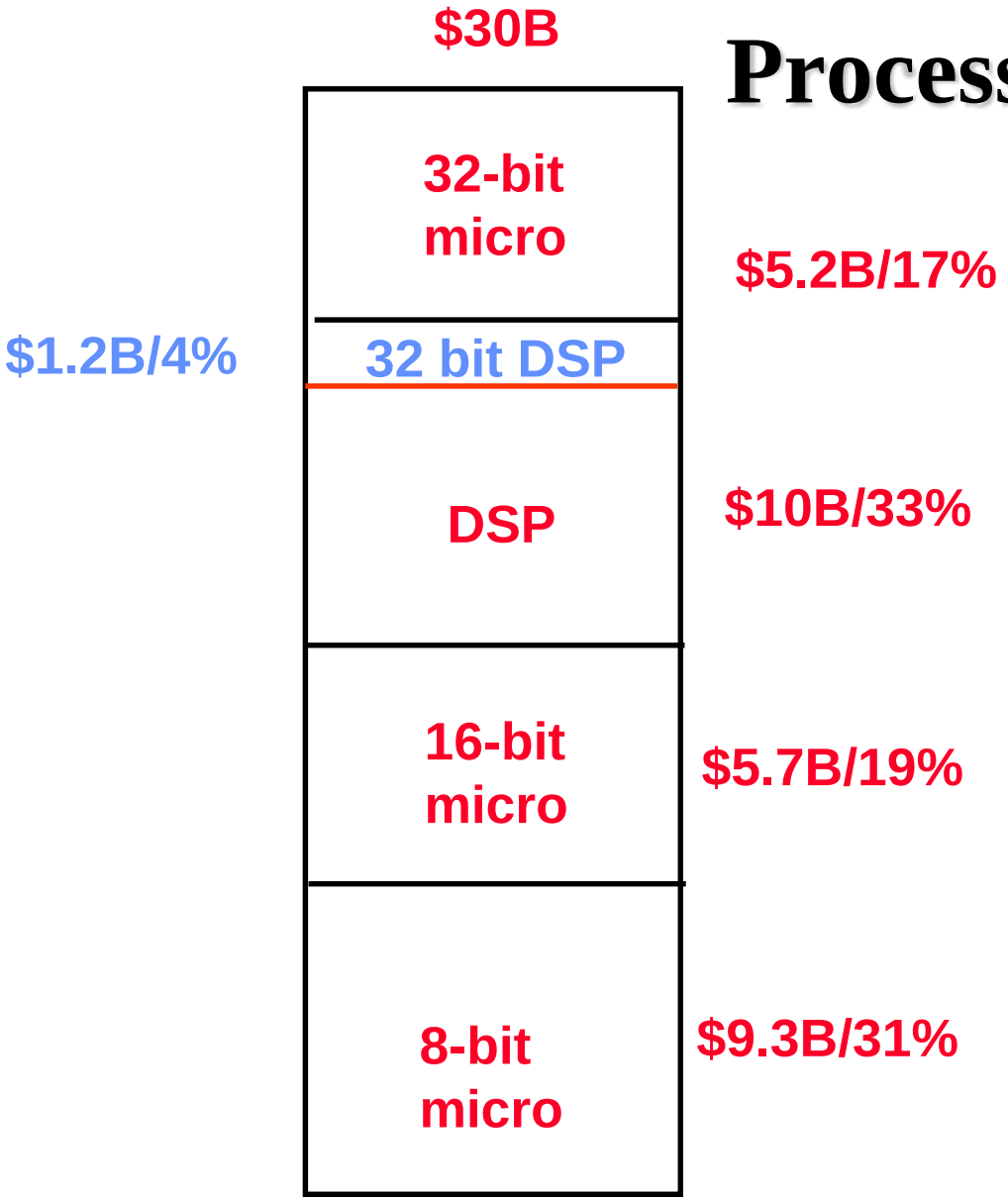
Processor Applications

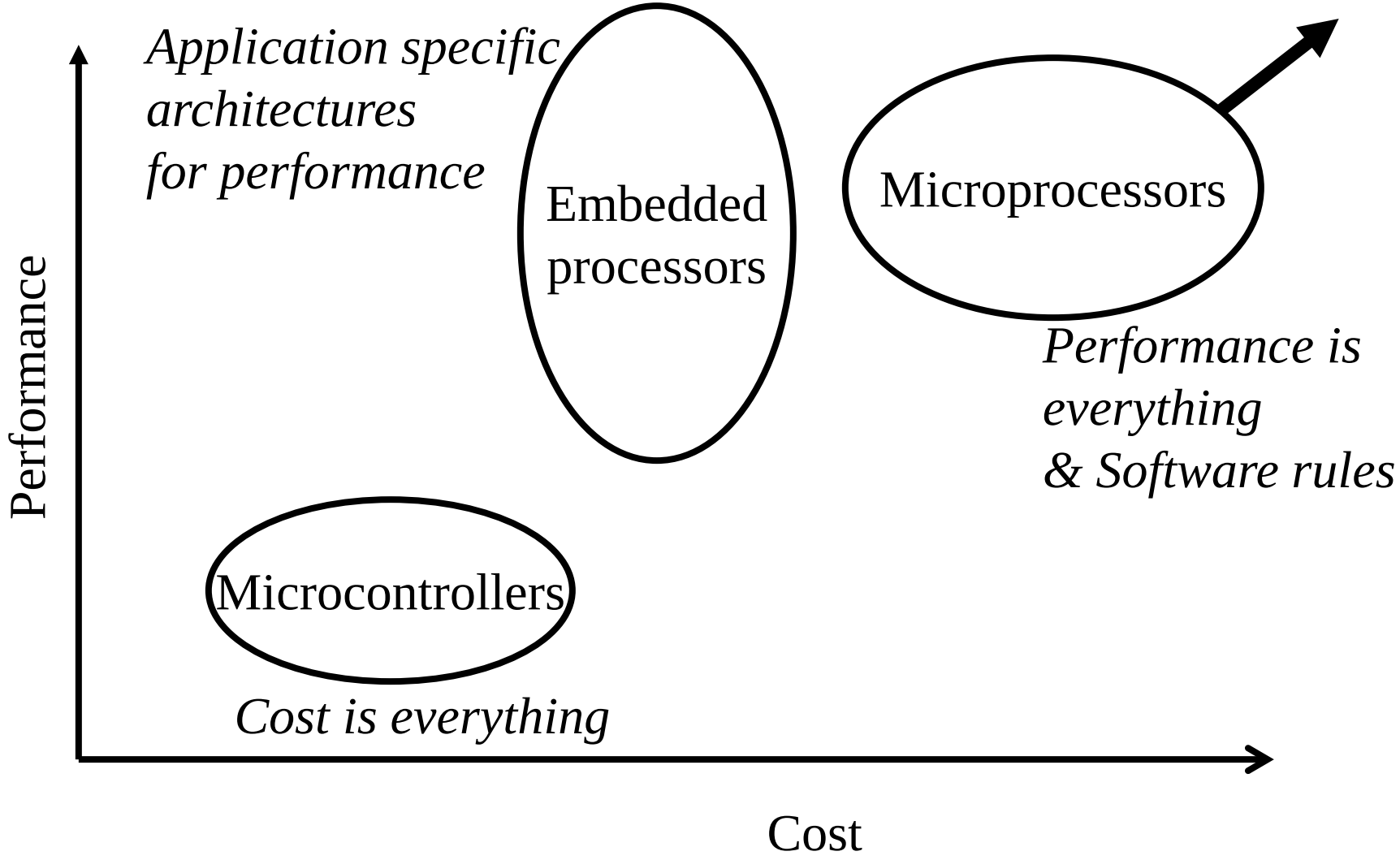
- **General Purpose - high performance**
  - **Alpha's, SPARC, MIPS ..**
  - **Used for general purpose software**
  - **Heavy weight OS - UNIX, NT**
  - **Workstations, PC's**
- **Embedded processors and processor cores**
  - **ARM, 486SX, Hitachi SH7000, NEC V800**
  - **Single program**
  - **Lightweight, 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, ...**



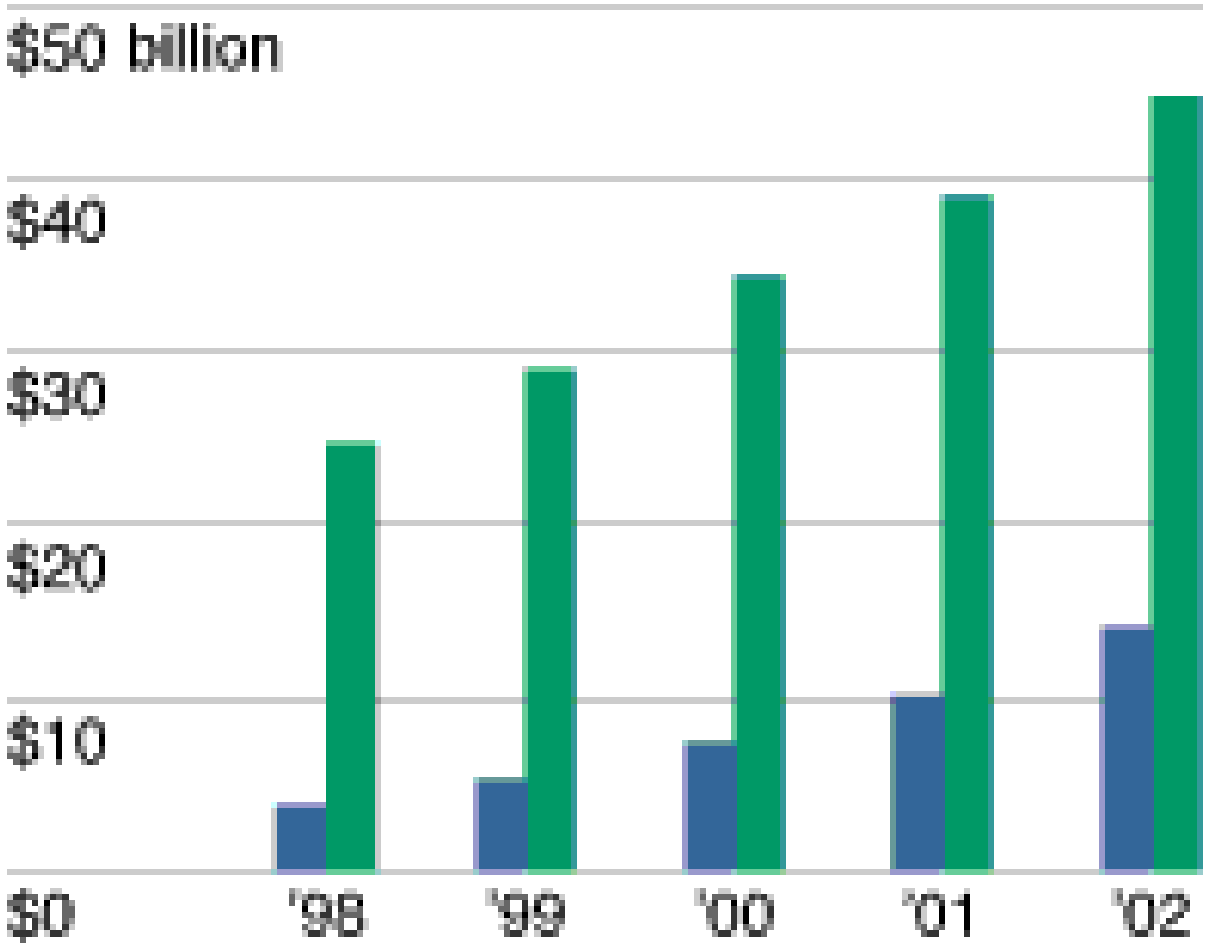
Processor Markets



The Processor Design Space



Market for DSP Products



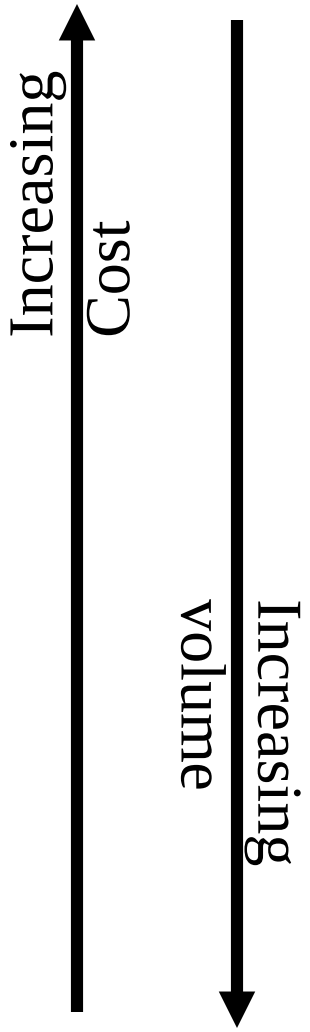
DSP is the fastest growing segment of the semiconductor market

DSP Applications

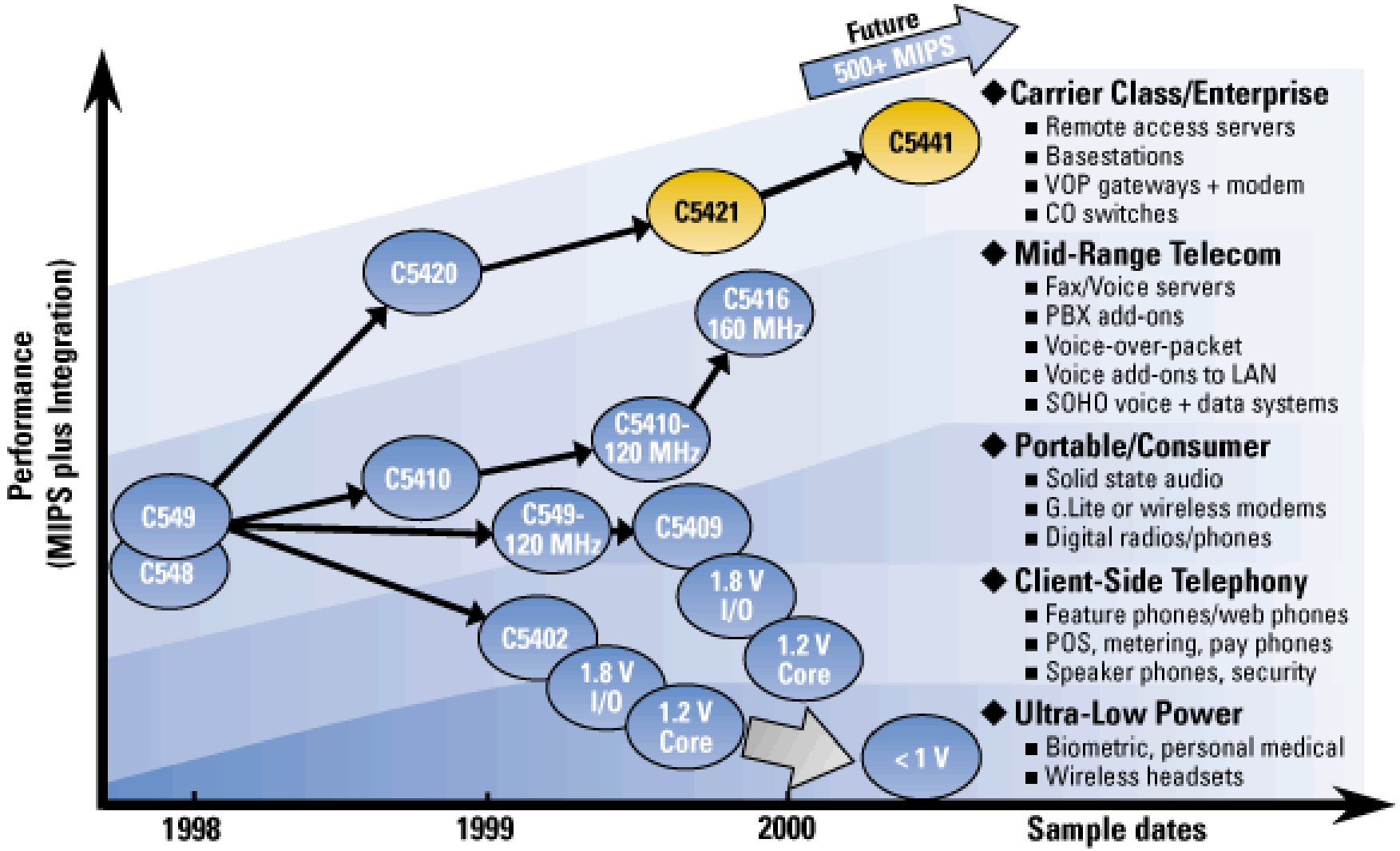
- Audio applications
- MPEG Audio
- Portable audio
- Digital cameras
- Wireless
- Cellular telephones
- Base station
- Networking
- Cable modems
- ADSL
- VDSL

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, ...**



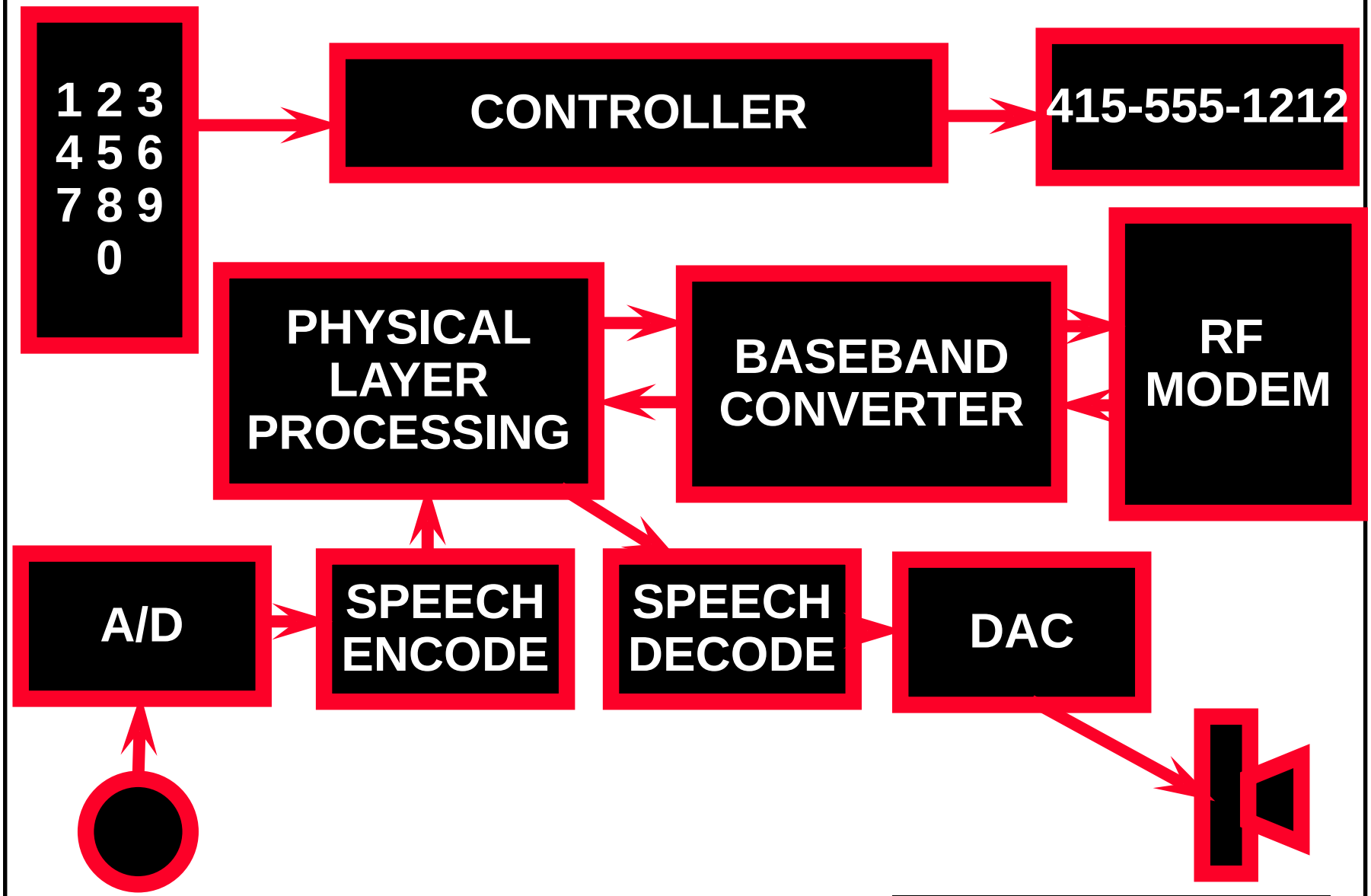
DSP range of applications



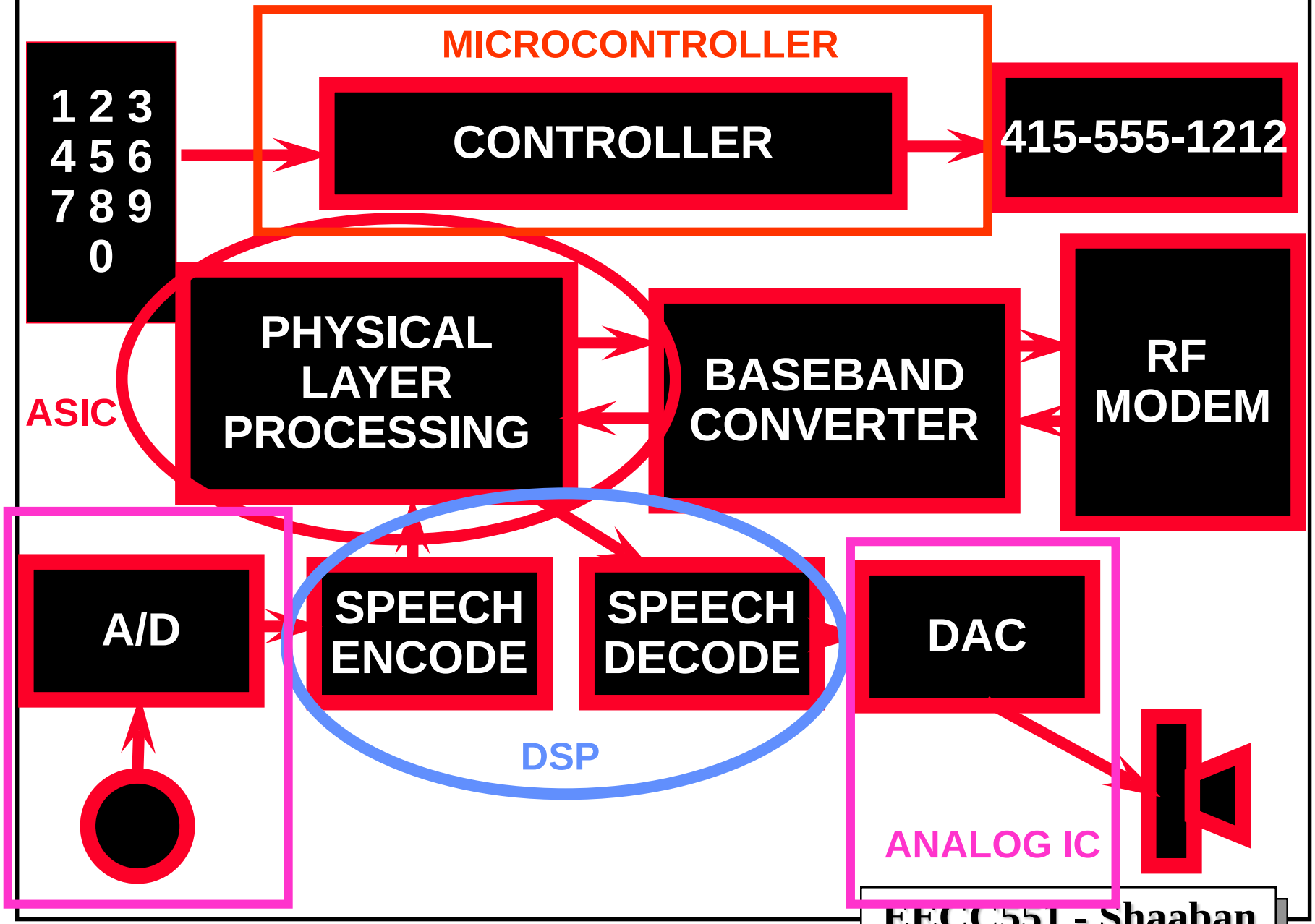
DSP ARCHITECTURE
Enabling Technologies

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

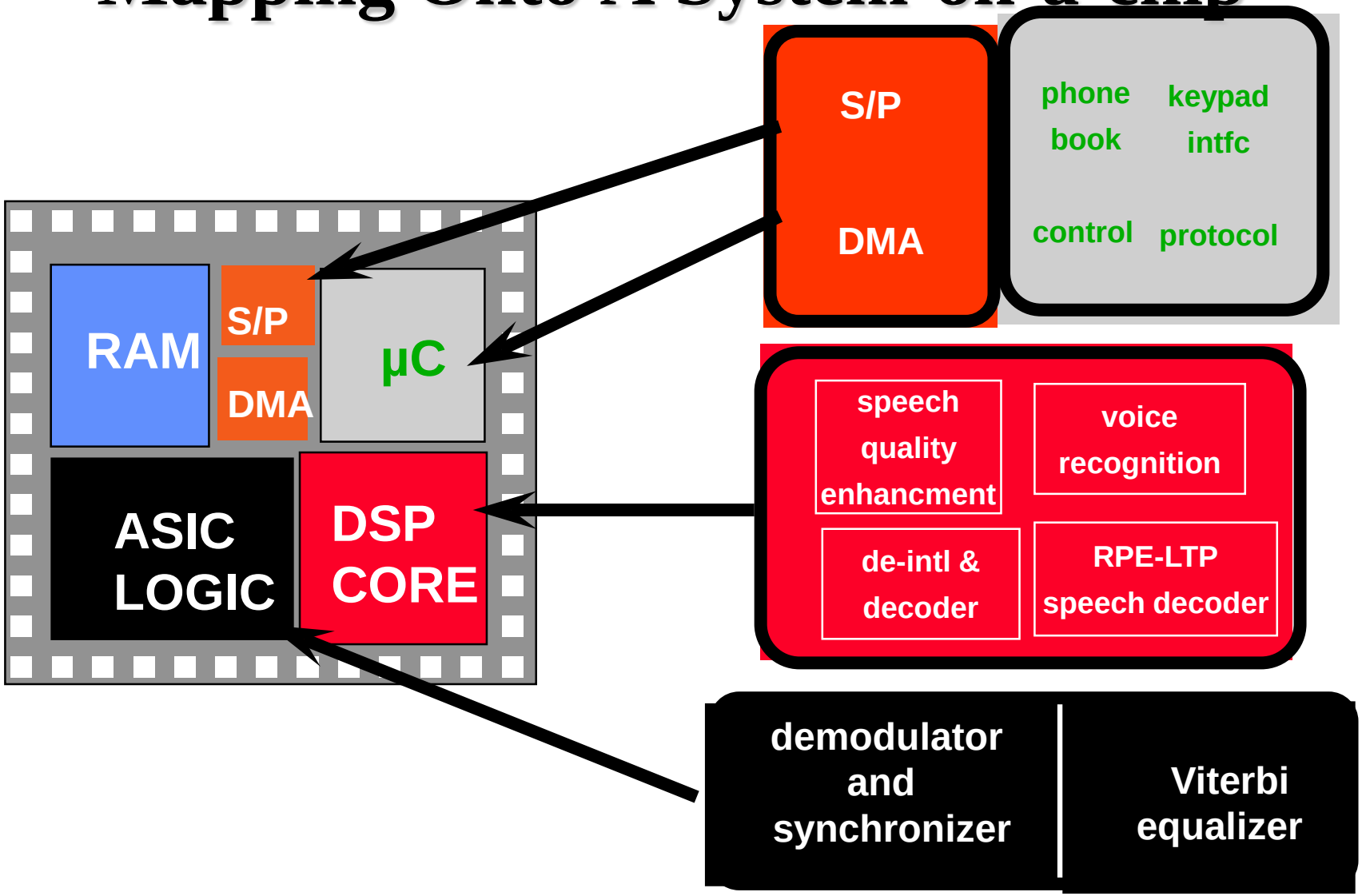
CELLULAR TELEPHONE SYSTEM



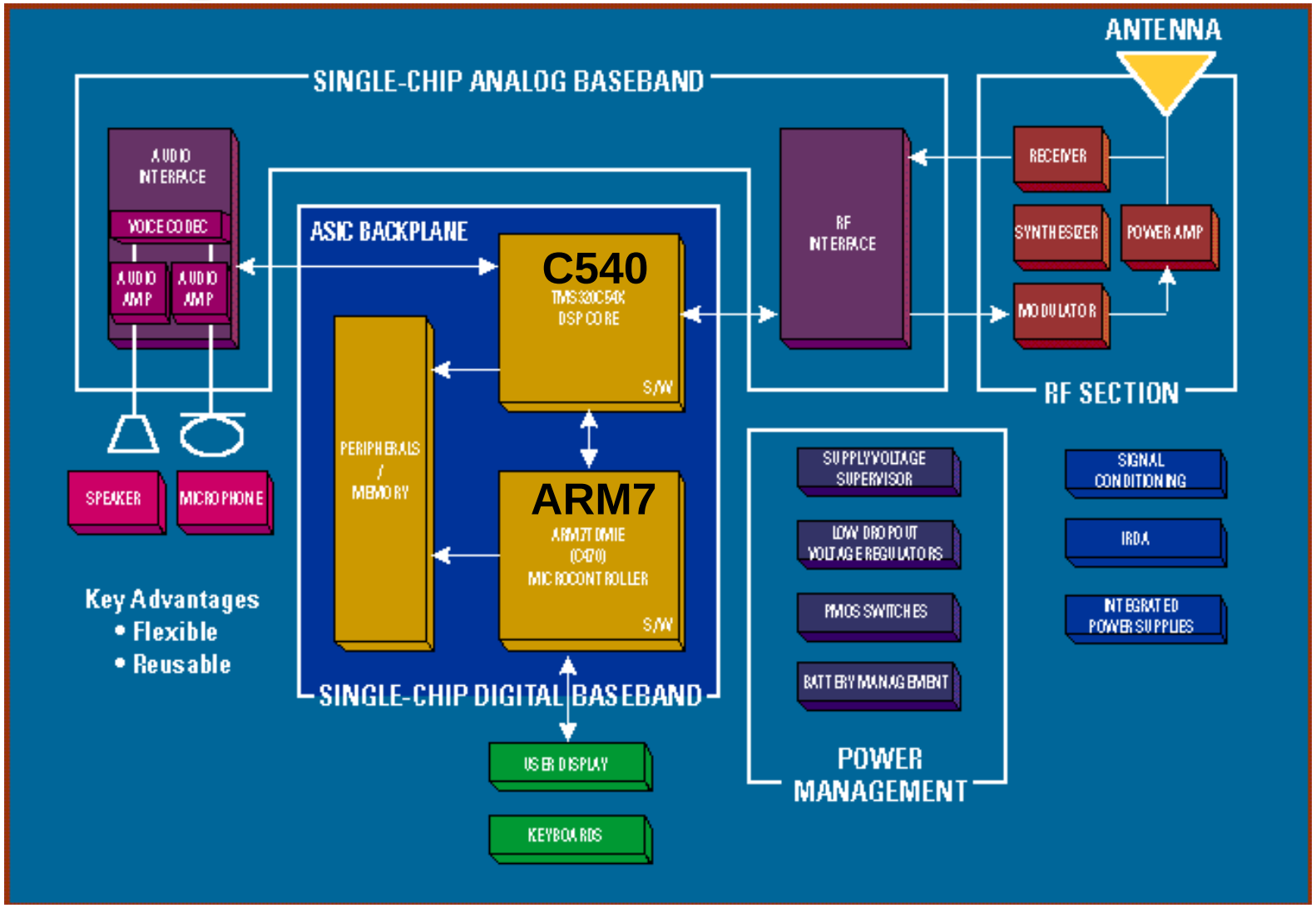
HW/SW/IC PARTITIONING



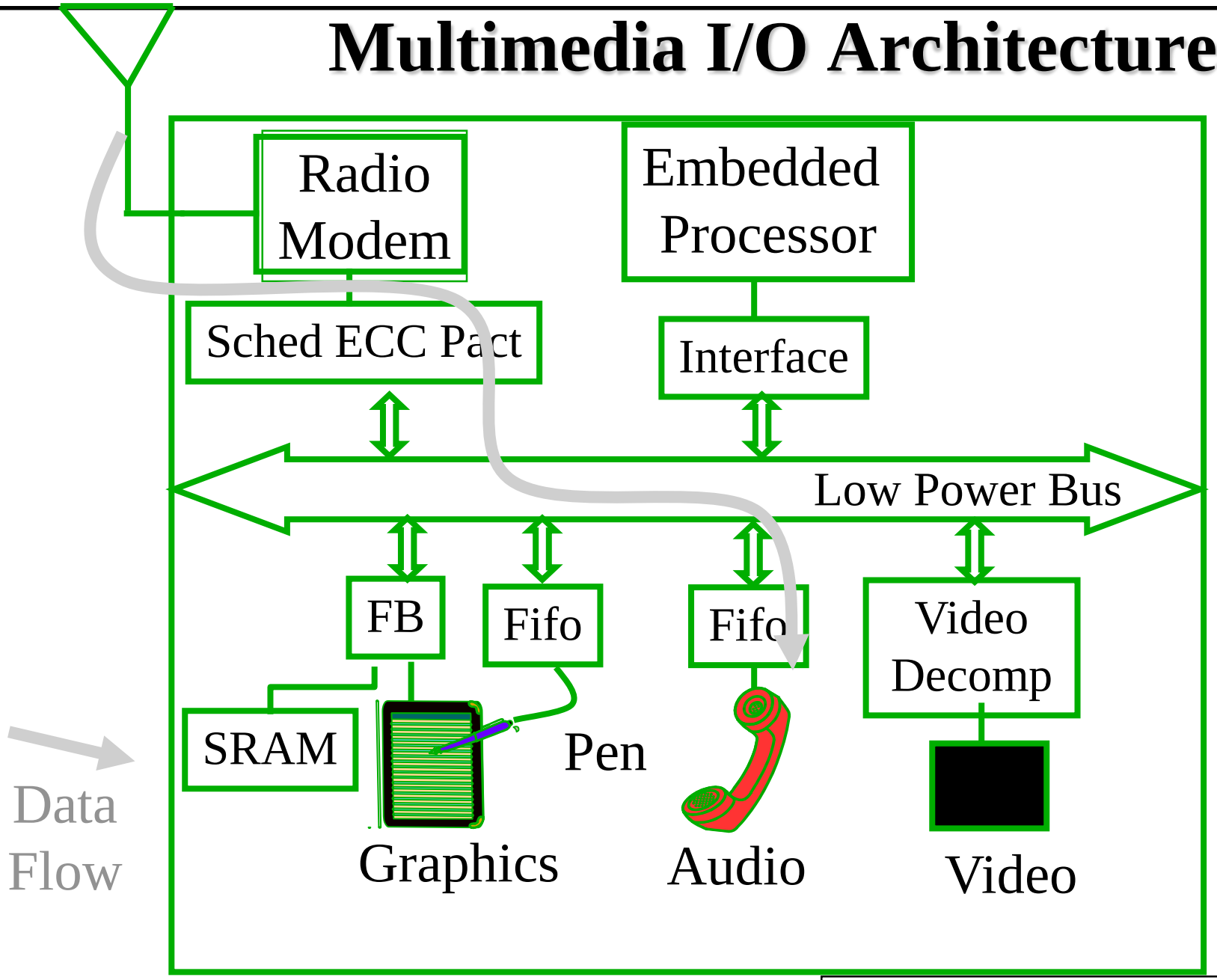
Mapping Onto A System-on-a-chip



Example Wireless Phone Organization

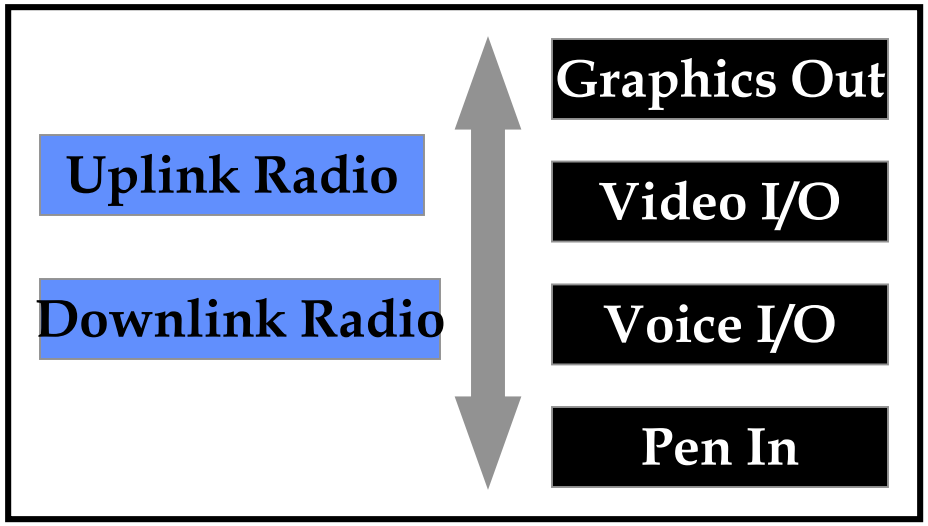


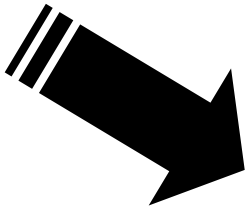
Multimedia I/O Architecture

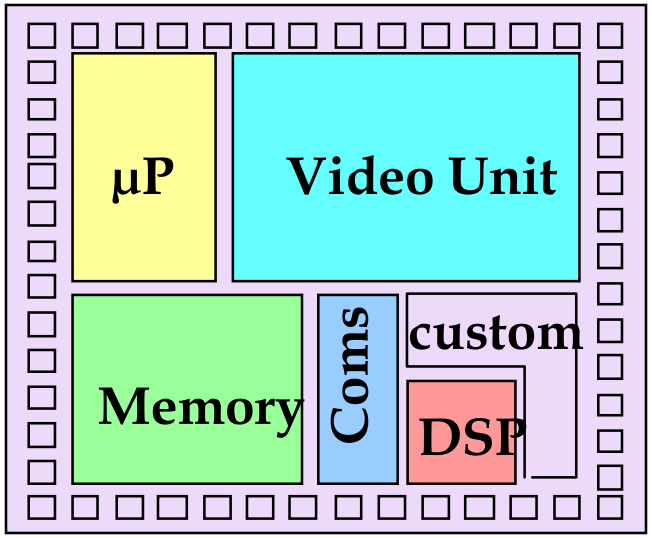


Multimedia System-on-a-Chip

E.g. Multimedia terminal electronics





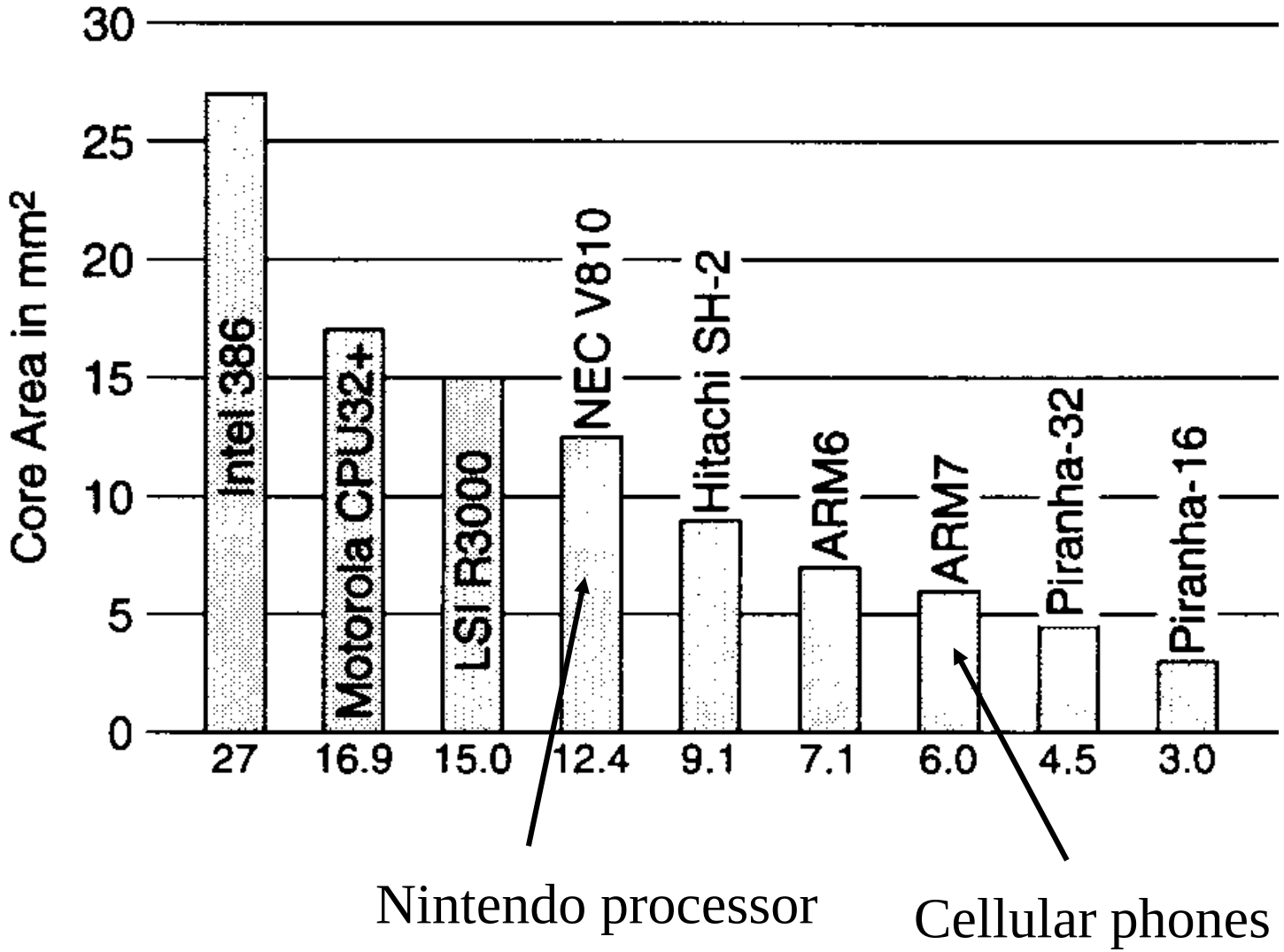


- **Future chips will be a mix of processors, memory and dedicated hardware for specific algorithms and I/O**

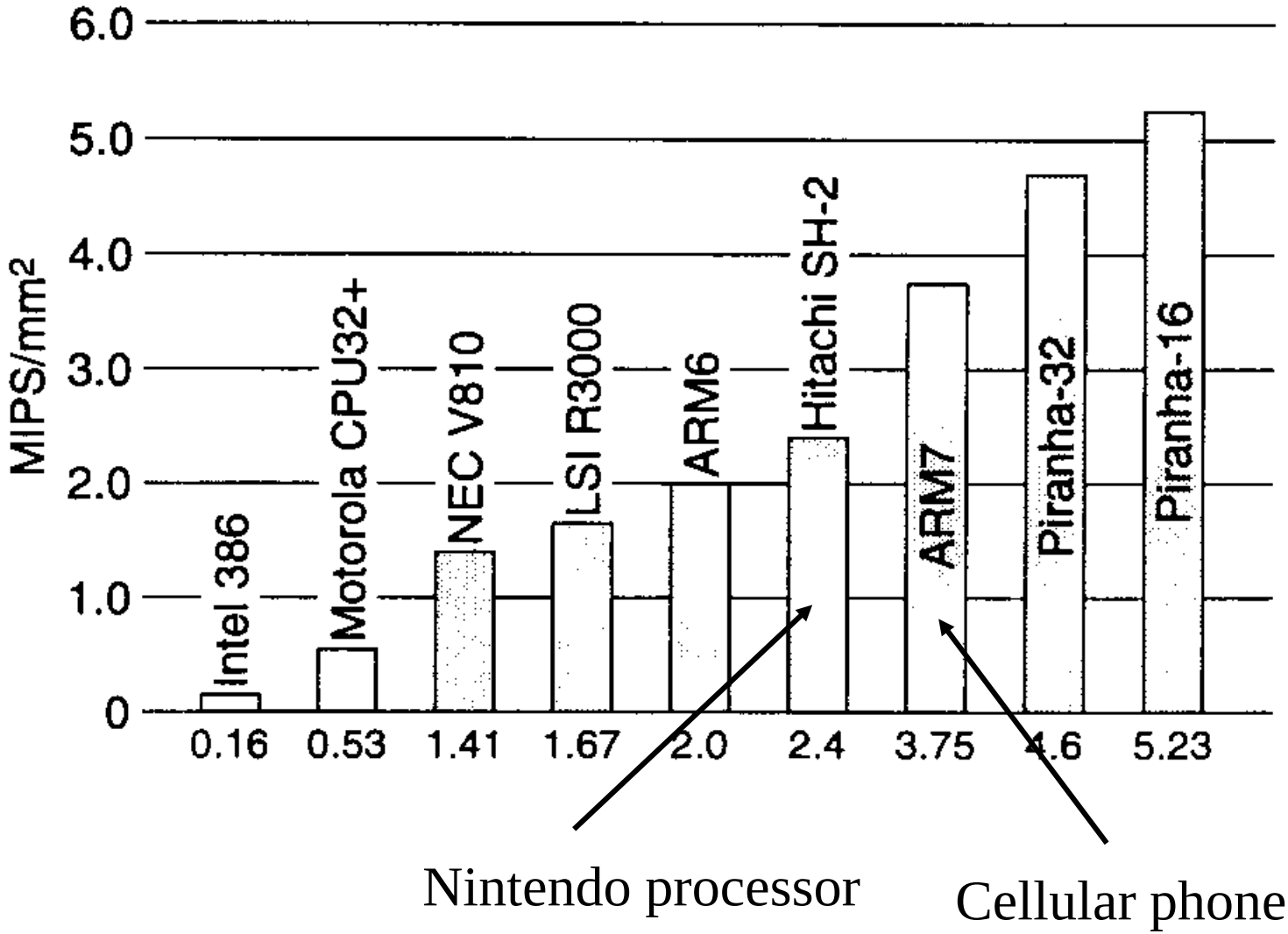
Requirements of the Embedded Processors

- 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

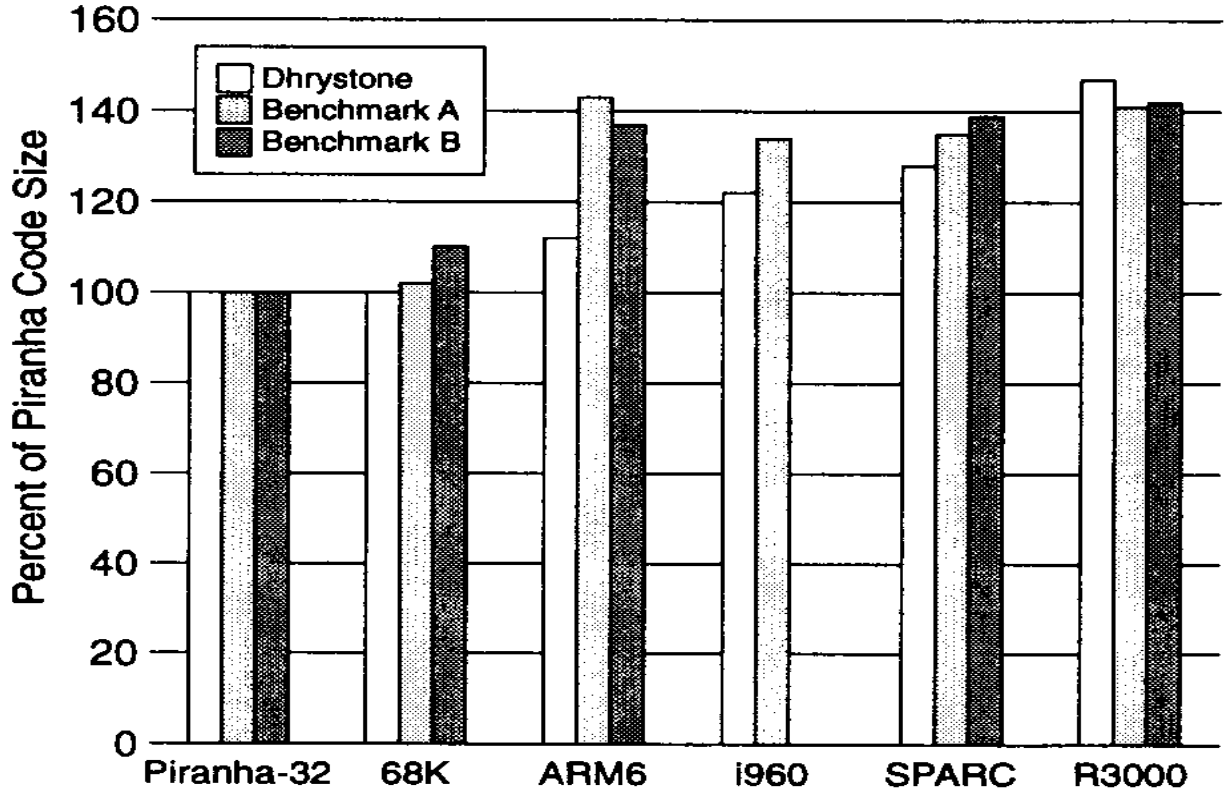
Area of processor cores = Cost



Another figure of merit: Computation per unit area



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**

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 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
- 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:
  - Inifinite Impule Response (IIR) filters
  - Finite Impule Response (FIR) filters
  - FFT, and
  - convolvers
- In DSPs, algorithms are important:
  - Binary compatability 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: 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**

Data Path Comparison

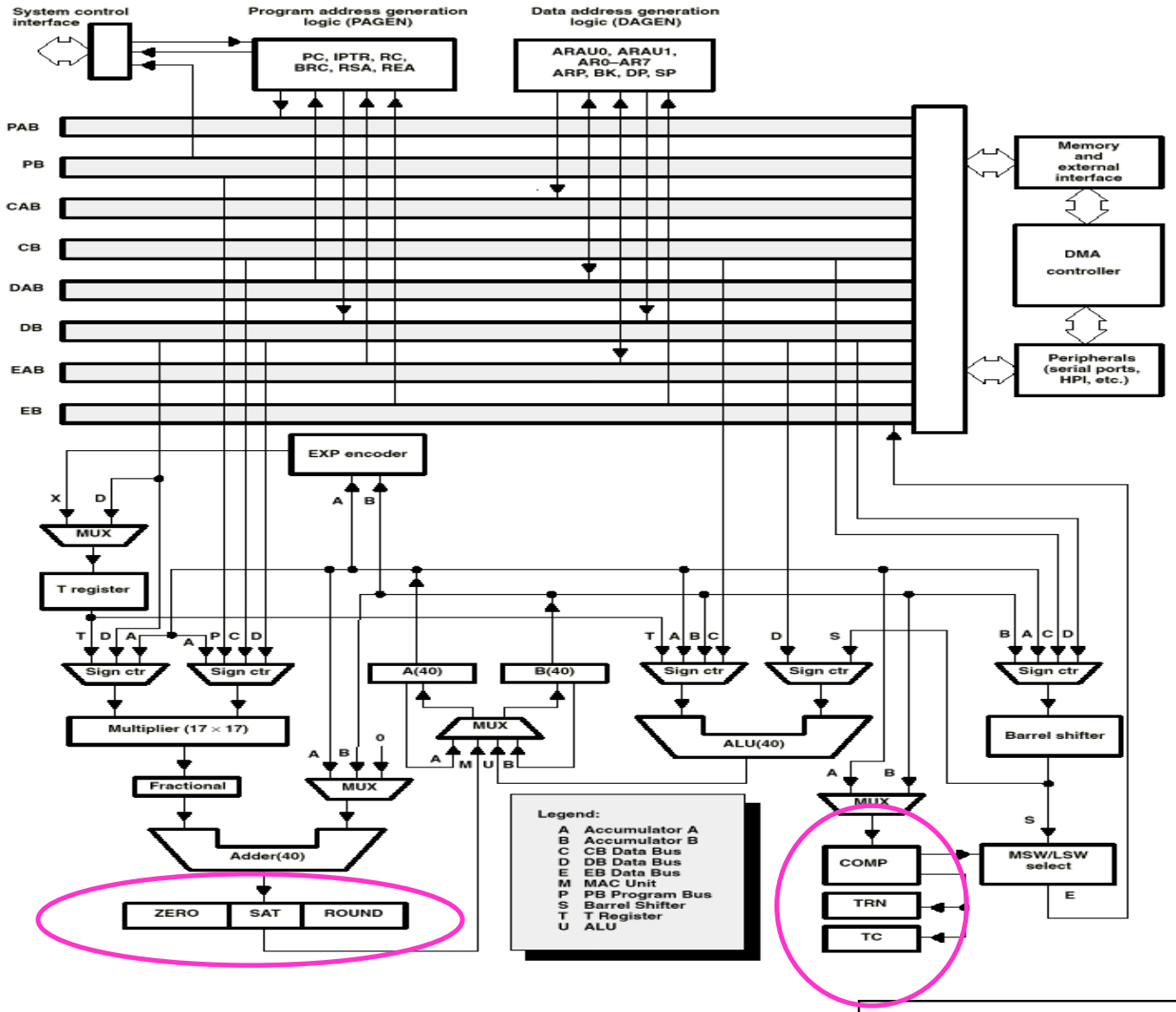
DSP Processor

- **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.**

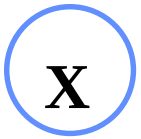
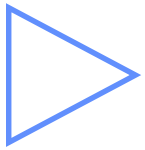
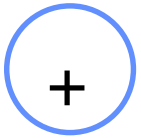
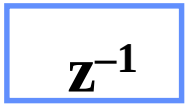
320C54x DSP Functional Block Diagram



DSP Algorithm Format

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

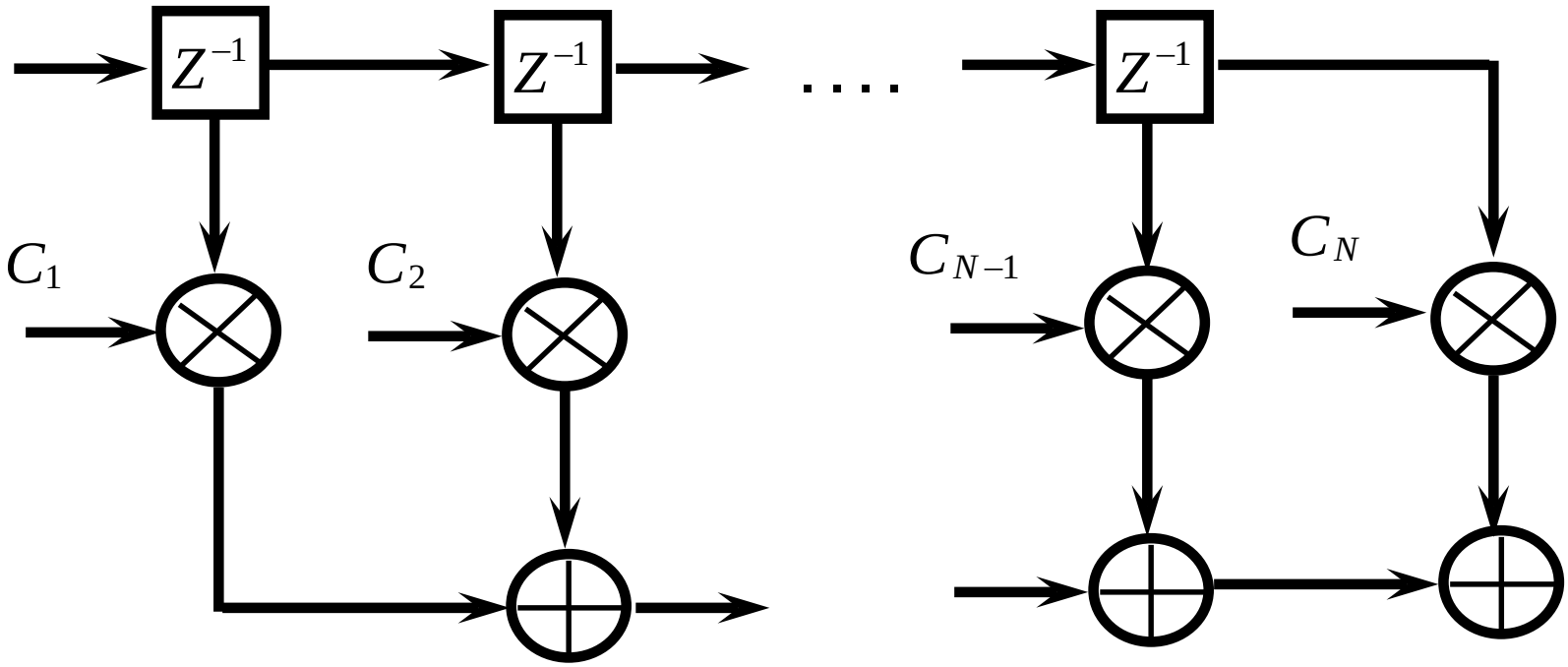
DSP Algorithm Notation

- Uses "flowchart" notation instead of equations
- Multiply is  or 
- Add is  or 
- Delay/Storage is  or  or 

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



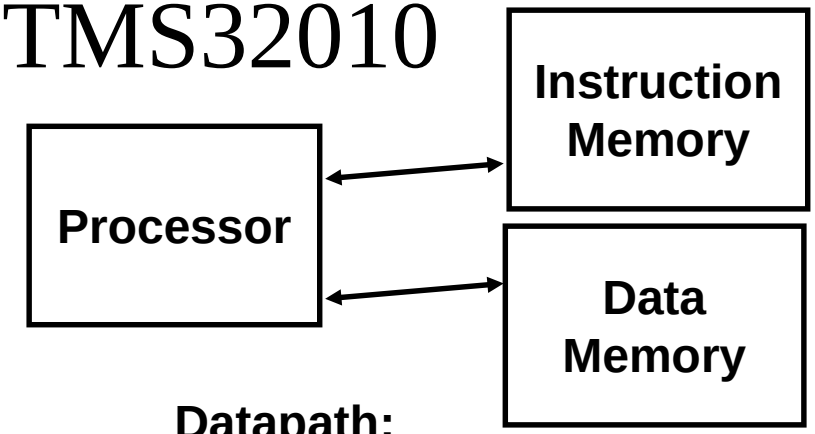
FIR filter on (simple) General Purpose Processor

```
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
```

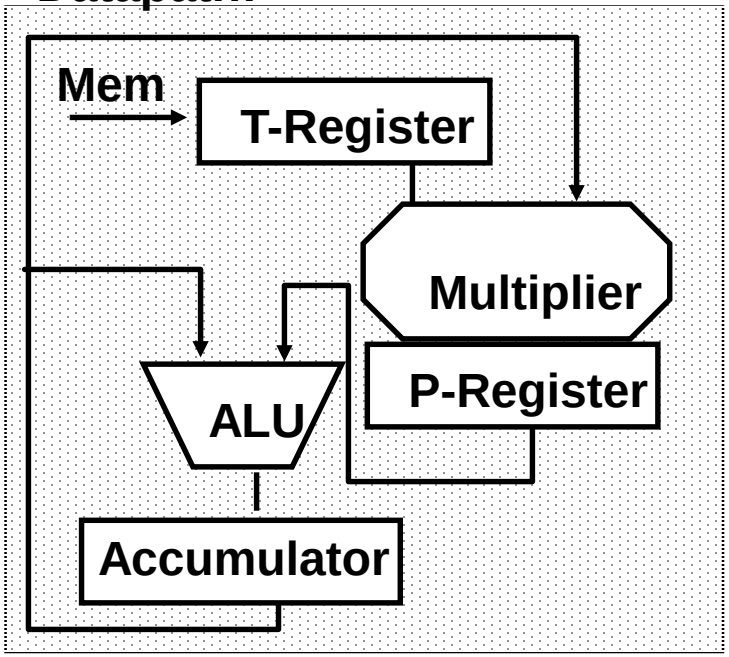
- Problems: Bus / memory bandwidth bottleneck, control code overhead

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**



Datapath:



TMS32010 FIR Filter Code

- Here X4, H4, ... are direct (absolute) memory addresses:

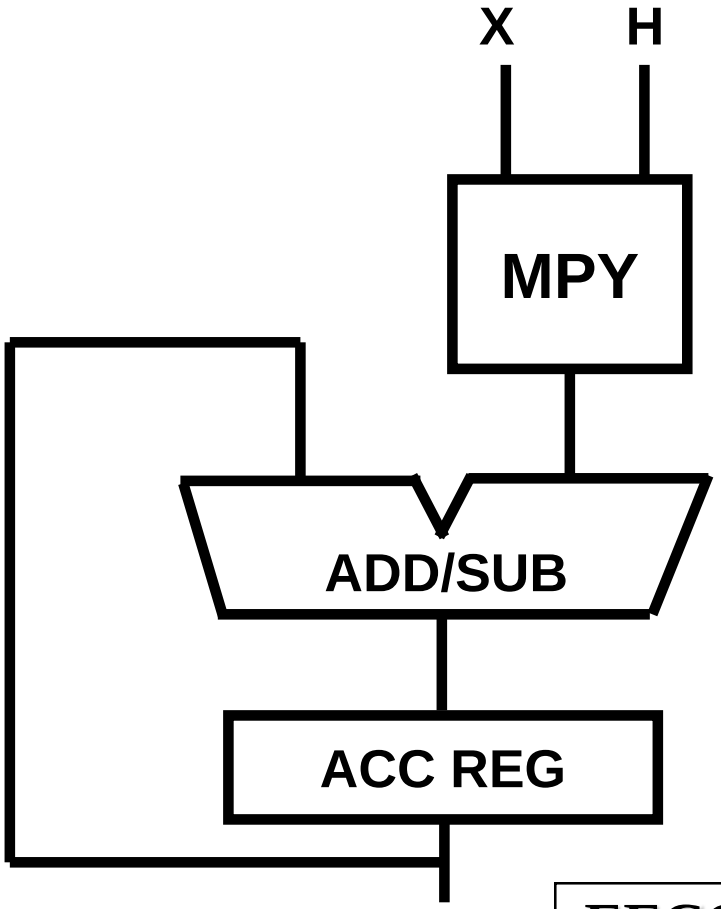
```
LT X4   ; Load T with x(n-4)
MPY H4  ; P = H4*X4
LTD X3  ; Load T with x(n-3); x(n-4) = x(n-3);
        ; Acc = Acc + P
MPY H3  ; P = H3*X3
LTD X2
MPY H2
...
```

- Two instructions per tap, but requires unrolling

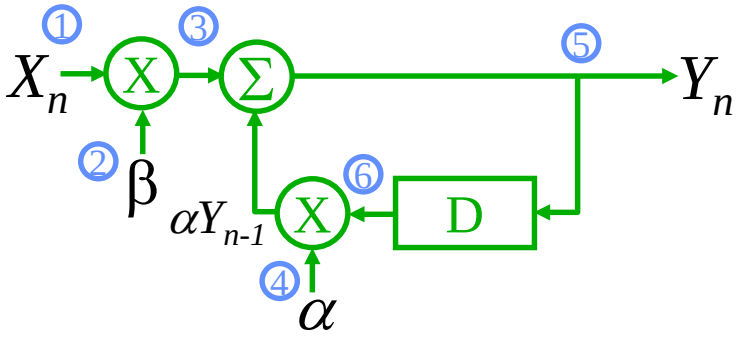
Micro-architectural impact - MAC

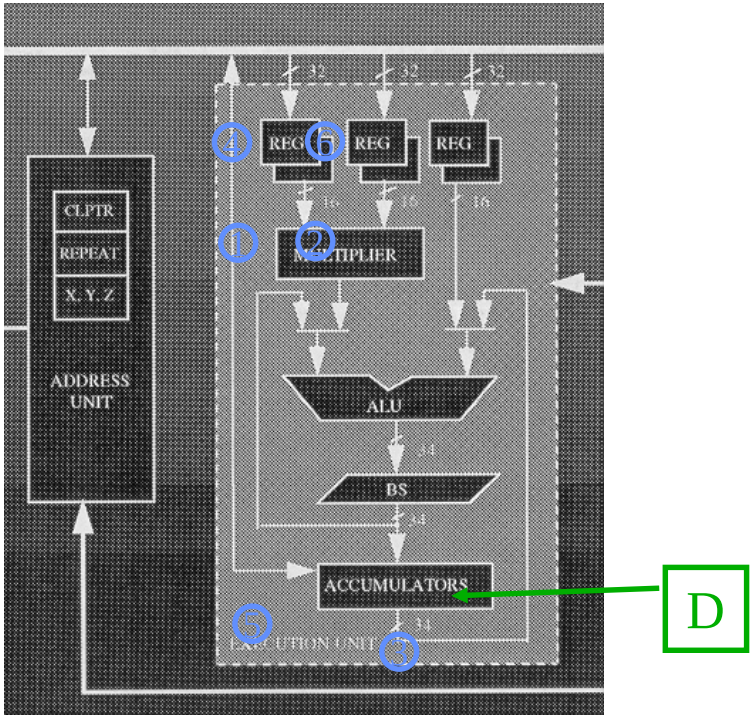
$$y(n) = \sum_{0}^{N-1} h(m)x(n-m)$$

element of finite-impulse response filter computation



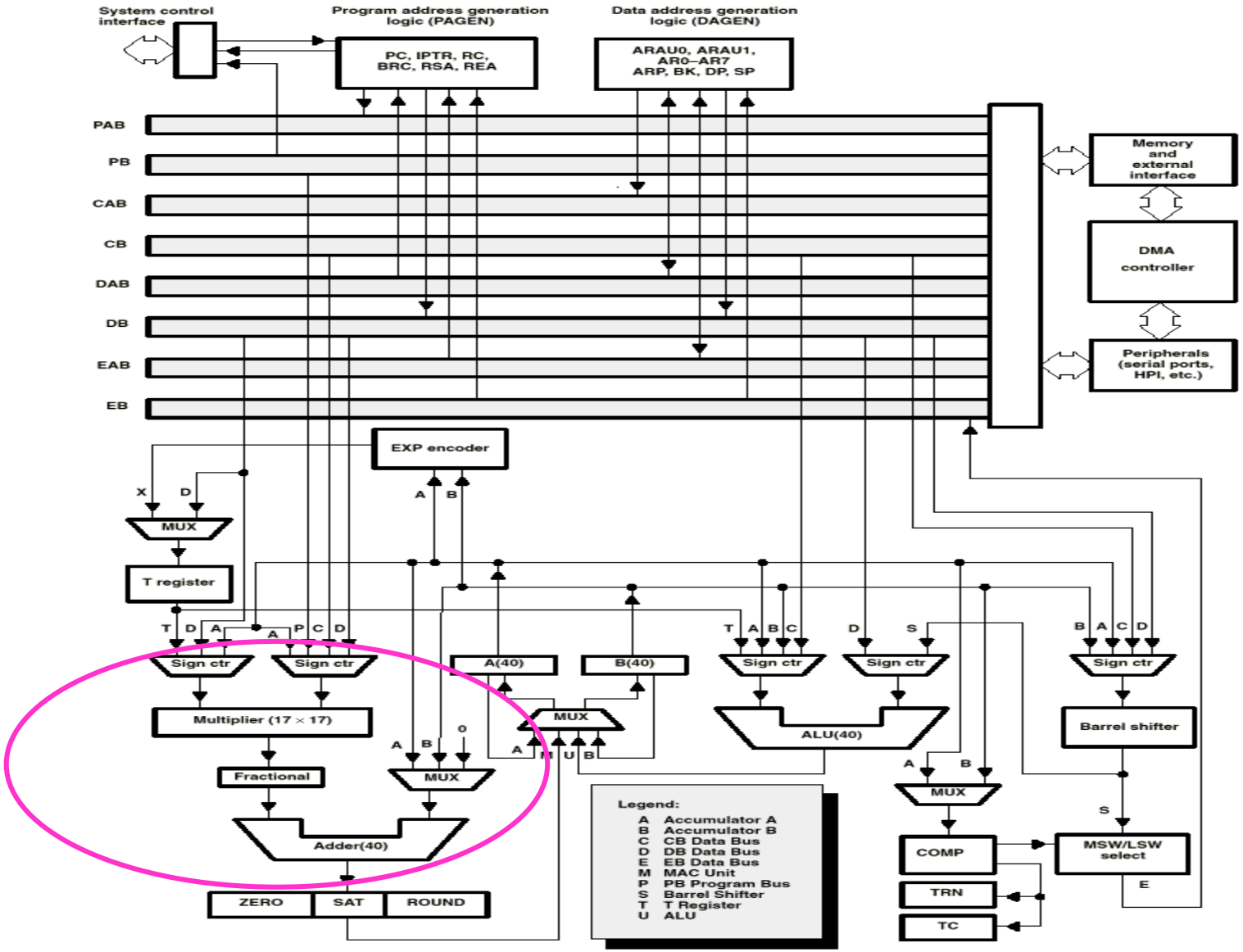
Mapping of the filter onto a DSP execution unit





- **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**

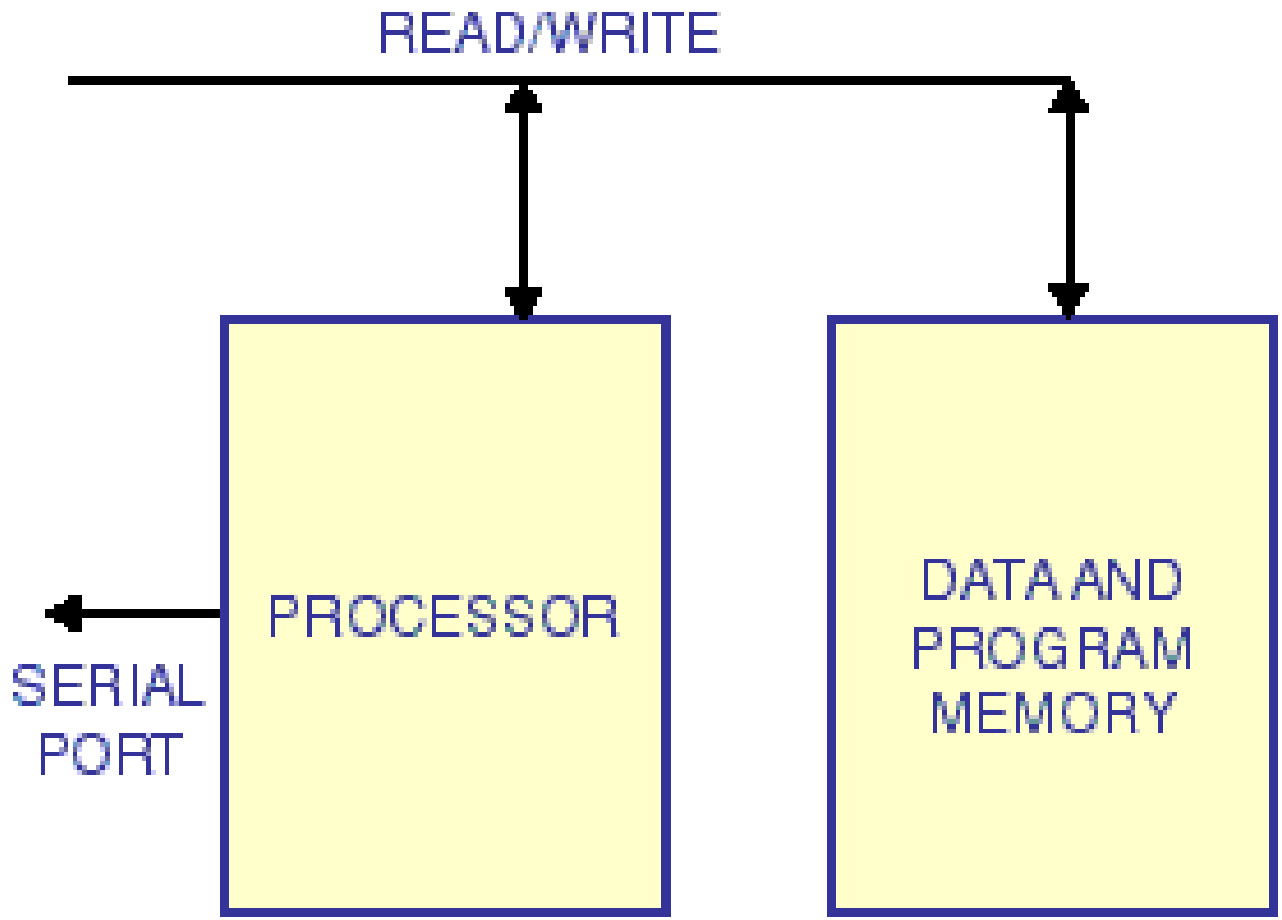
MAC Eg. 320C54x DSP Functional Block Diagram



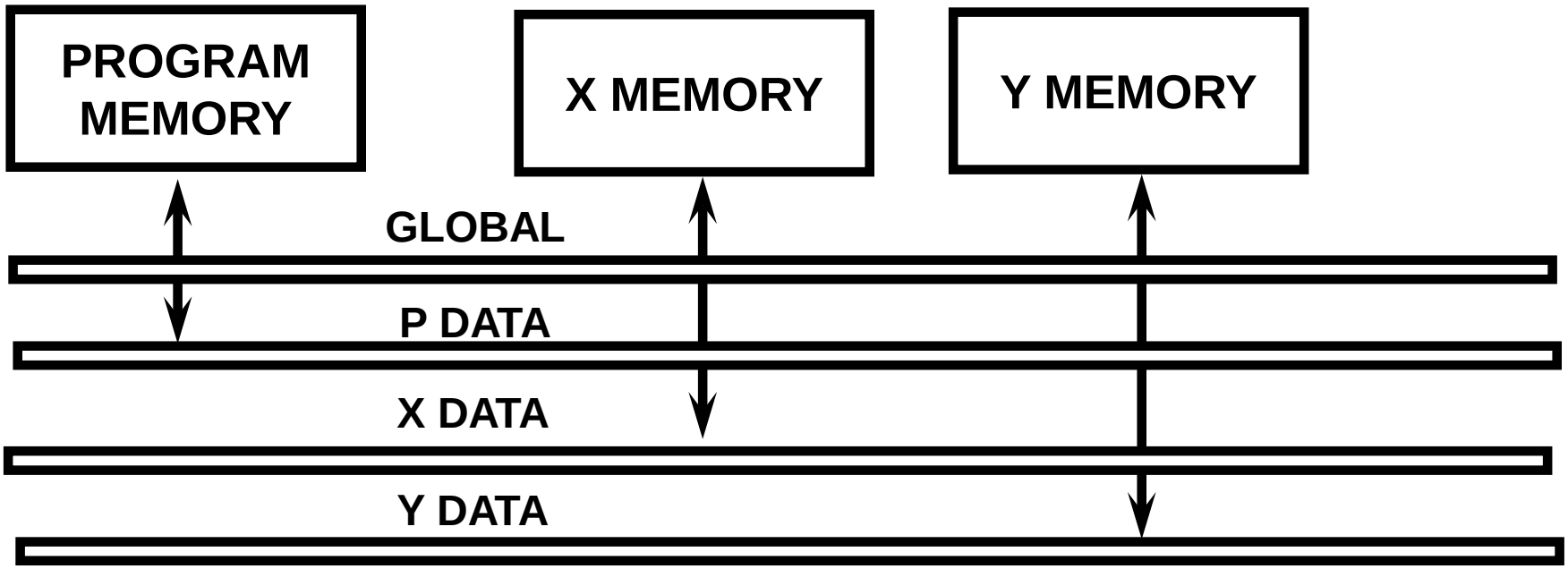
DSP Memory

- 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 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



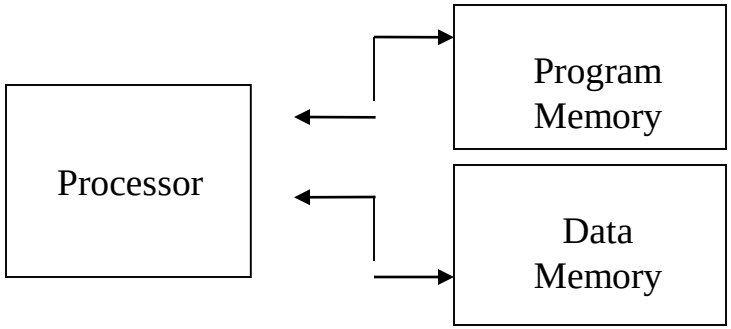
Memory Architecture Comparison

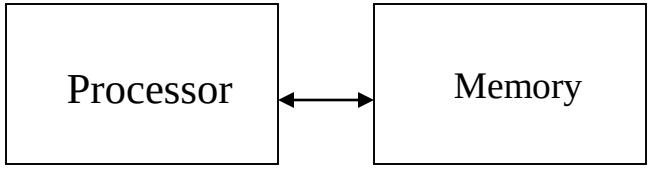
DSP Processor

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

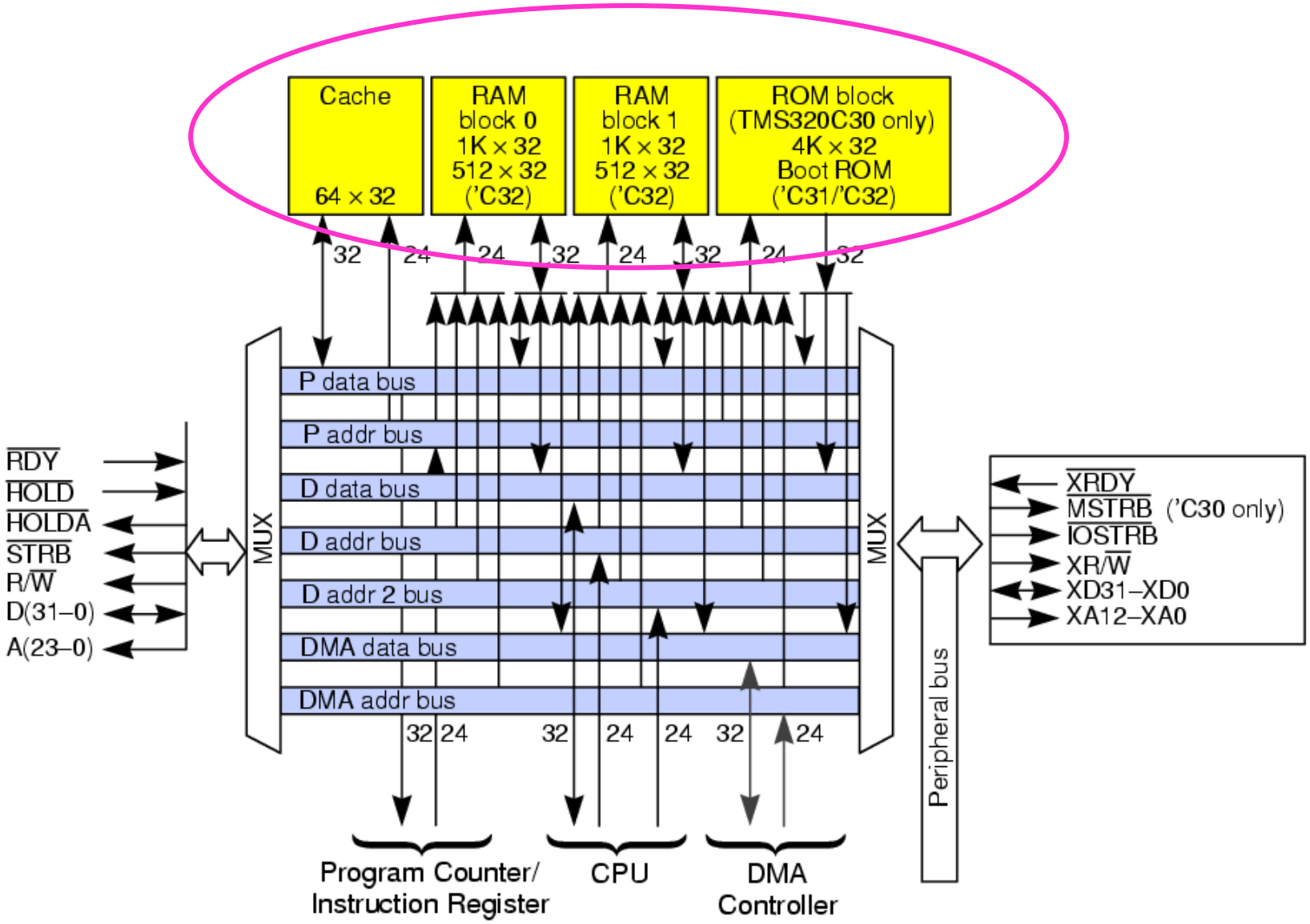
General-Purpose Processor

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

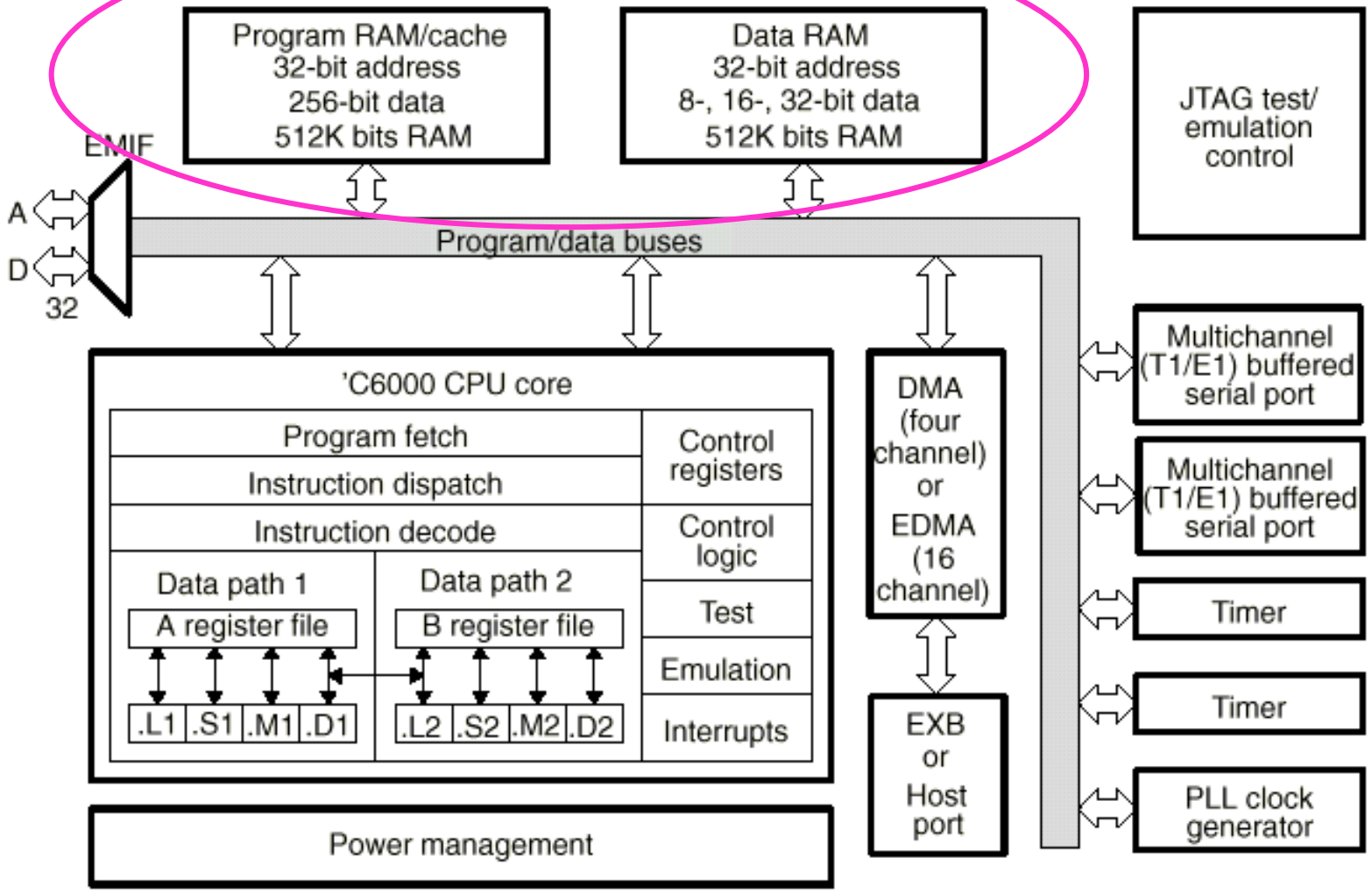




Eg. TMS320C3x MEMORY BLOCK DIAGRAM - Harvard Architecture



Eg. 320C62x/67x DSP



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**

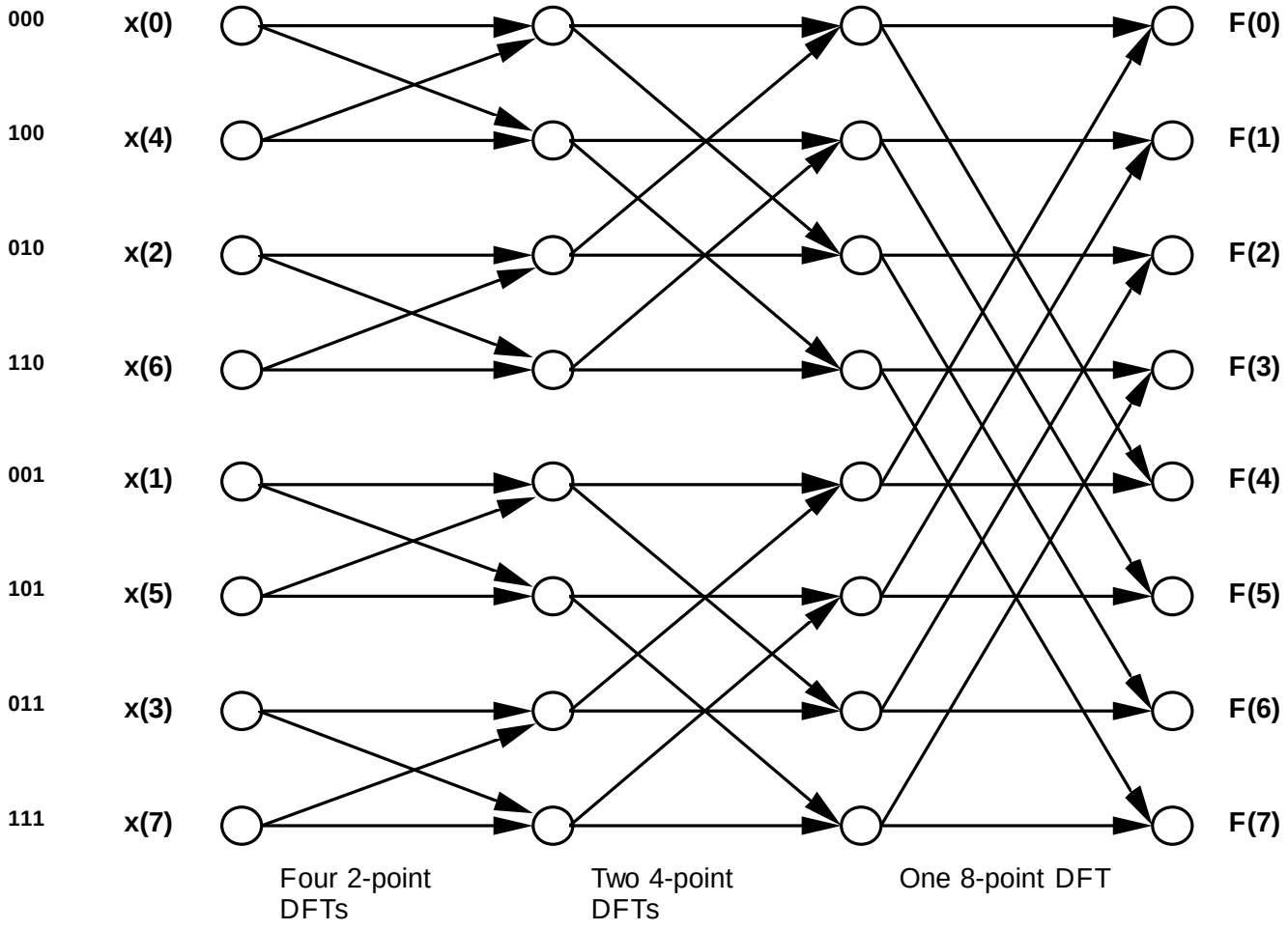
DSP Addressing: FFT

- **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 "modulo" or "circular" addressing

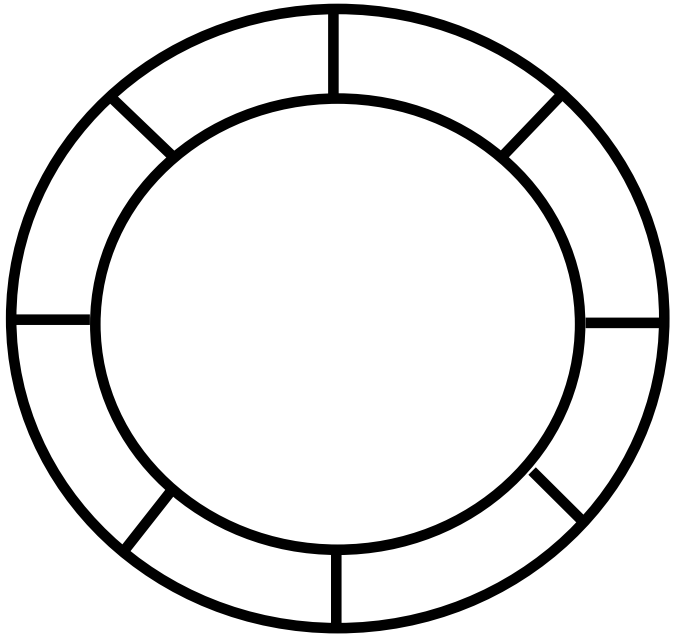
CIRCULAR BUFFERS

Instructions accomodate three elements:

- **buffer address**
- **buffer size**
- **increment**

Allows for cyling through:

- **delay elements**
- **coefficients in data memory**



Addressing Comparison

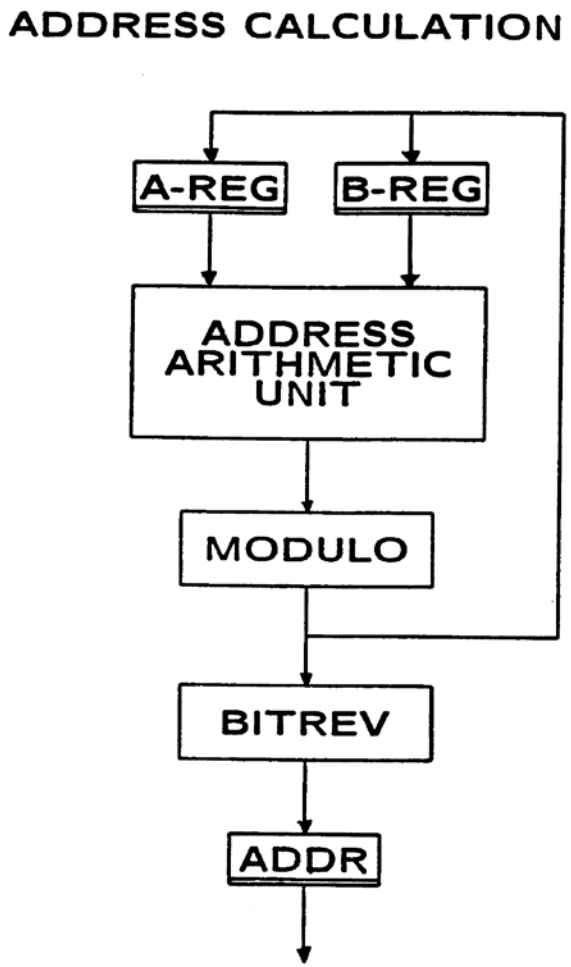
DSP Processor

- **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

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**

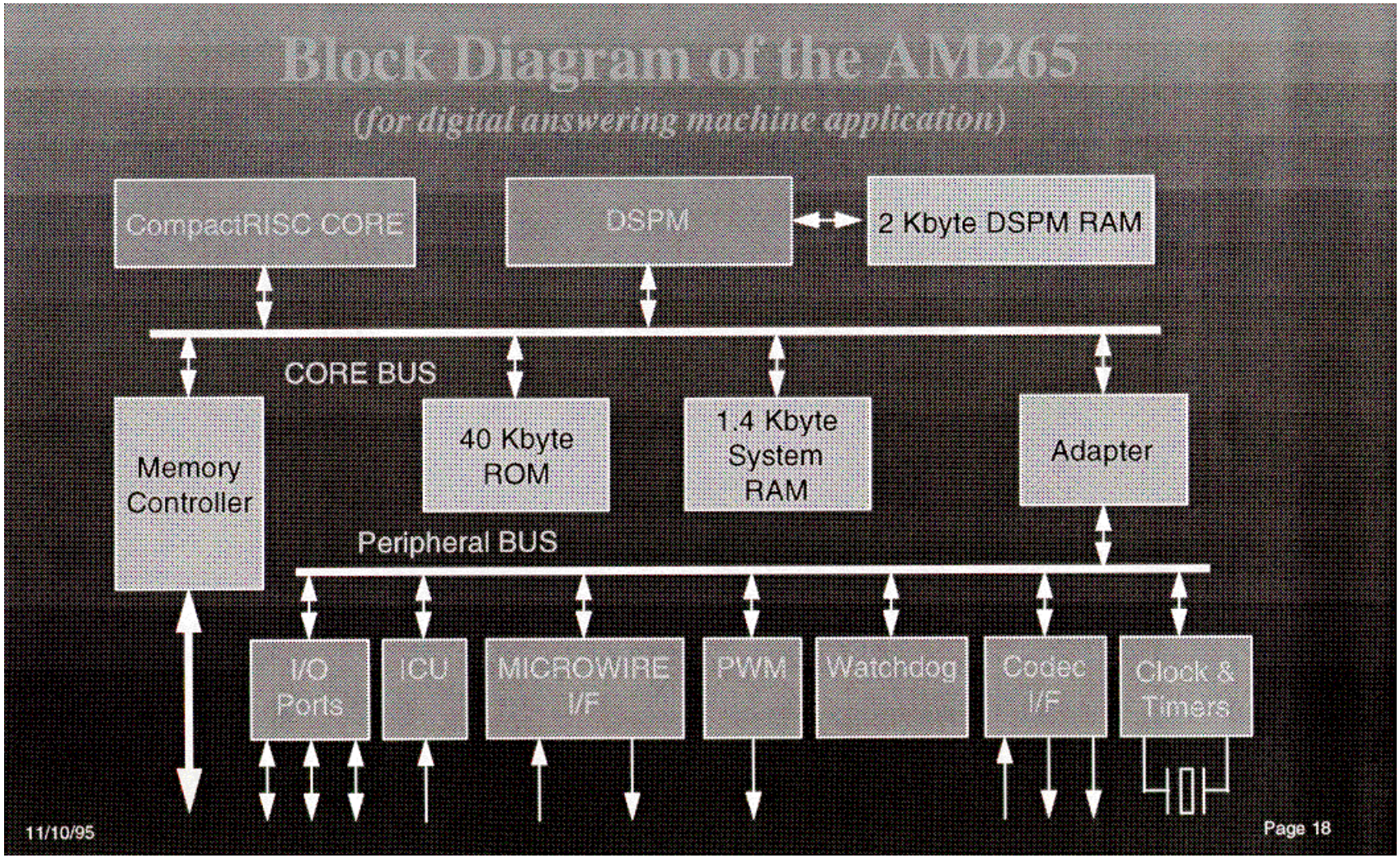
```
mov *r0,x0
mov *r1,y0
mpy x0, y0, a
add a, b
mov y0, *r2
inc r0
inc rl
```

Specialized Peripherals for DSPs

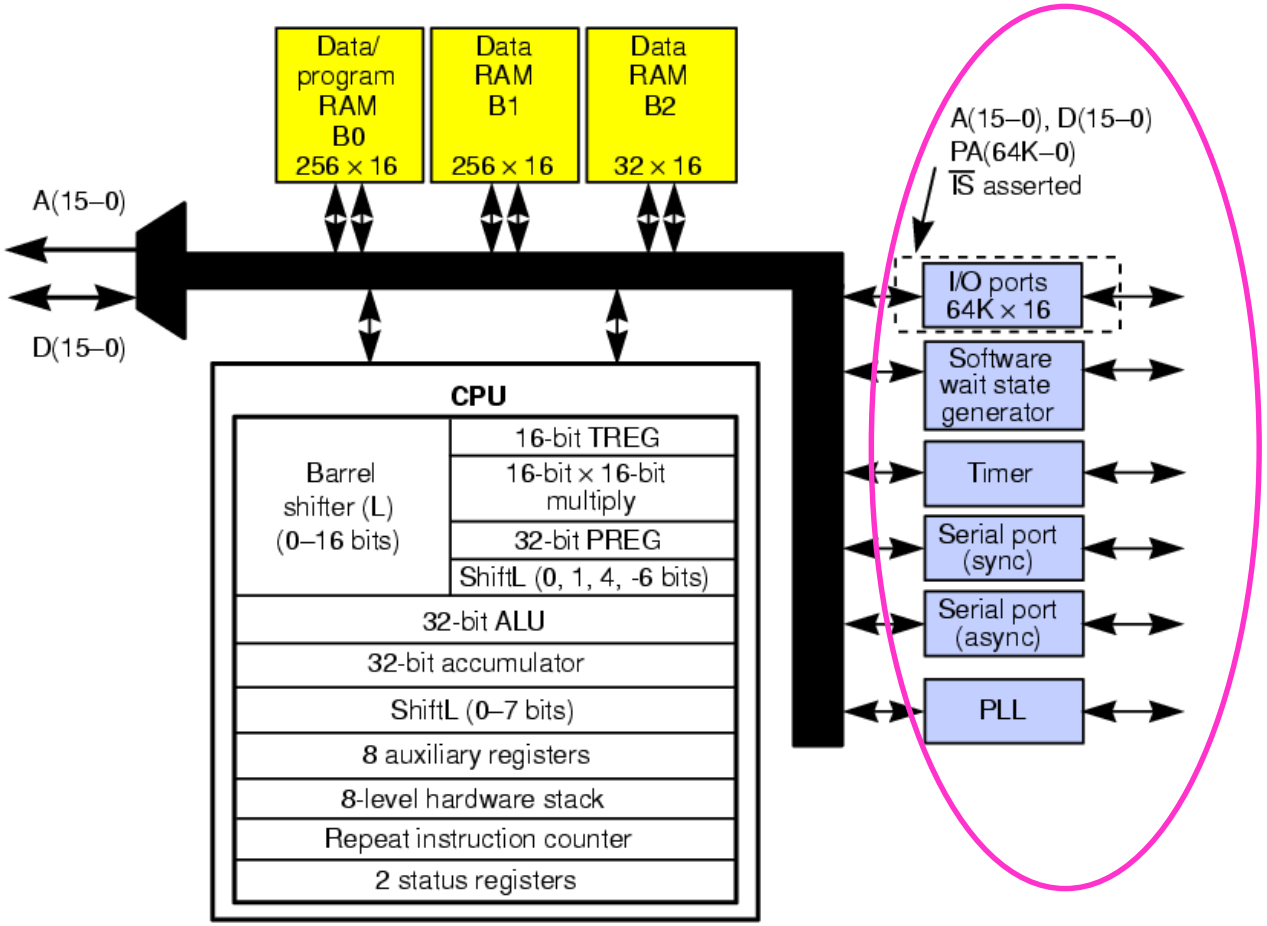
- **Synchronous serial ports**
- **Parallel ports**
- **Timers**
- **On-chip A/D, D/A converters**
- **Host ports**
- **Bit I/O ports**
- **On-chip DMA controller**
- **Clock generators**

- On-chip peripherals often designed for “background” operation, even when core is powered down.

Specialized DSP peripherals



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.***

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

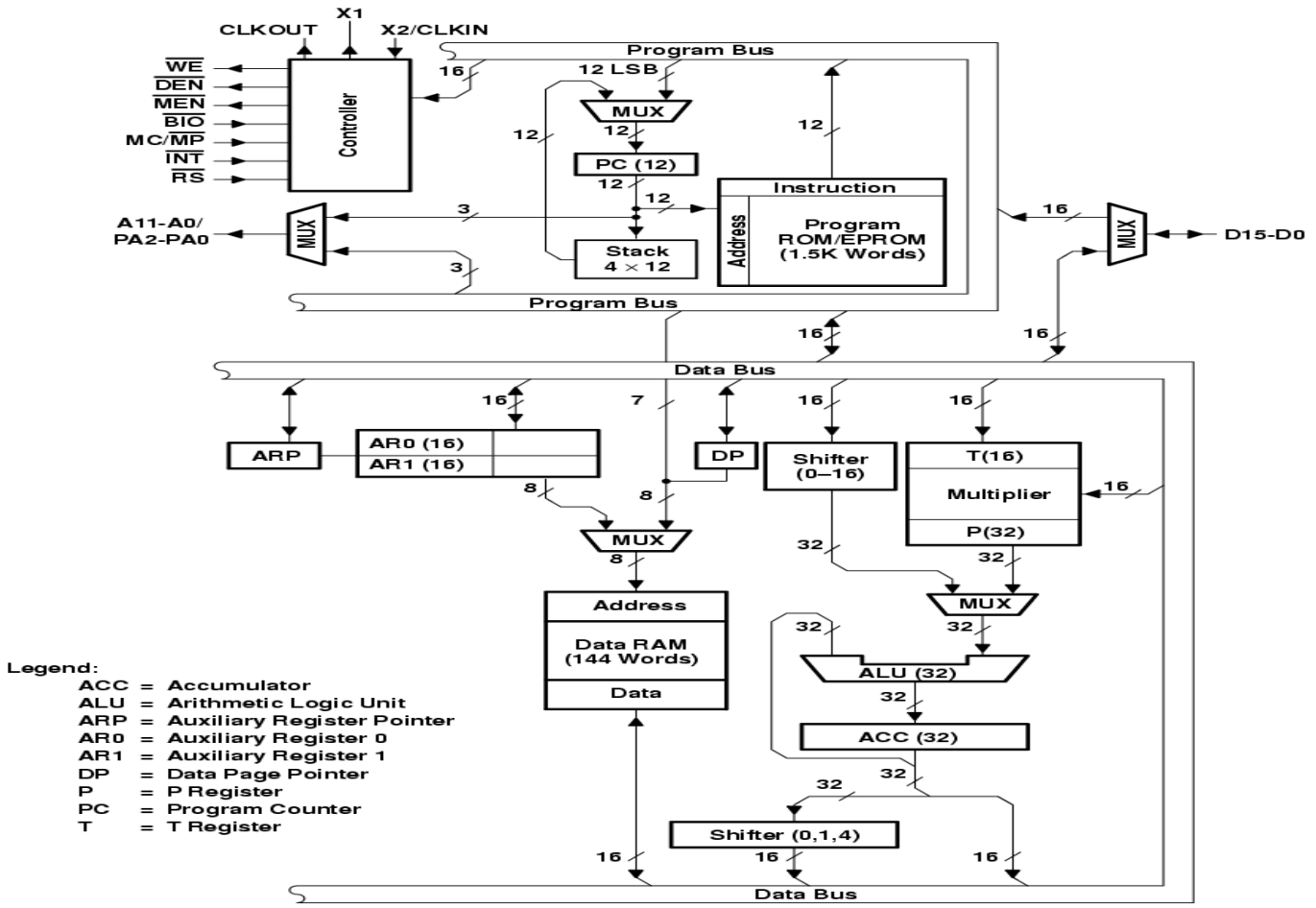
First Generation DSP μP Case Study

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

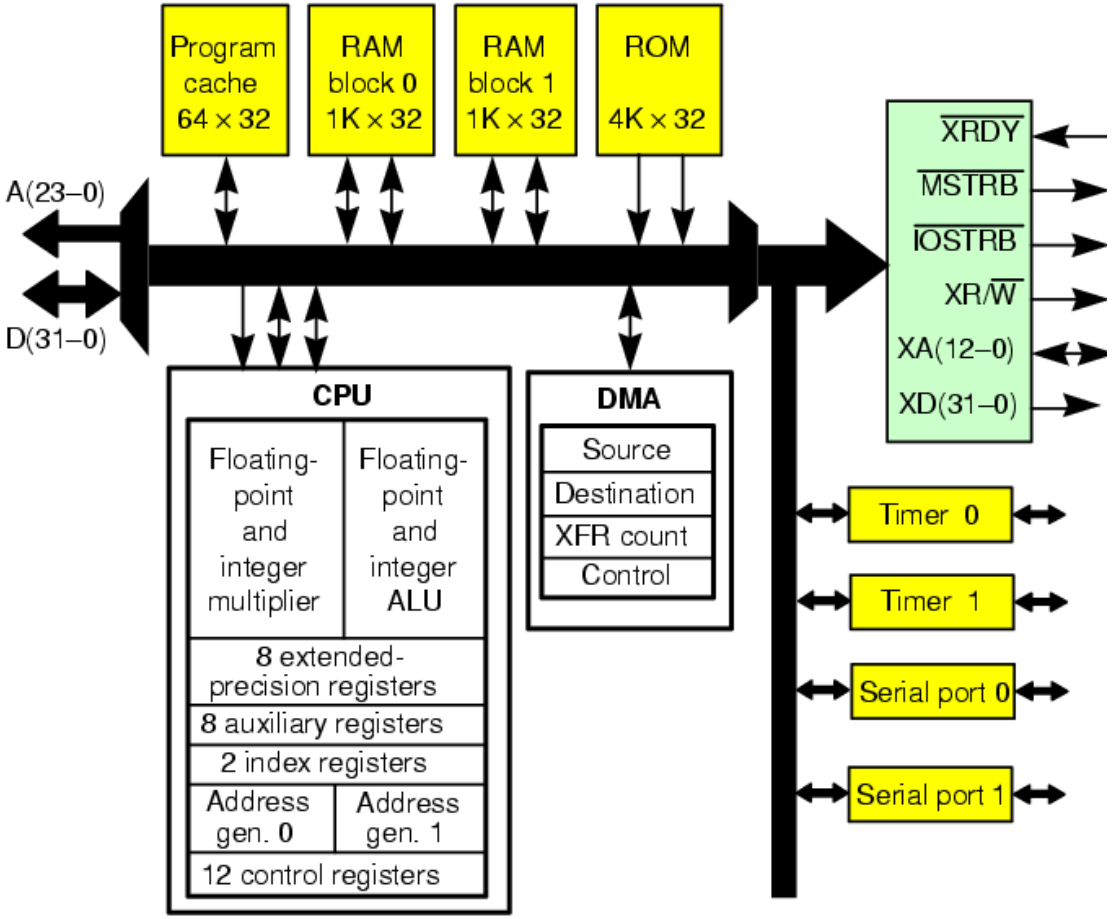


Third Generation DSP μP Case Study
TMS320C30 - 1988

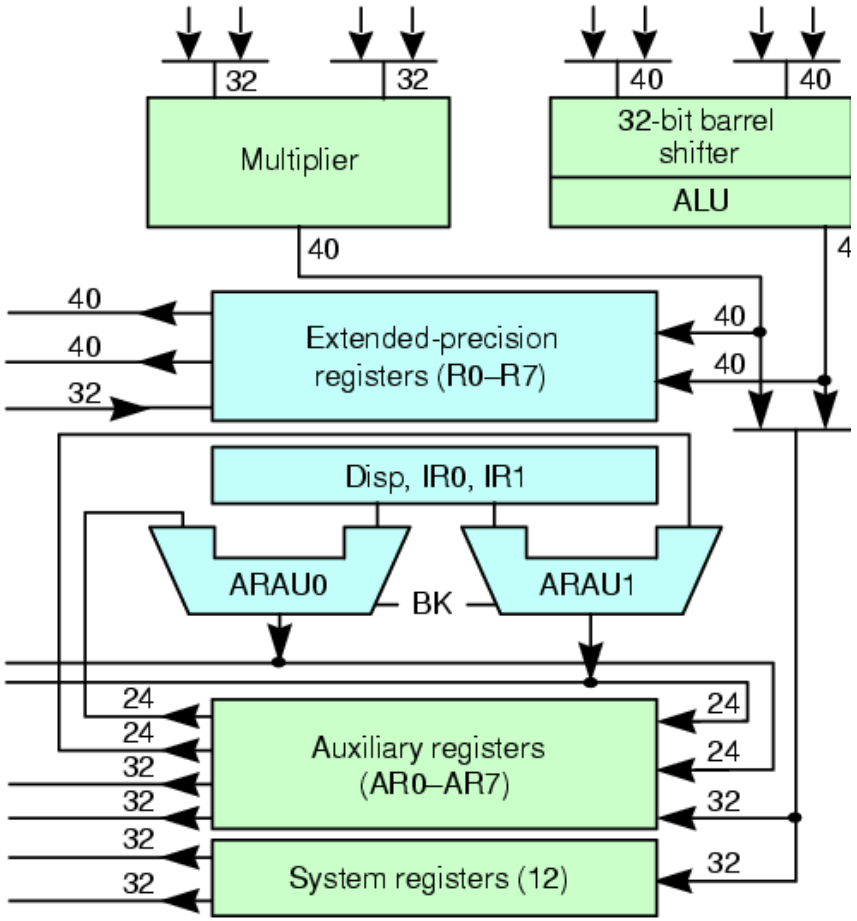
TMS320C30 Key Features

- **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**

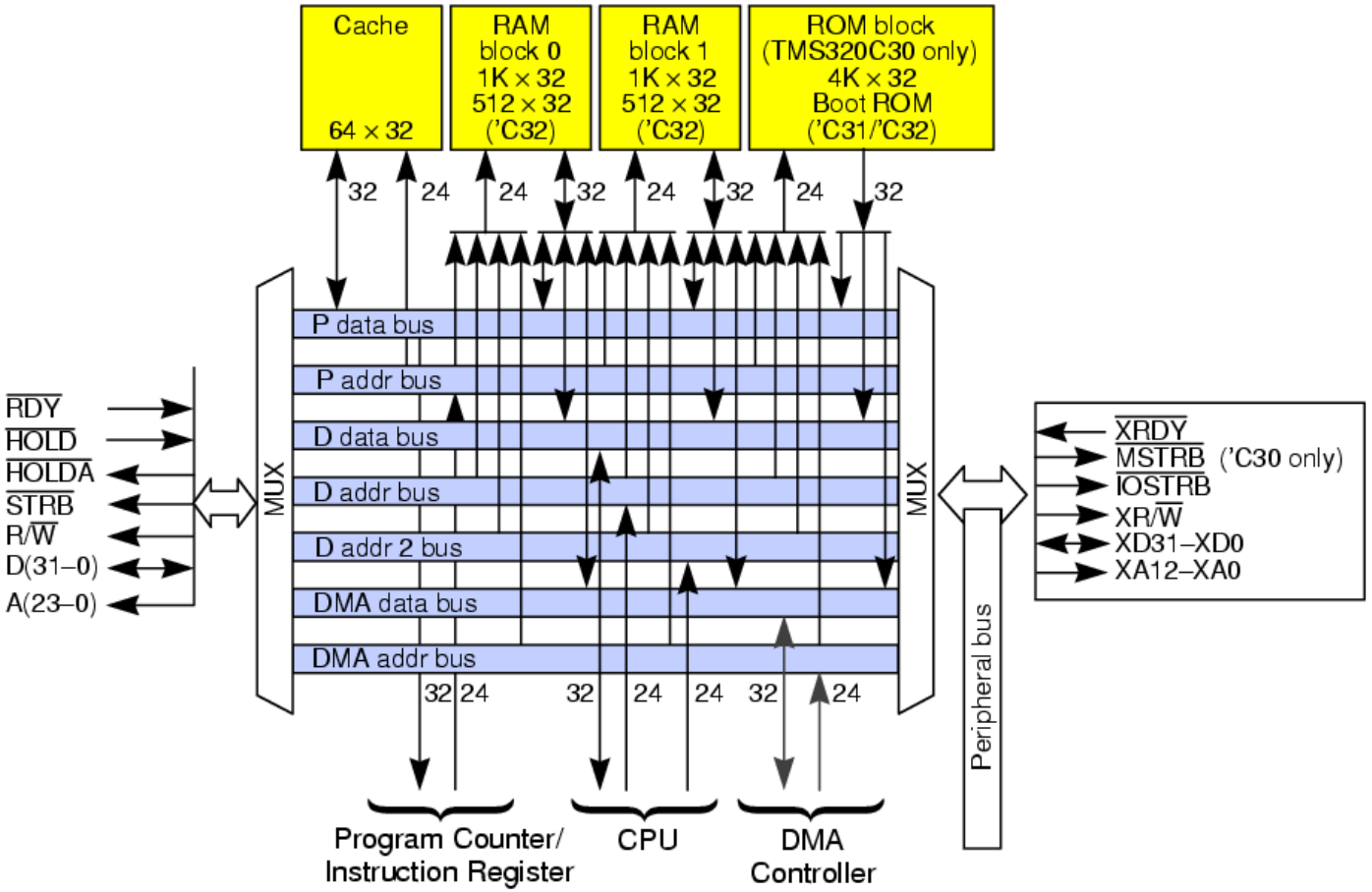
TMS320C30 BLOCK DIAGRAM



TMS320C3x CPU BLOCK DIAGRAM



TMS320C3x MEMORY BLOCK DIAGRAM



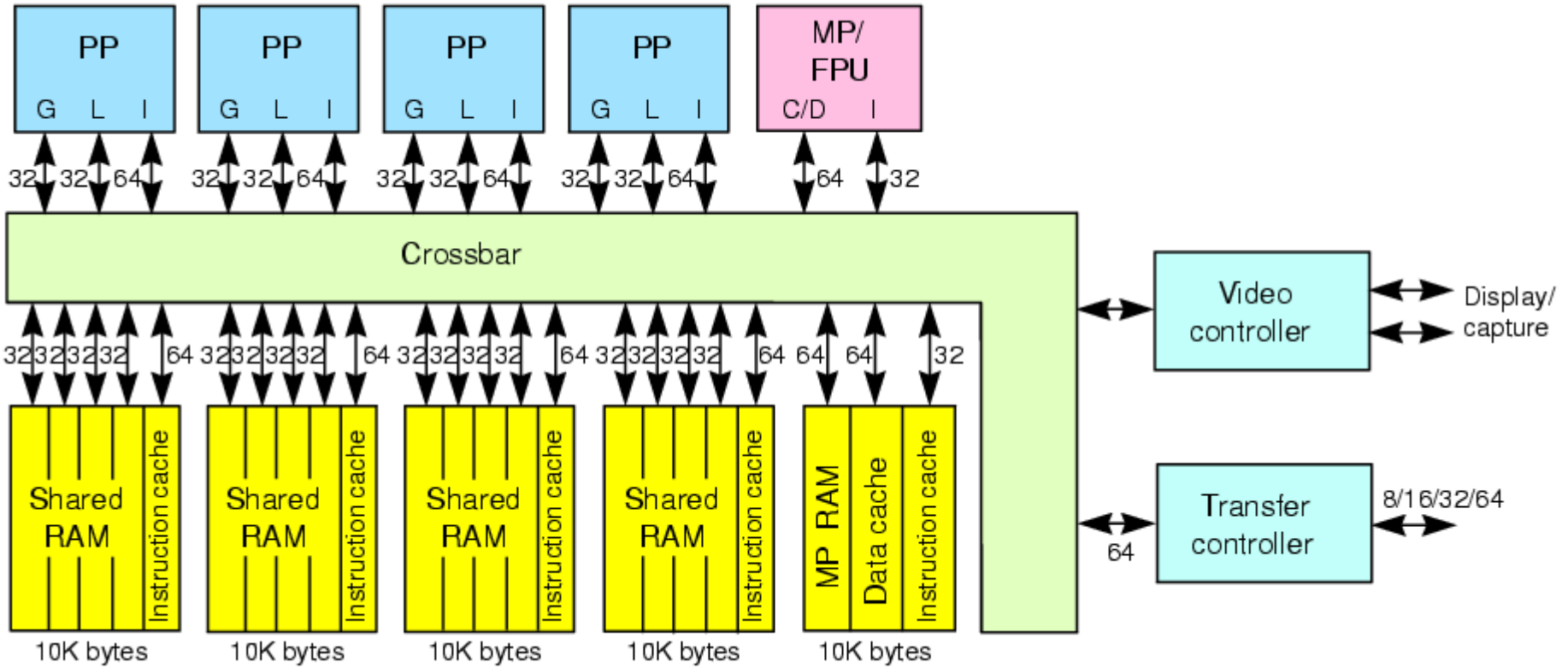
TMS320C30 FIR FILTER PROGRAM

Y(n) = x[n-(N-1)] . h(N-1) + x[n-(N-2)] . h(N-2) +…+ x(n) . h(0)

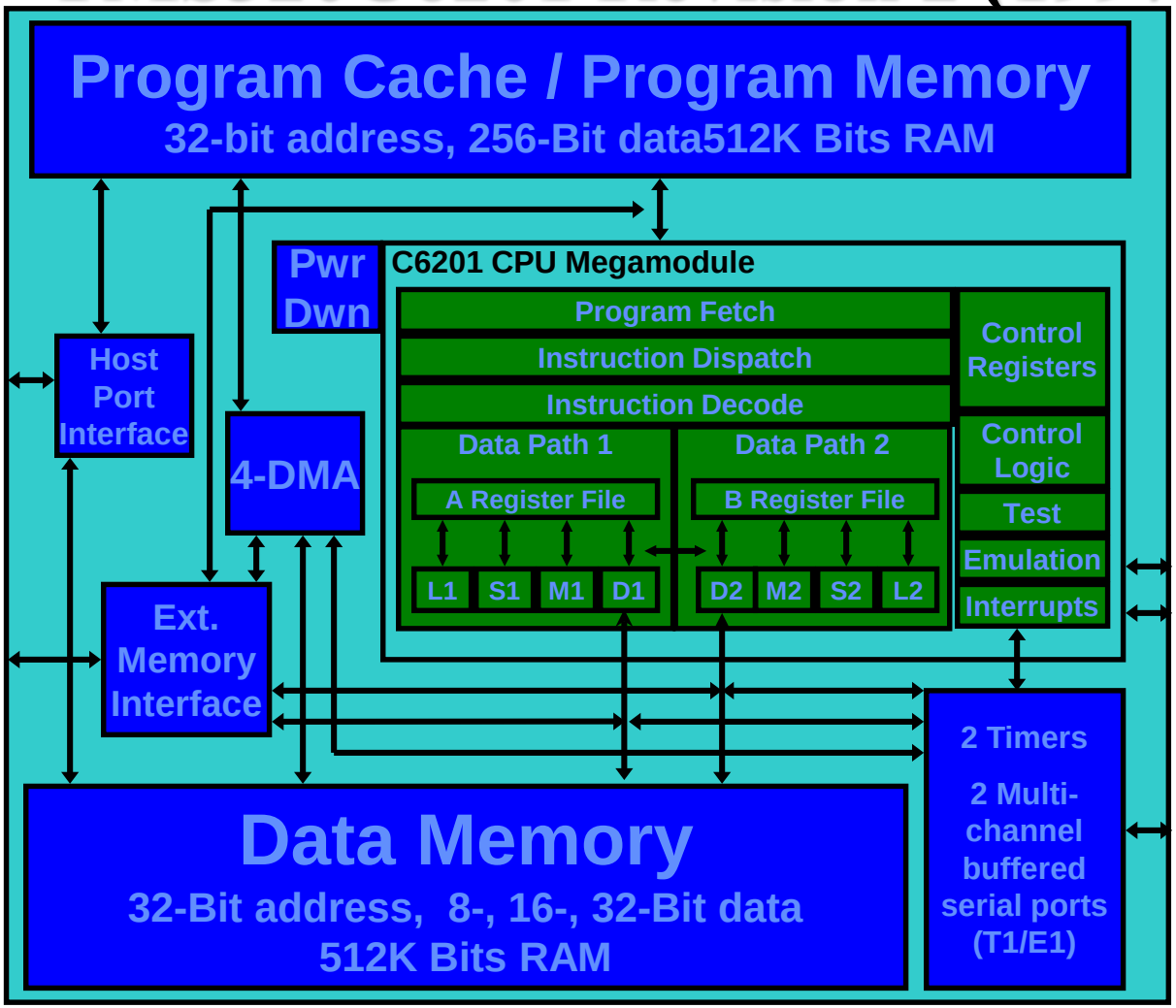
```
TOP     LDF     IN,R3               ;Read input sample.
        STF     R3,*AR1++%          ;Store with other samples,
                                    ;and point to top of buffer.
        LDF     0,R0                ;Initialize R0.
        LDF     0,R2                ;Initialize R2.
*
*       Filter
*
        RPTS    N-1                 ;Repeat next instruction.
        MPYF3   *AR0++%,*AR1++%,R0
||      ADDF3   R0,R2,R2            ;Multiply and accumulate.
        ADDF    R0,R2               ;Last product accumulated.
*
        STF     R2,Y                ;Save result.
        B       TOP                 ;Repeat.
```

For N=50, t=3.6 μs (277 KHz)

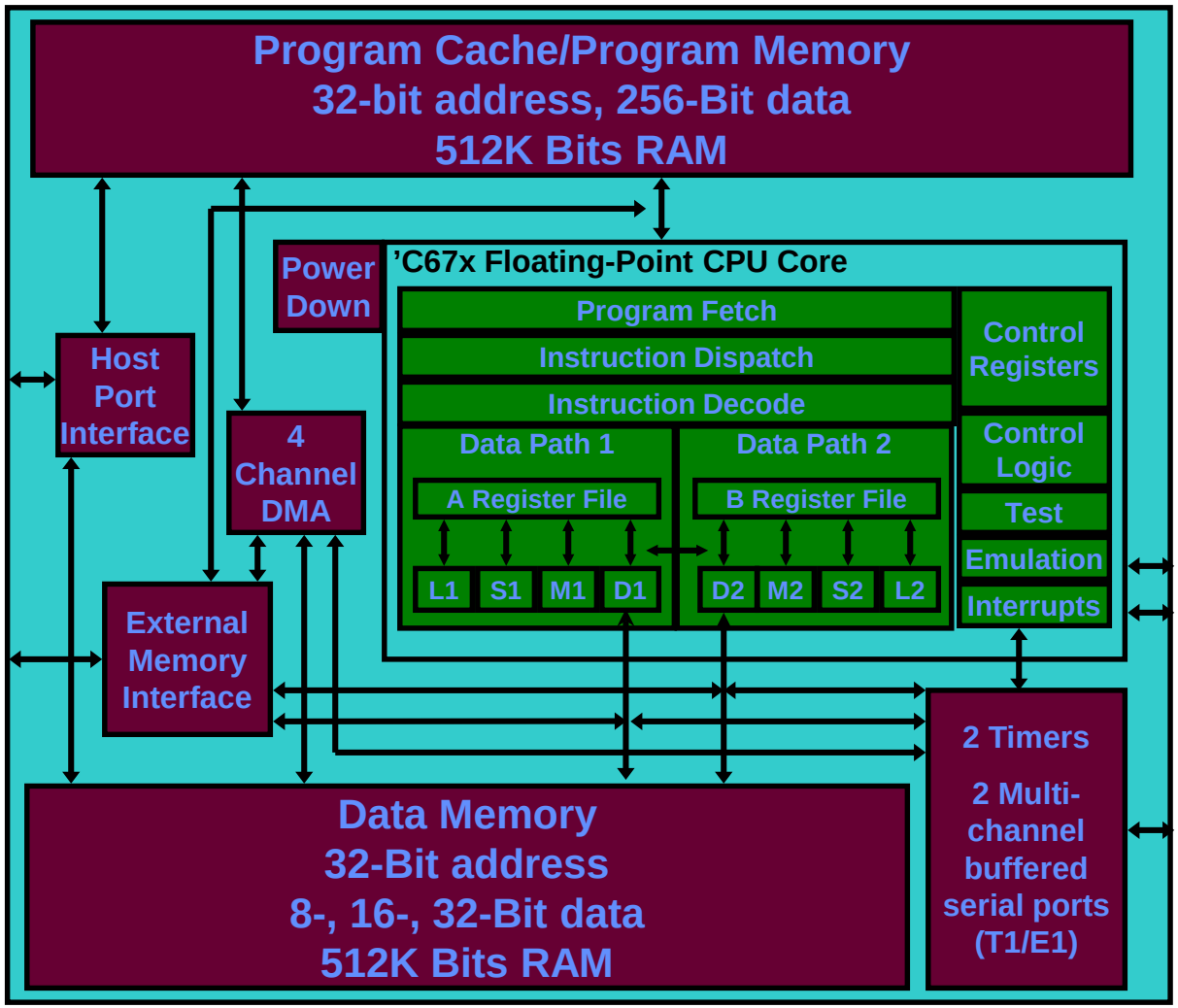
Texas Instruments TMS320C80
MIMD MULTIPROCESSOR DSP (1996)



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



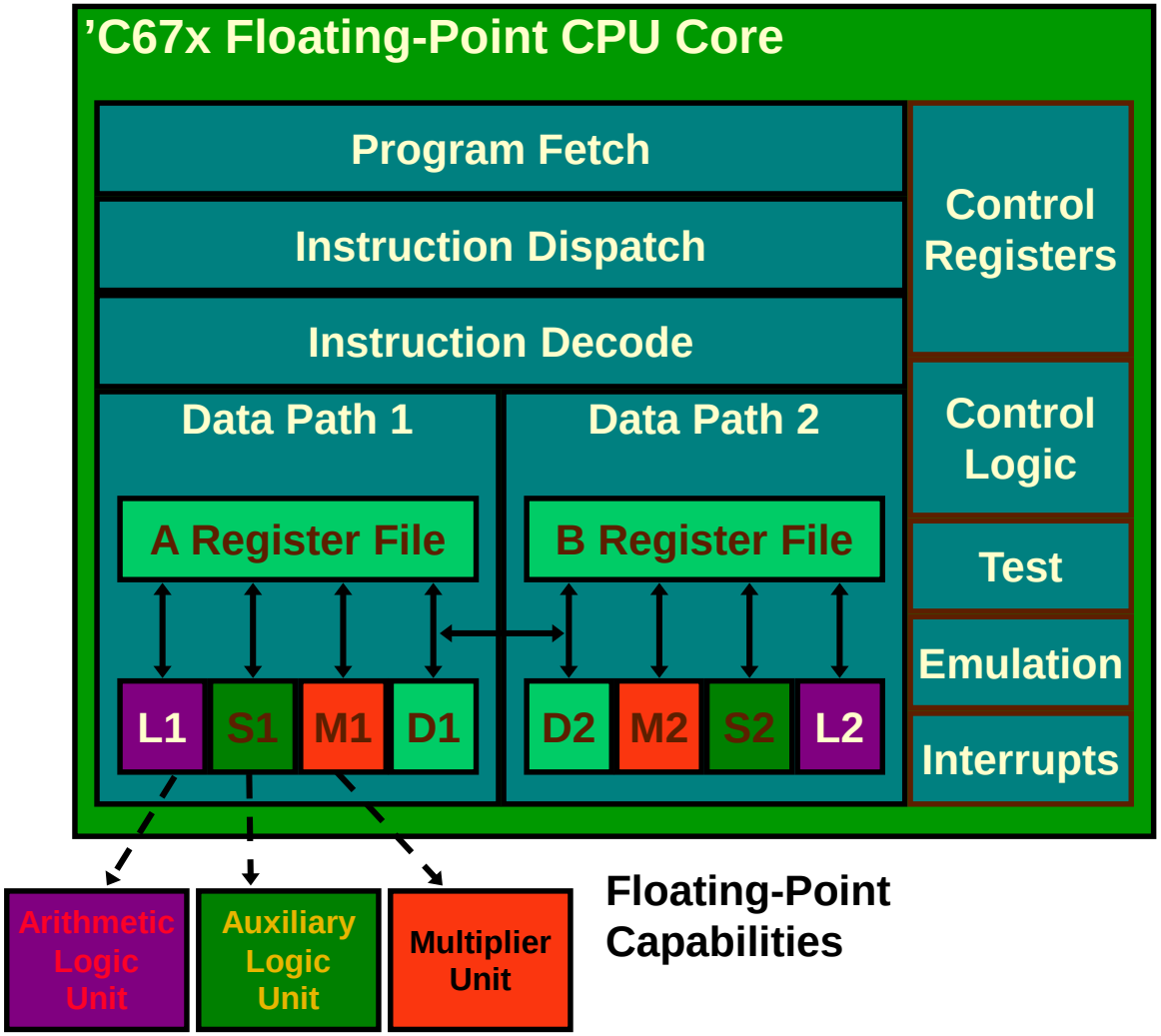
32 Bit Floating Point VLIW DSP: TMS320C6701 (1997)



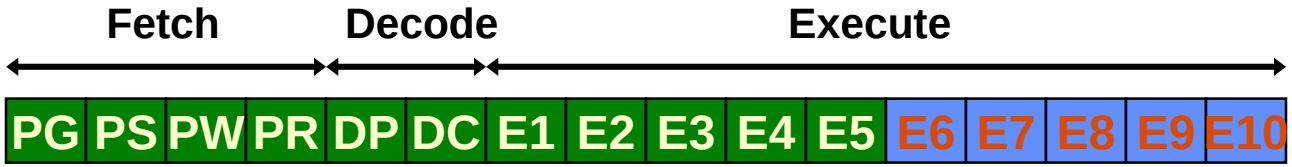
TMS320C6701
Advanced VLIW CPU (VelociTI™)

- **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 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

C55x bloc funcional

1.3 Functional Block Diagram

Figure 1-1 shows the functional block diagram of the device.

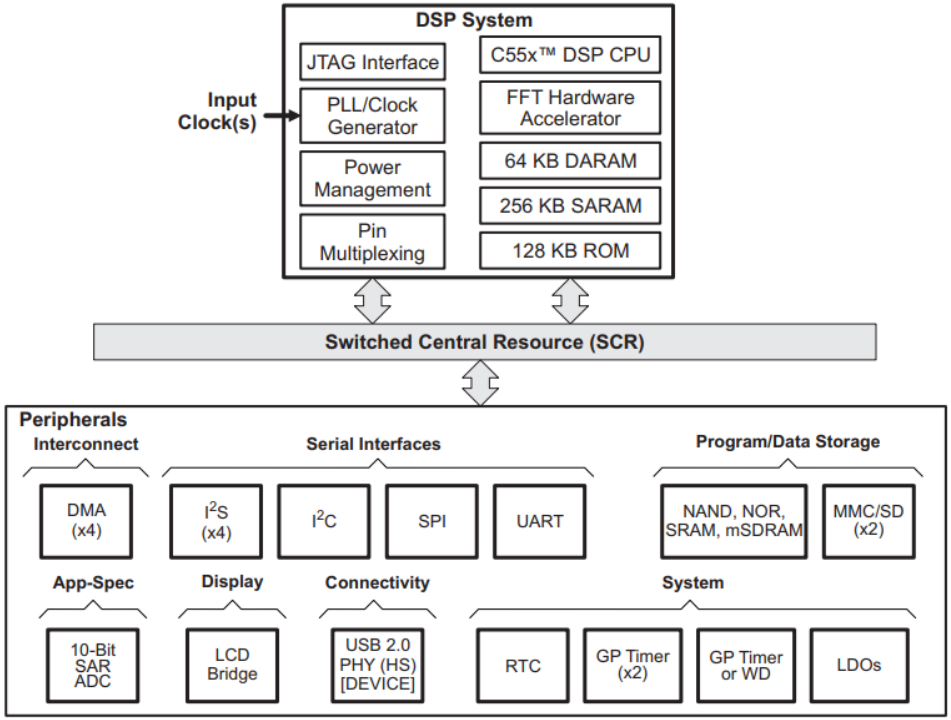


Figure 1-1. Functional Block Diagram

1.1 Overview of the CPU Architecture

Figure 1–1 shows a conceptual block diagram of the CPU. Sections 1.1.1 through 1.1.6 describe the buses and units represented in the figure.

Figure 1–1. CPU Diagram