Today's Agenda

- ✔ What are my system requirements?
- ✔ How do I work with TI's 'C6000?
- ✔ How do I work with TI's 'C5000?
- How do TI's tools make my development easier?
- What support can I count on?

TMS320C5000

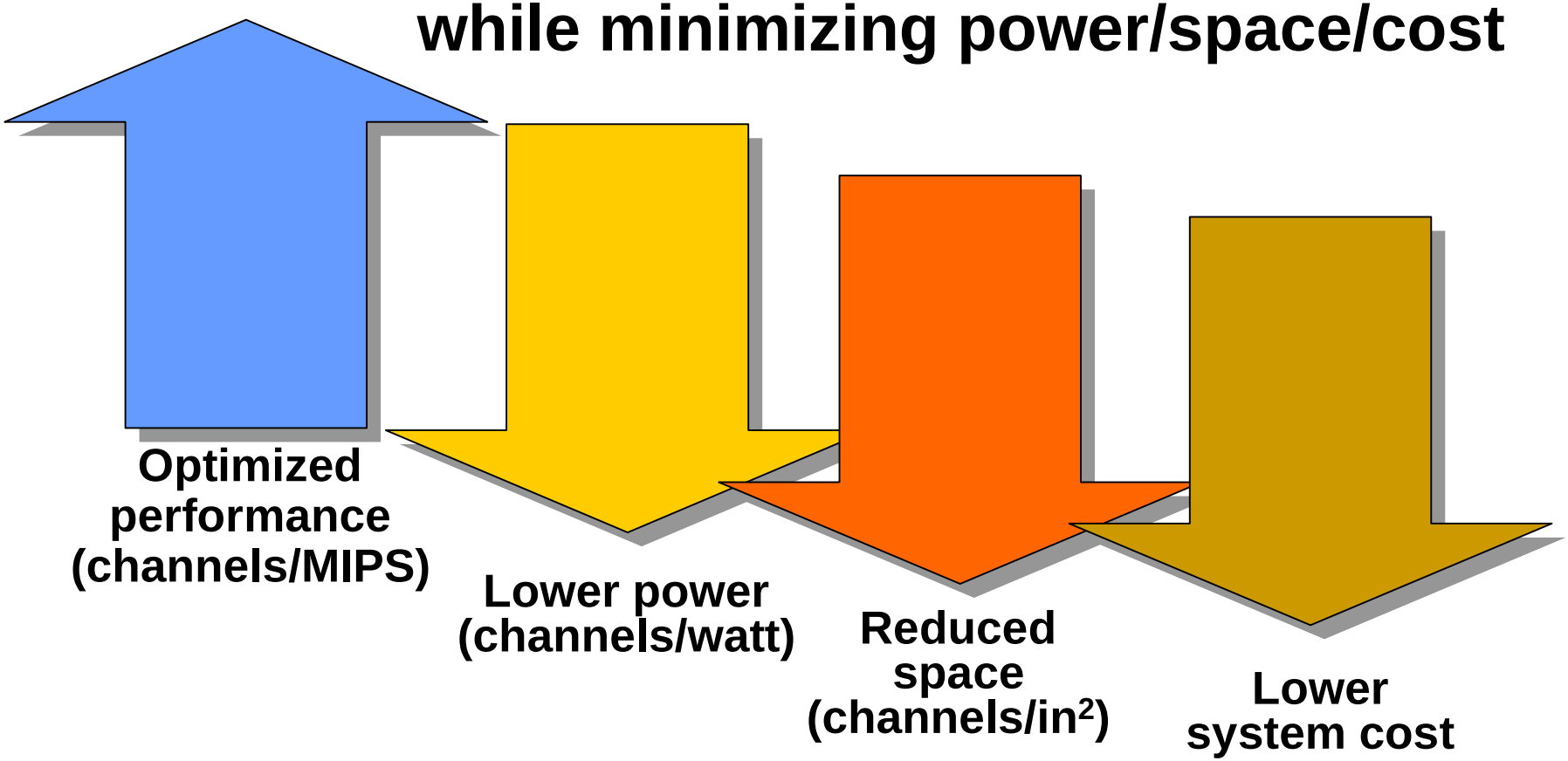
How do I work with TI's 'C5000?

How do I get my performance?

- What performance can I expect?
- How do I interface easily?
- What are the new 'C5000 devices?
- How do I minimize power consumption?
- How does TI enable power-efficient performance at lower cost?

'C5000: Power-efficient performance

Efficient MIPS *and* on-chip memory/peripherals... while minimizing power/space/cost



Architecture optimized for DSP

#1: CPU designed as a DSP engine

- An execution environment that handles 32-bit constructs in a 16-bit architecture

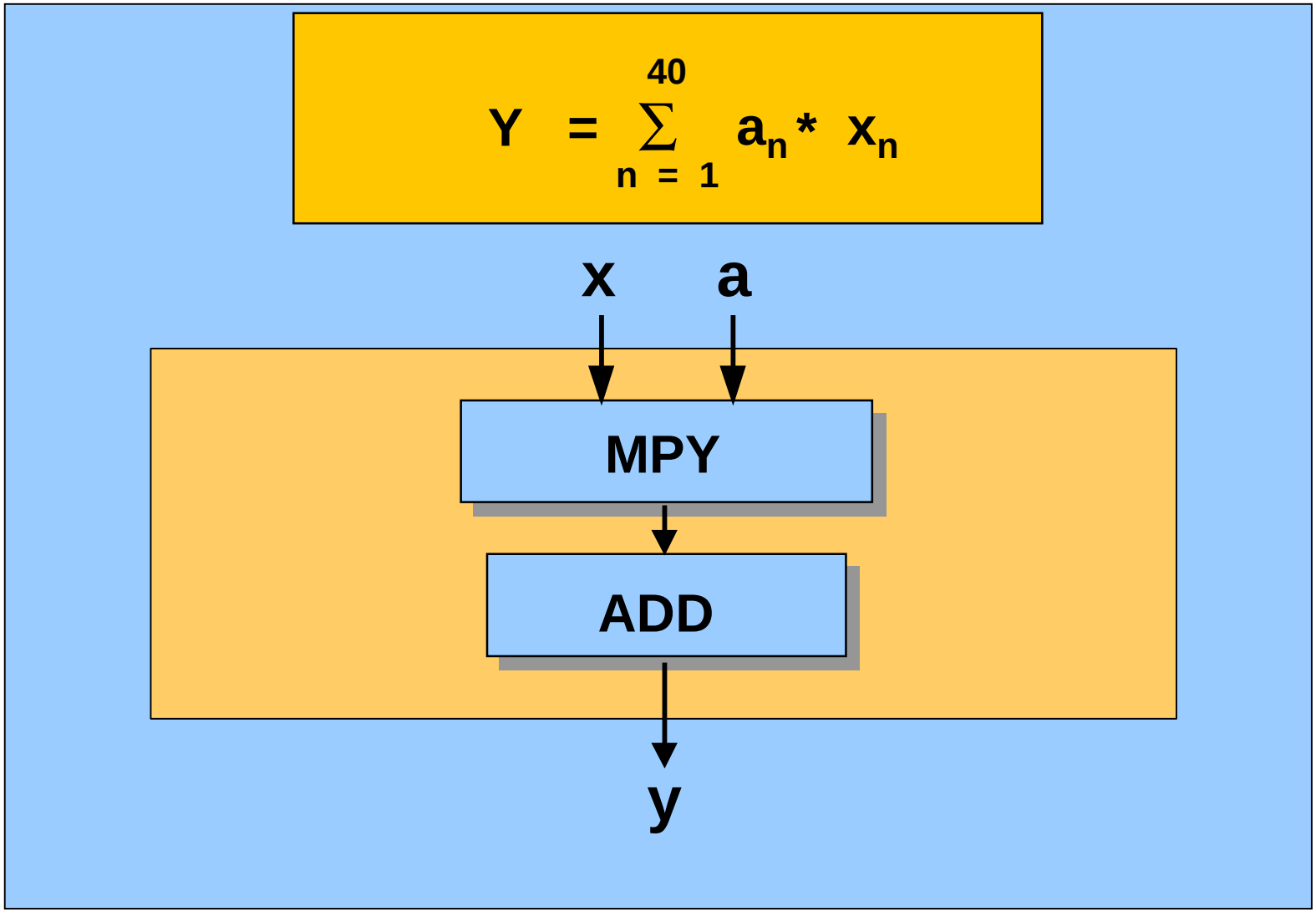
#2: Multiple busses for efficient data and program flow

- Four busses and large on-chip memory that result in sustained performance near peak

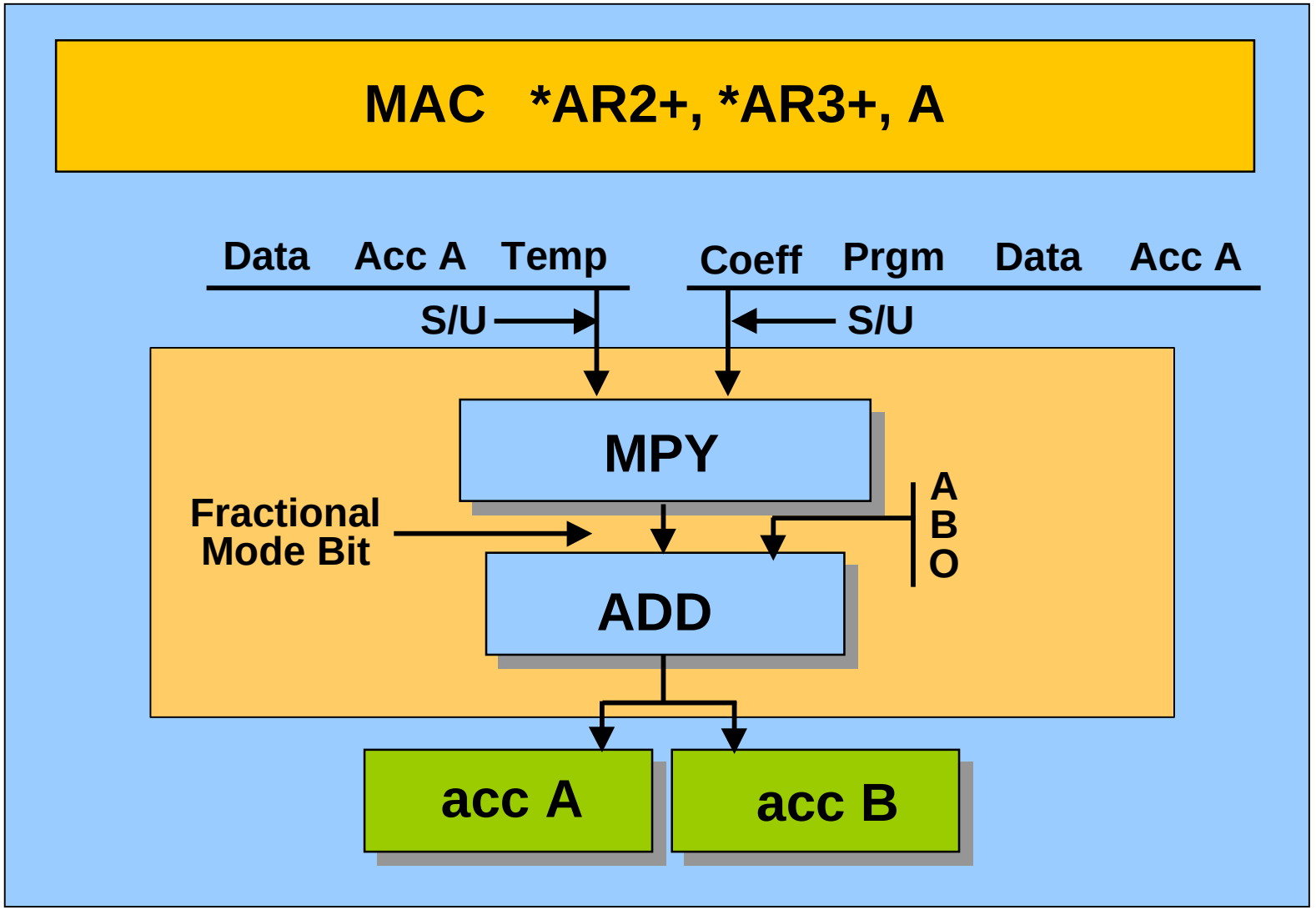
#3: Highly tuned instruction set for powerful DSP computing

- Sophisticated instructions that execute in fewer cycles, with less code and low power demands

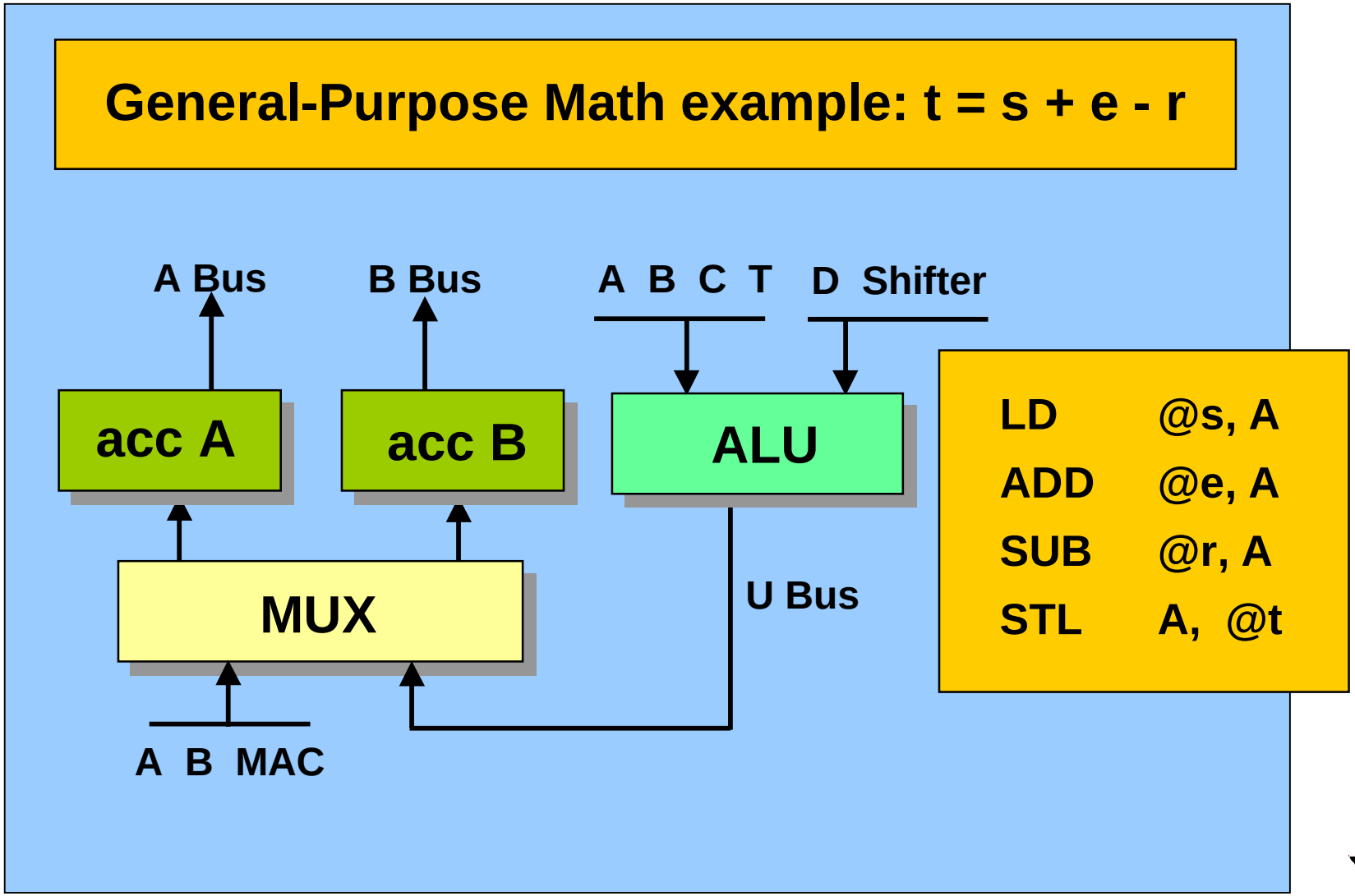
Key #1: DSP engine



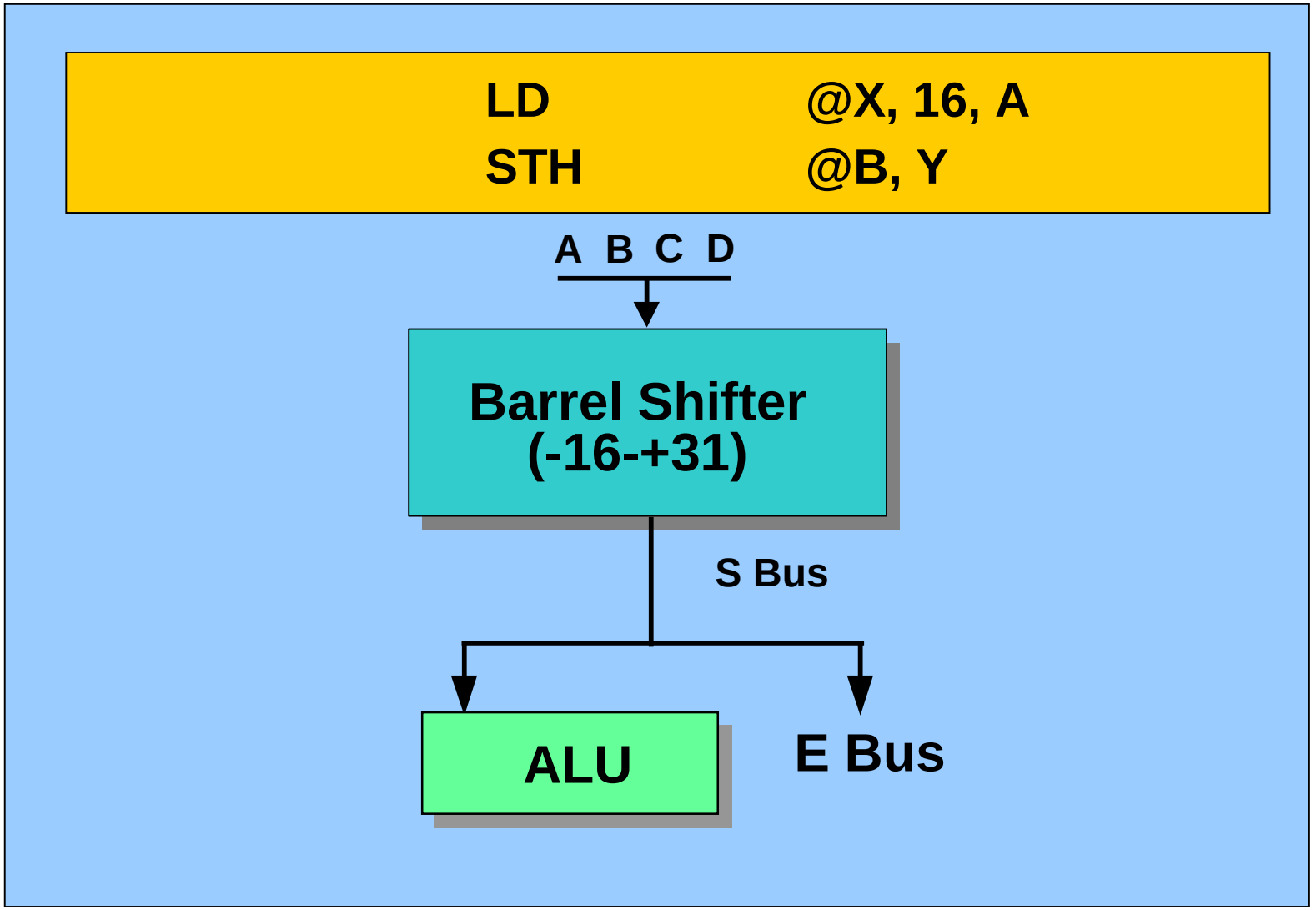
Key #1: MAC Unit



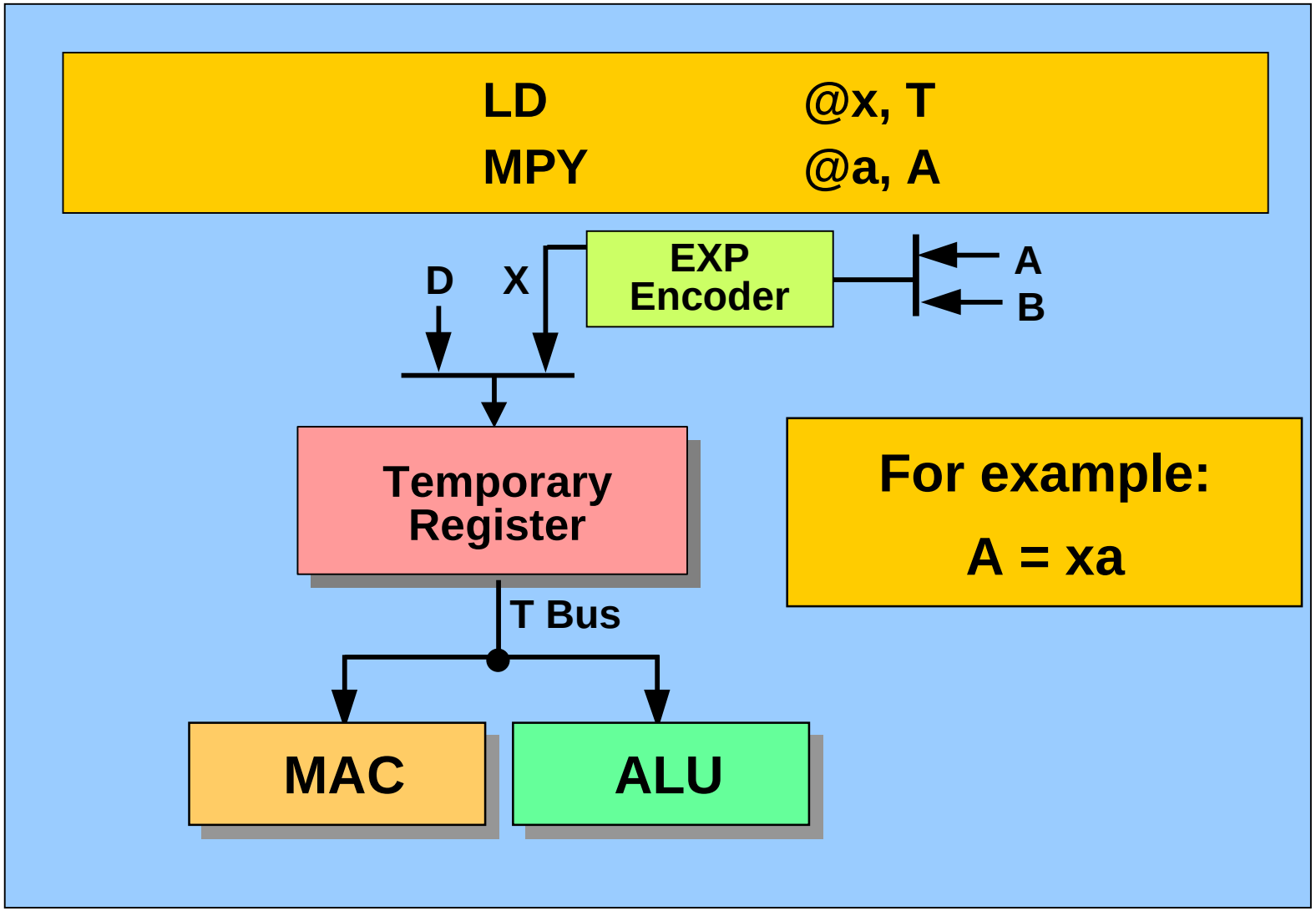
Key #1: Accumulators + Adder



Key #1: Barrel shifter



Key #1: Temporary register



Key #2: Efficient data/program flow

#1: CPU designed as a DSP engine

- An execution environment that handles 32-bit constructs in a 16-bit architecture

#2: Multiple busses for efficient data and program flow

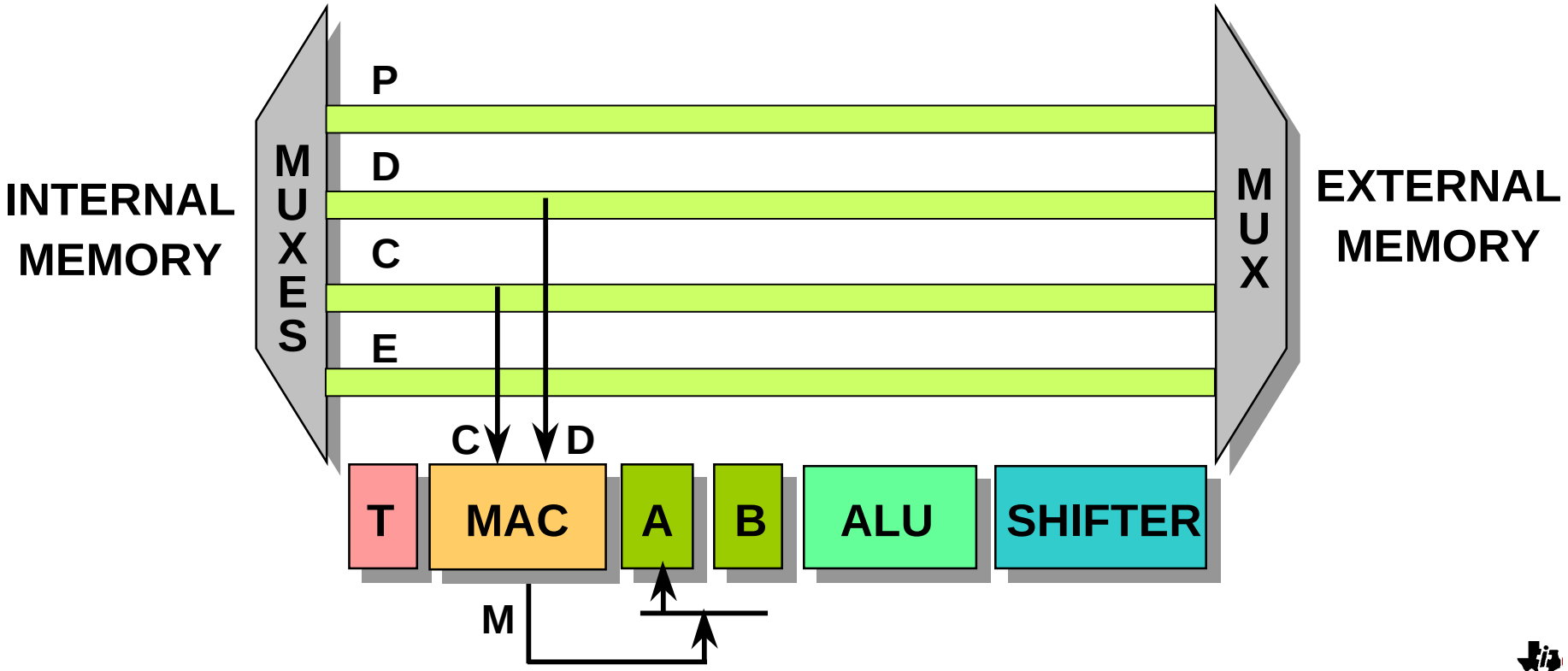
- Four busses and large on-chip memory that result in sustained performance near peak

#3: Highly tuned instruction set for powerful DSP computing

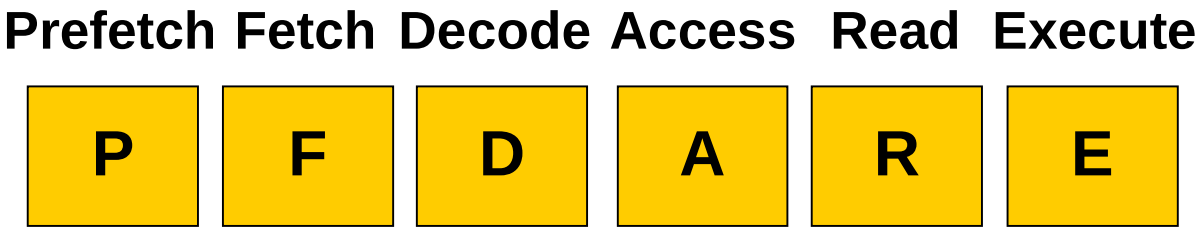
- Sophisticated instructions that execute in fewer cycles, with less code and low power demands

Key #2: Multiple busses

MAC *AR2+, *AR3+, A

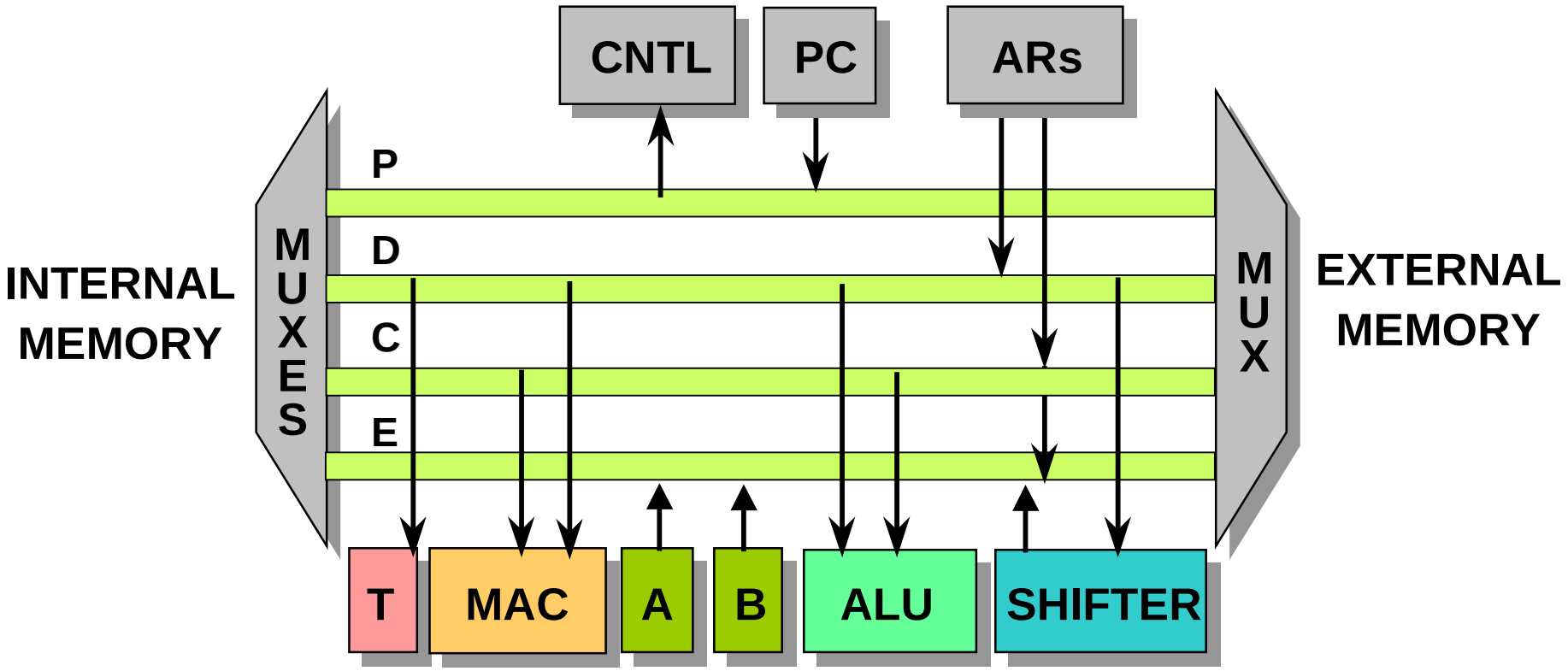


Key #2: Pipeline

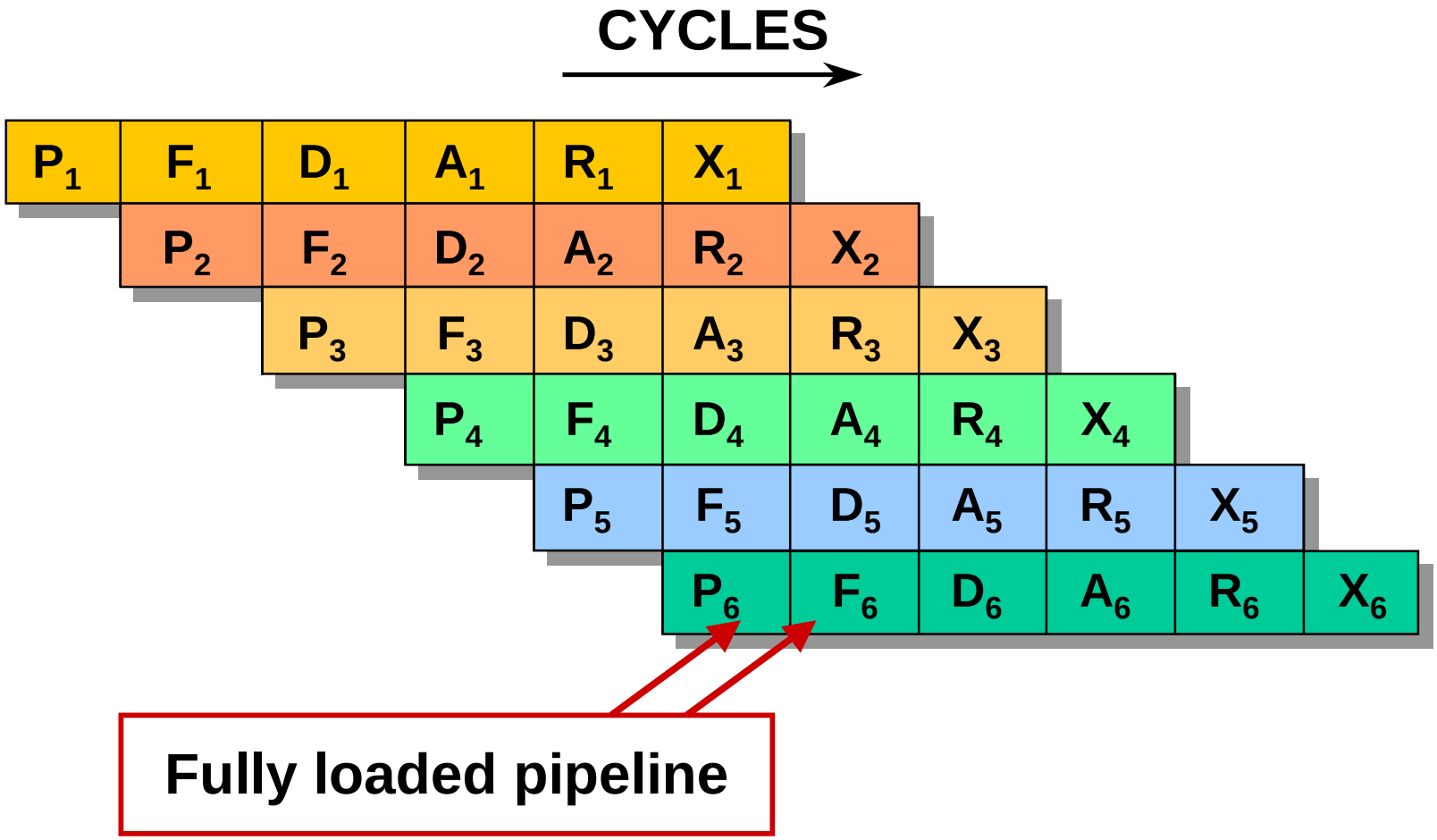


- Prefetch: Calculate address of instruction
- Fetch: Collect instruction
- Decode: Interpret instruction
- Access: Collect address of operand
- Read: Collect operand
- Execute: Perform operation

Key #2: Bus usage



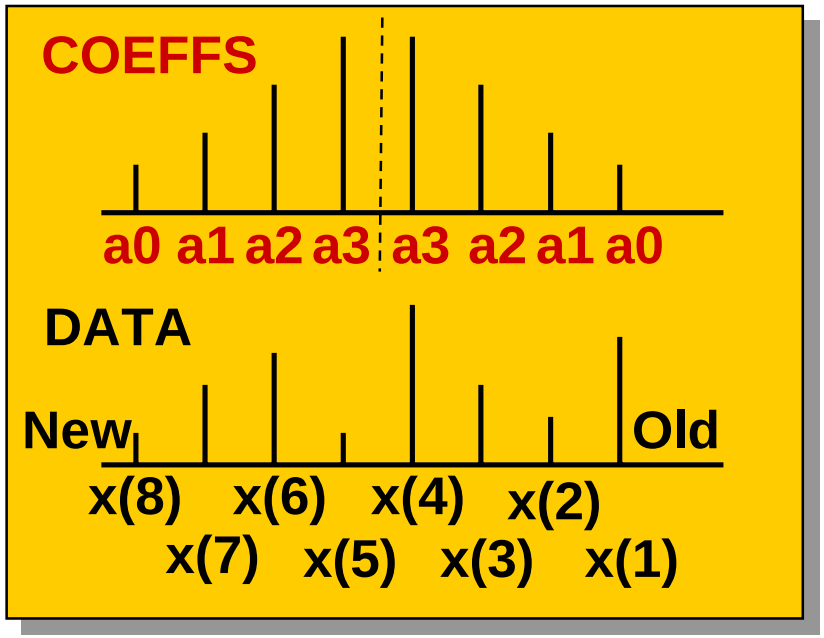
Key #2: Pipeline performance



Key #3: Powerful instructions

- **#1: CPU designed as a DSP engine**
  - An execution environment that handles 32-bit constructs in a 16-bit architecture
- **#2: Multiple busses for efficient data and program flow**
  - Four busses and large on-chip memory that result in sustained performance near peak
- **#3: Highly tuned instruction set for powerful DSP computing**
  - Sophisticated instructions that execute in fewer cycles, with less code and low power demands

Key #3: Symmetric FIR Filter



Symmetric FIR Filters are commonly used in applications where phase distortion may degrade the signal quality, e.g.: modems.

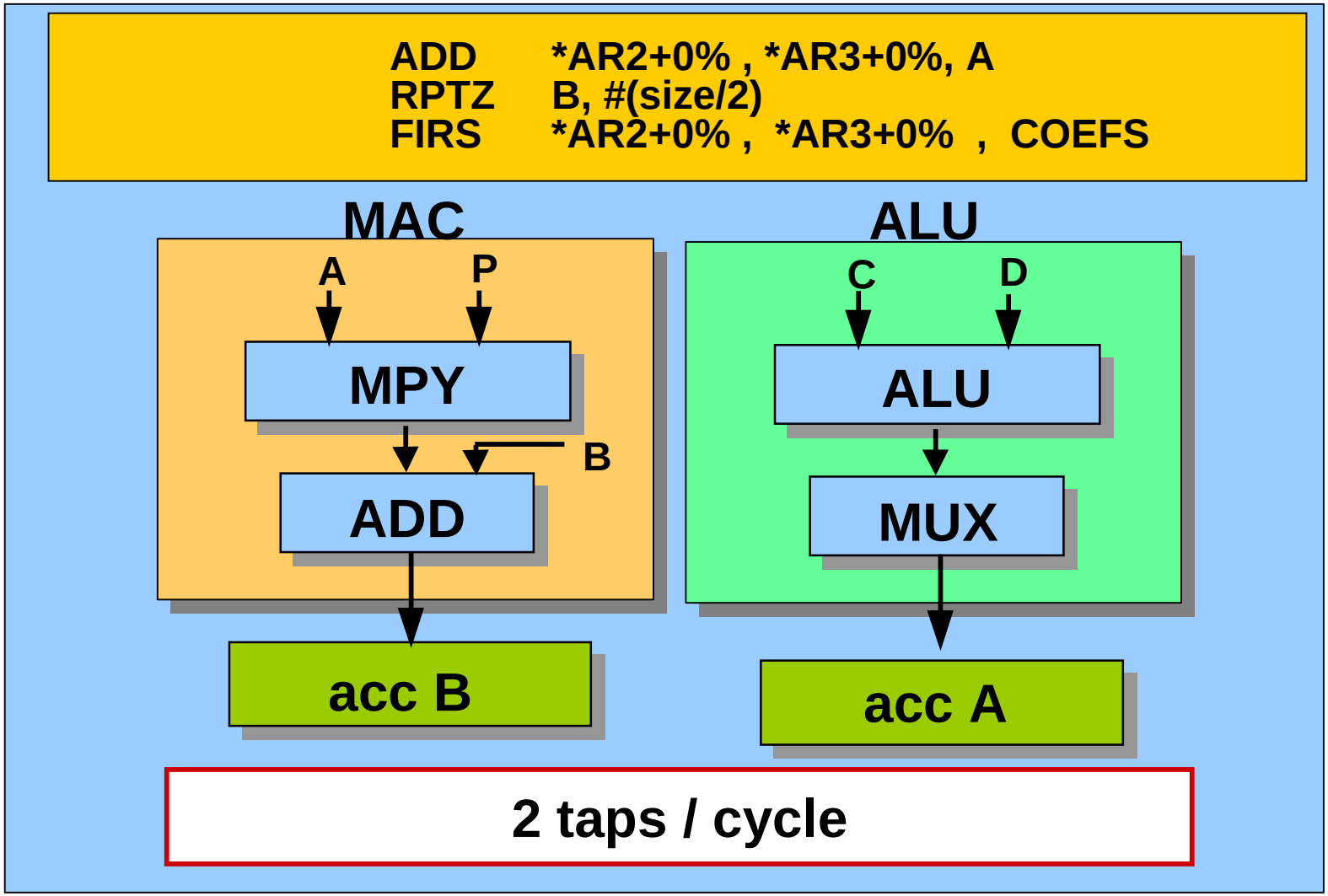
The general form of this FIR equation is written using 8 Mults, 7 Adds

$$Y(n) = a0x(8)+a1x(7)+a2x(6)+a3x(5)+a3x(4)+a2x(3)+a1x(2)+a0x(1)$$

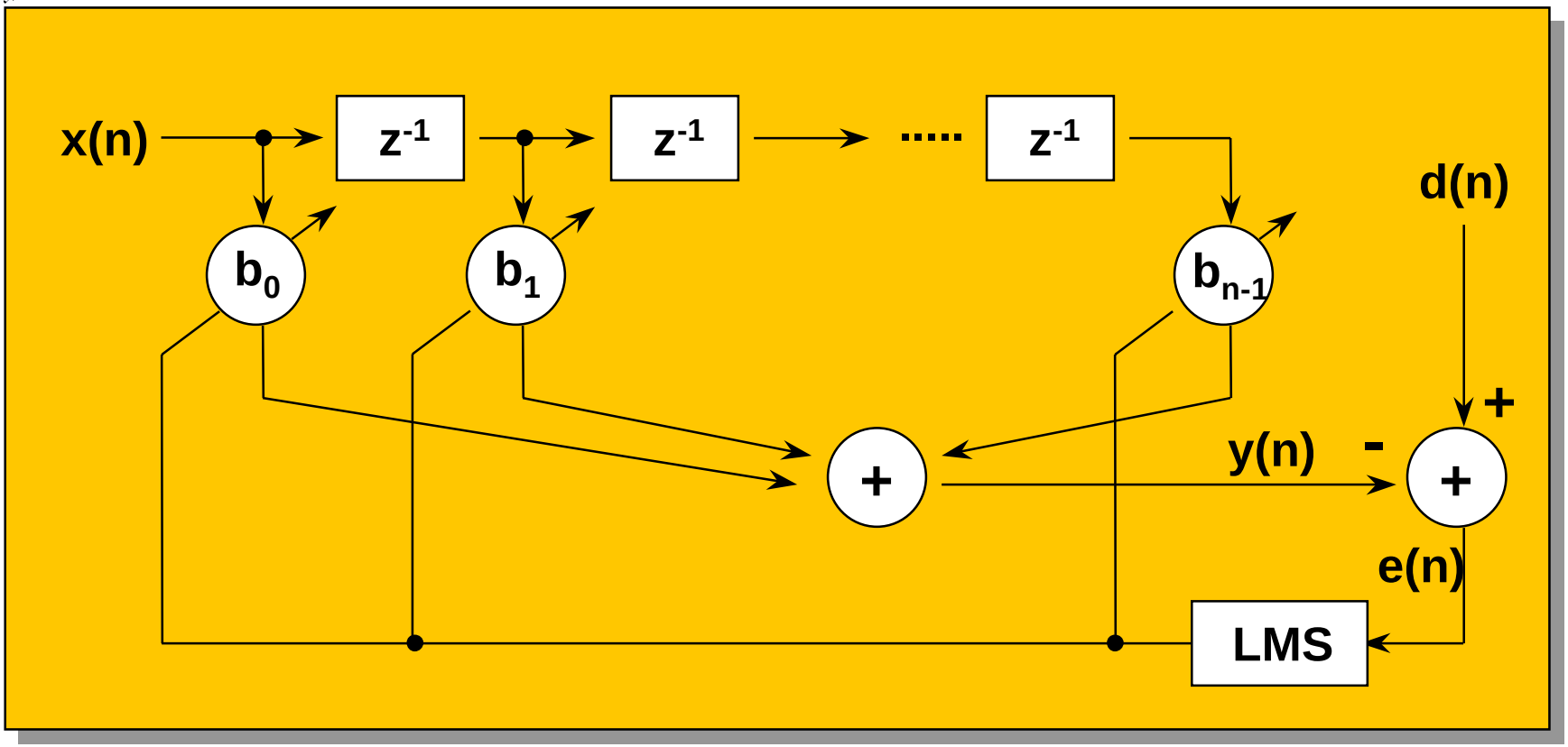
In the specific case of a Symmetric FIR we can use 4 Mults, 7 Adds

$$Y(n) = a0(x(8)+x(1))+a1(x(7)+x(2))+a2(x(6)+x(3))+a3(x(5)+x(4))$$

Key #3: FIRS


ADD *AR2+0% , *AR3+0%, A
RPTZ B, #(size/2)
FIRS *AR2+0% , *AR3+0% , COEFS
MAC
A
P
MPY
B
ADD
acc B
ALU
C
D
ALU
MUX
acc A
2 taps / cycle

Key #3: Adaptive FIR filter using LMS



FIR type filters are usually used in an adaptive algorithm since they are more tolerant of non-optimal coefficients.

Key #3: LMS loading

Each Iteration (only once)

1 - determine error :	e(i)	=	d(i) - y(i)
2 - scale by “rate” term B :	e´(i)	=	2*B*e(i)

Each Term (N sets)

3 - Qualify error with signal strength :	e´´(i)	=	x(i-k) * e´(i)
4 - Sum error with coefficient :	b(i+1)	=	b(i) + e´´(i)
5 - Update coefficient :	b(i)	=	b(i+1)

Analysis :

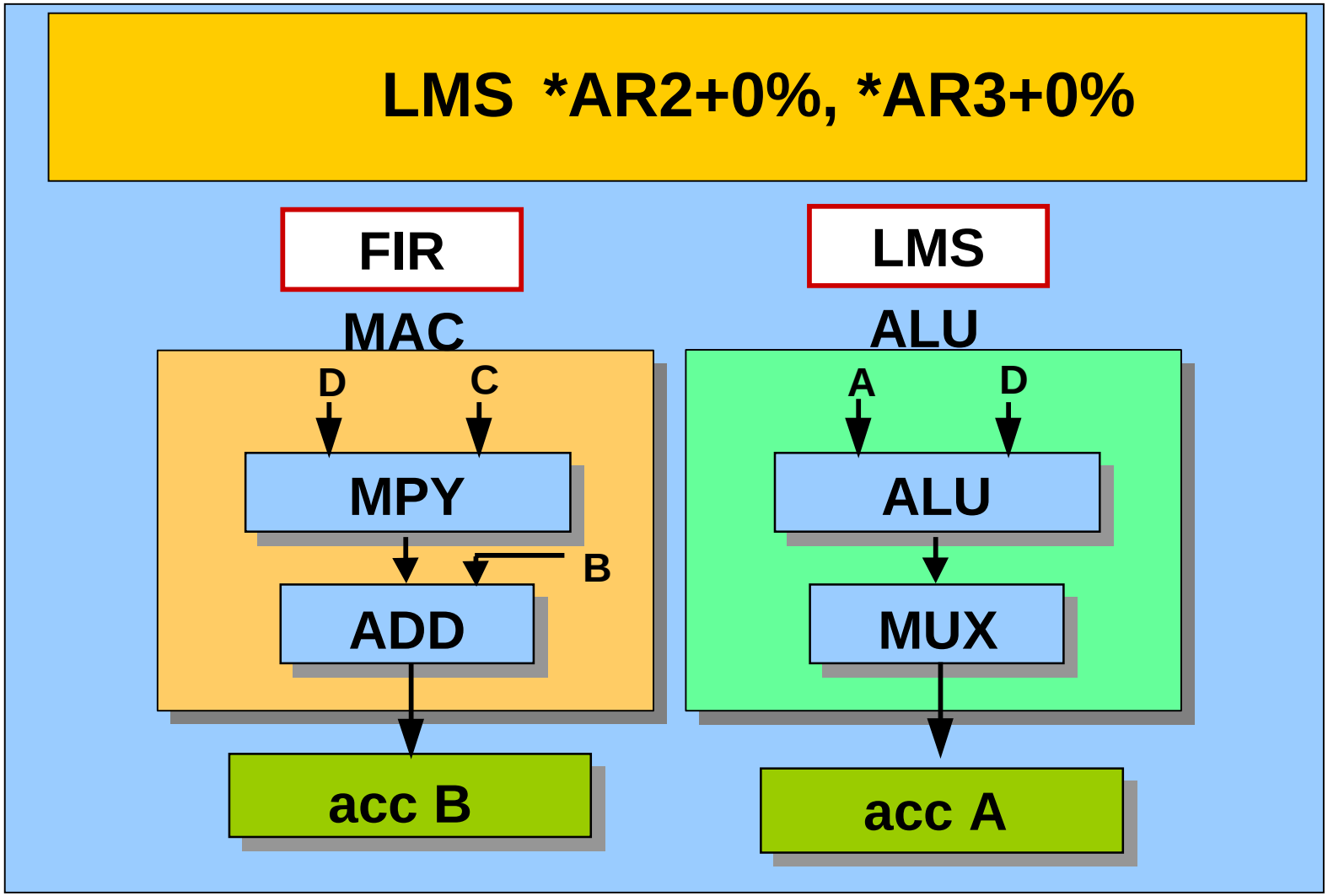
@100 tap: 200 cycles

LMS:	1	1	SUB
	2	1	MPY
	3	N	MPY
	4	N	ADD
	5	N	STH
FIR	a	N	MPY
	b	N	ADD
	c	1	STH

ST || MPY

LMS

Key #3: MAC + ALU enables LMS

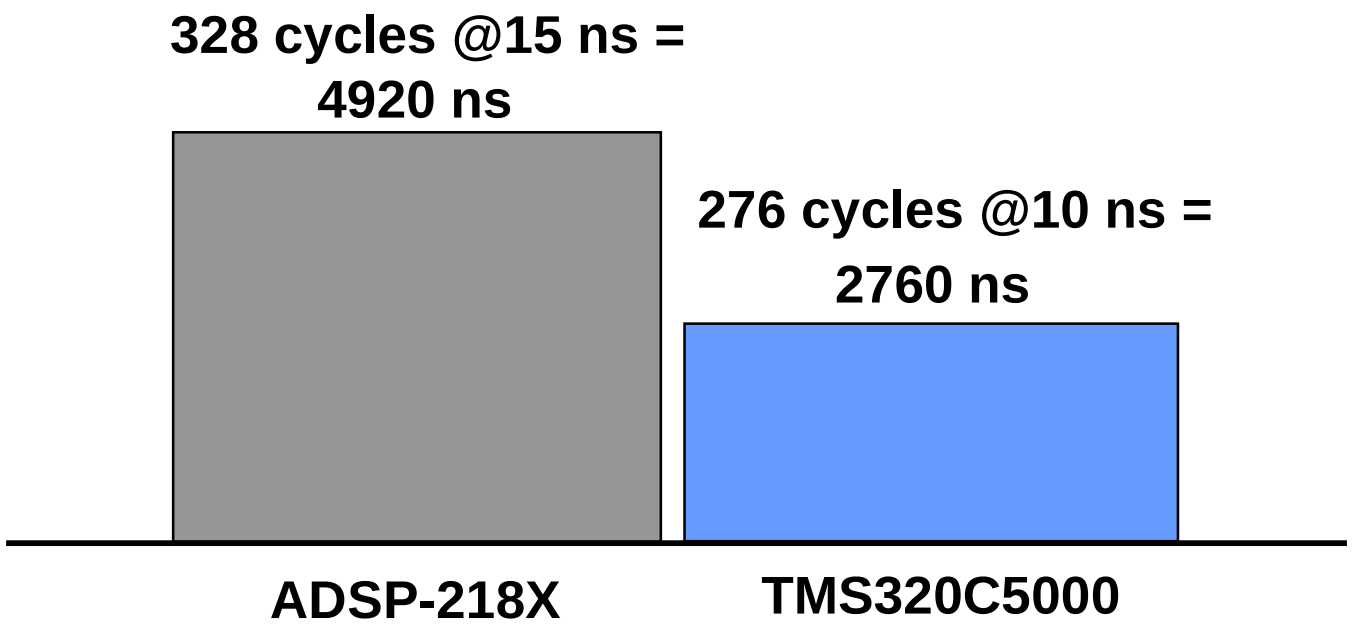


Key #3: Advanced applications

◆ **Symmetric FIR filter**	**FIRS**
◆ **Adaptive filtering**	**LMS**
Polynomial evaluation	**POLY**
Code book search	**STRCD**
	SACCD
	SRCCD
Viterbi	**DADST**
	DSADT
	CMPS

Key #3: FIR Filter in fewer cycles

'C5000 is squeezing the work done by a 24-bit processor into less time on a 16-bit processor



Full source code is listed in your Seminar Guide

'C5000: Performance summary

- ✔ **The true measure of power-efficient performance is your power consumption: the total mW required for *your* algorithm.**
- ✔ **'C5000 delivers more than the competition when designing within power, space or cost constraints.**
- ✔ **'C5000's highly efficient architecture enables robust performance on assembly source-code compatible devices -- from portable through infrastructure applications.**

How do I work with TI's 'C5000?

How do I get my performance?

What performance can I expect?

How do I interface easily?

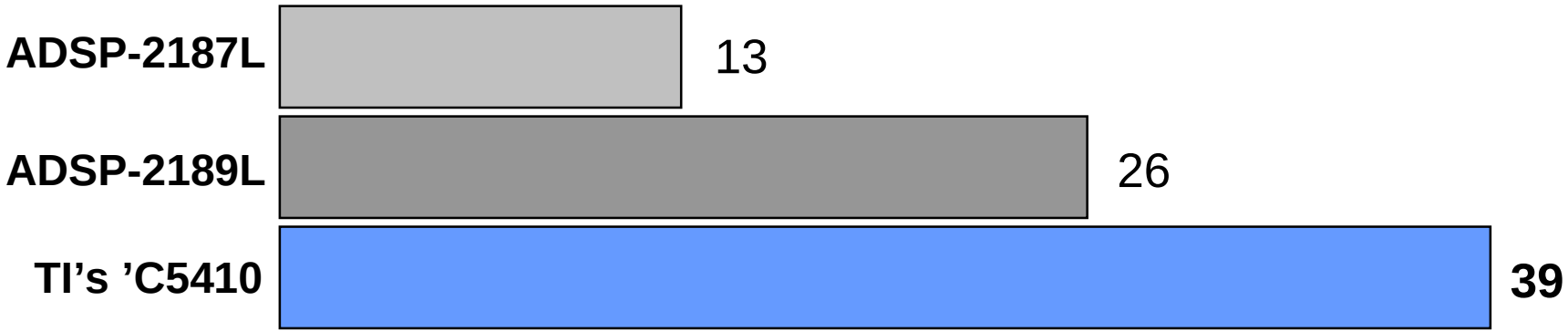
What are the new 'C5000 devices?

How do I minimize power consumption?

How does TI enable power-efficient performance at lower cost?

'C5000: More performance per Watt

Less power consumption = Up to 3x channels

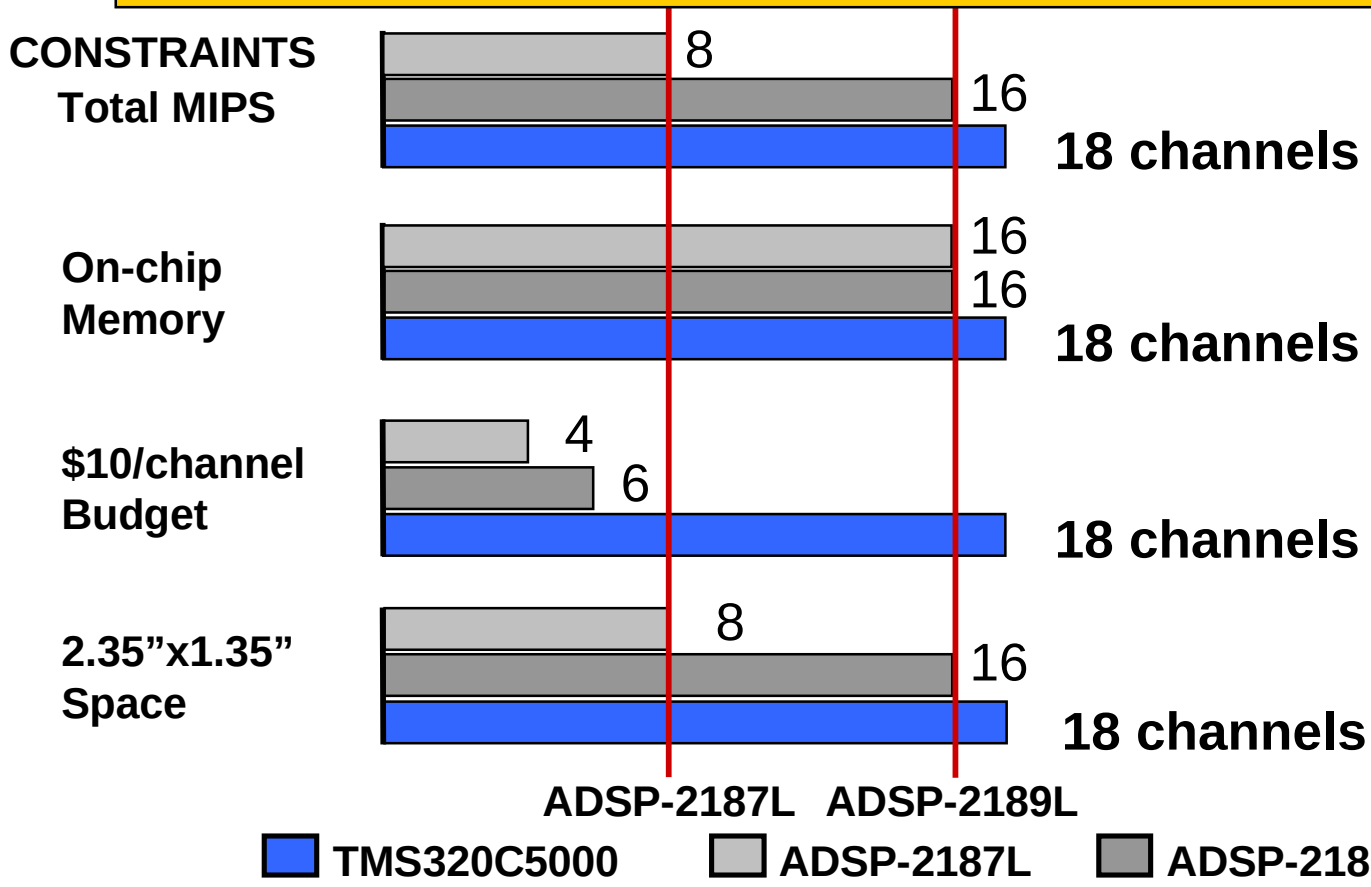


1 Channel = G.723.1 + Echo Cancellation + VOP processing
Power budget = 2 Watts

Source: Analog Devices news releases and web site

'C5000: Performance per constraint

Competitors are limited by available MIPS
(1 Channel = G.723.1 + Echo Cancellation + VOP processing)

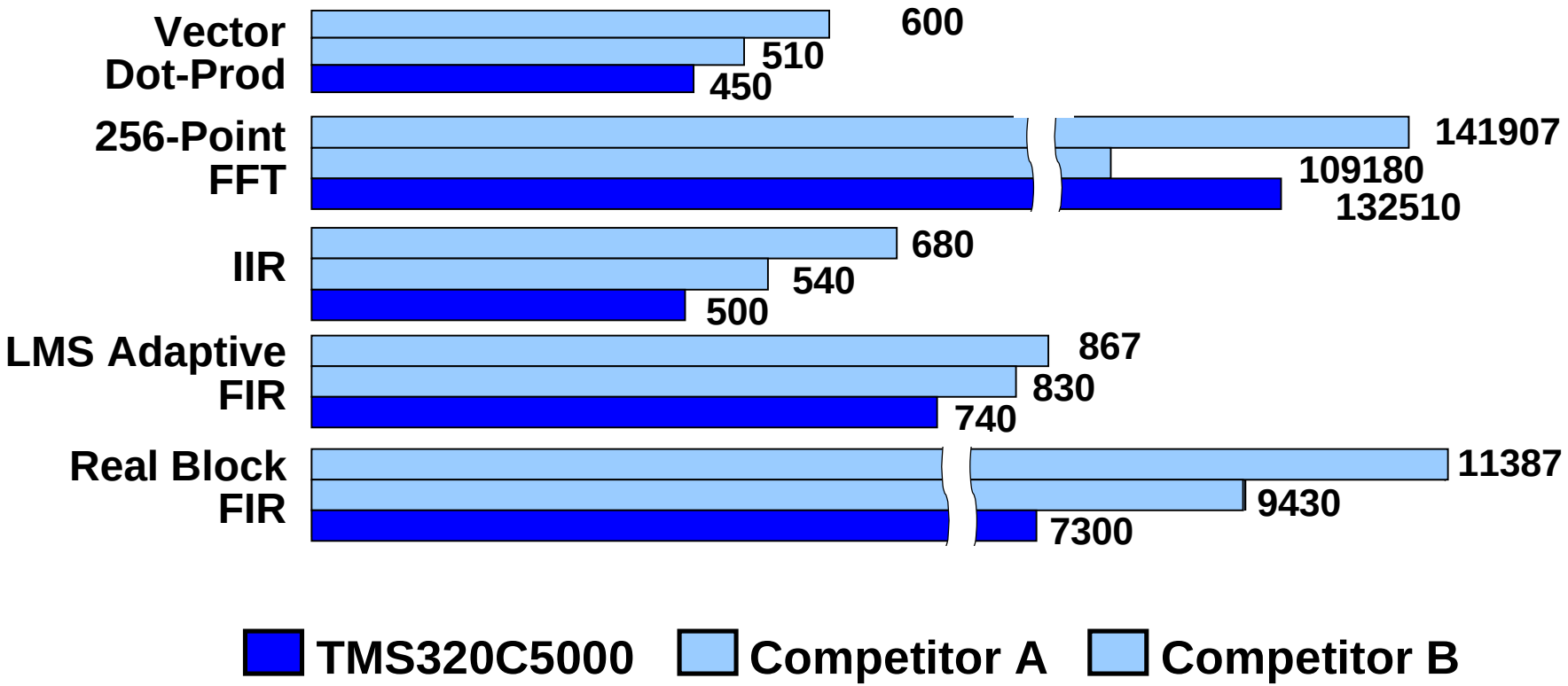


Source:
Analog Devices
news releases
and web site

'C5000: DSP functions

Execution Time vs. Major Competitors (in ns.)

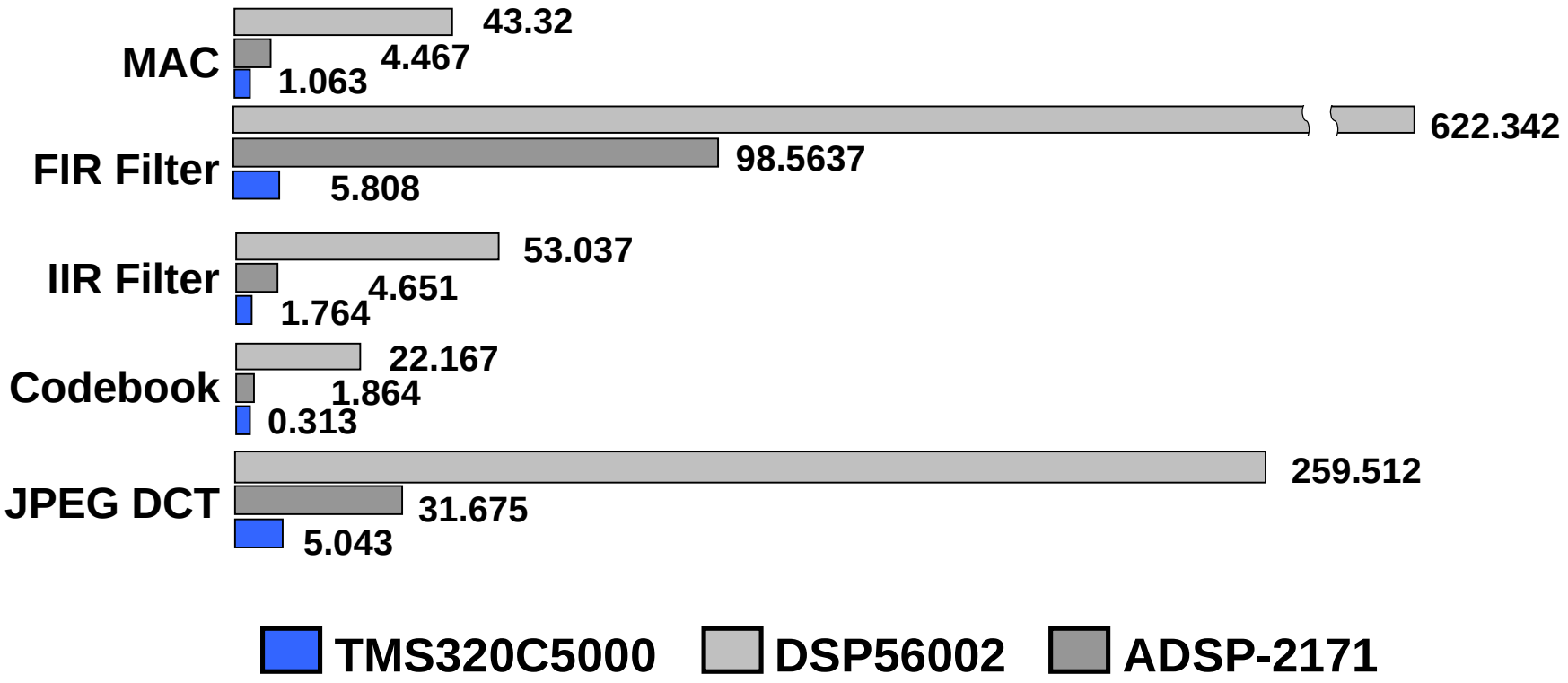
Source: *Buyer's Guide to DSP Processors*, ©1999 BDTI



'C5000 Compiler: Lowest cycle count

Compiler Performance vs. Competition

Cycles per task measured over 1000 cycles

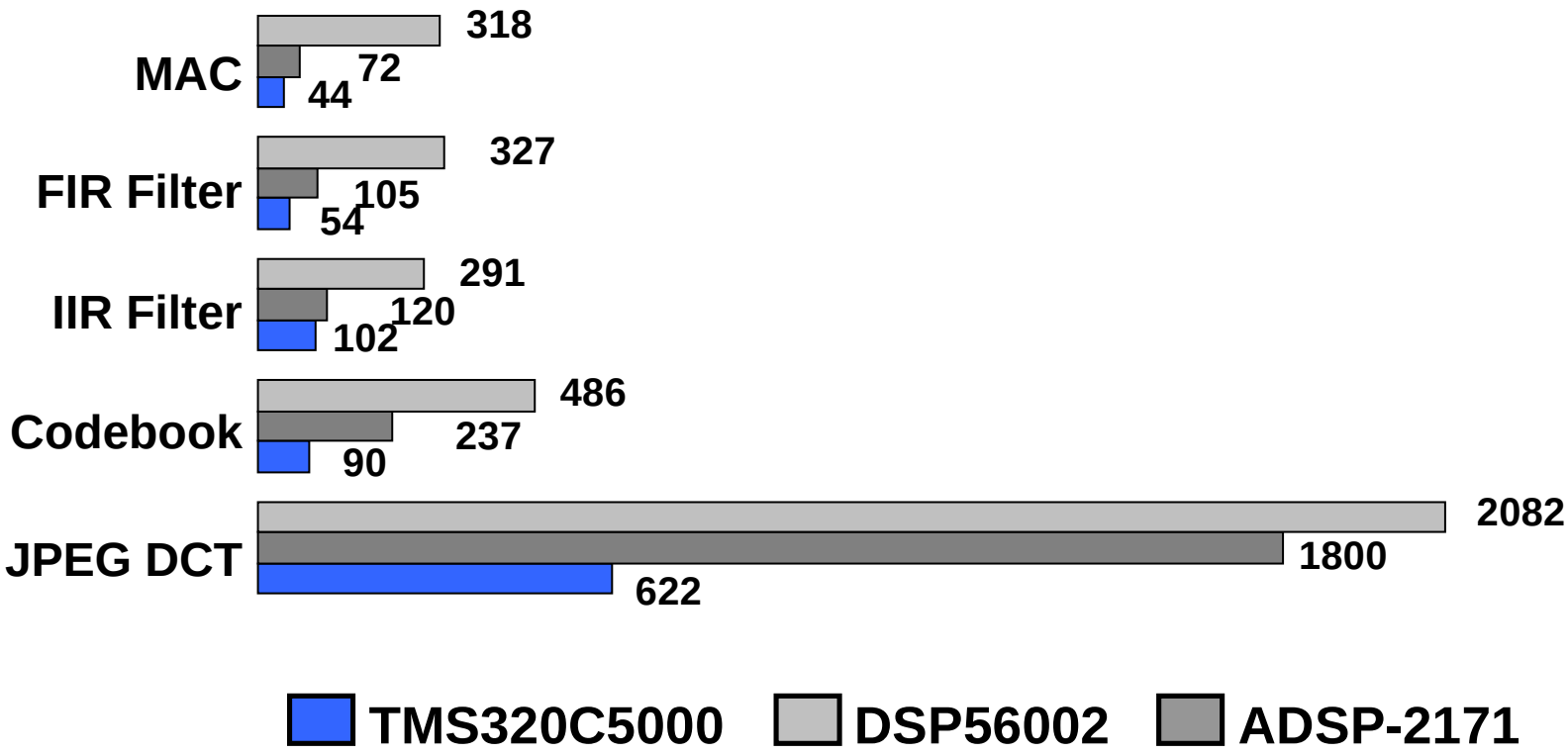


Source: TI internal testing on source code referenced in Seminar Guide.

'C5000 Compiler: Smallest code size

Compiler Performance vs. Competition

Measured in bytes



Source: TI internal testing on source code referenced in Seminar Guide.

How do I work with TI's 'C5000?

How do I get my performance?

What performance can I expect?

How do I interface easily?

What are the new 'C5000 devices?

How do I minimize my power consumption?

How does TI enable power-efficient performance at lower cost?

'C5000: Interfacing

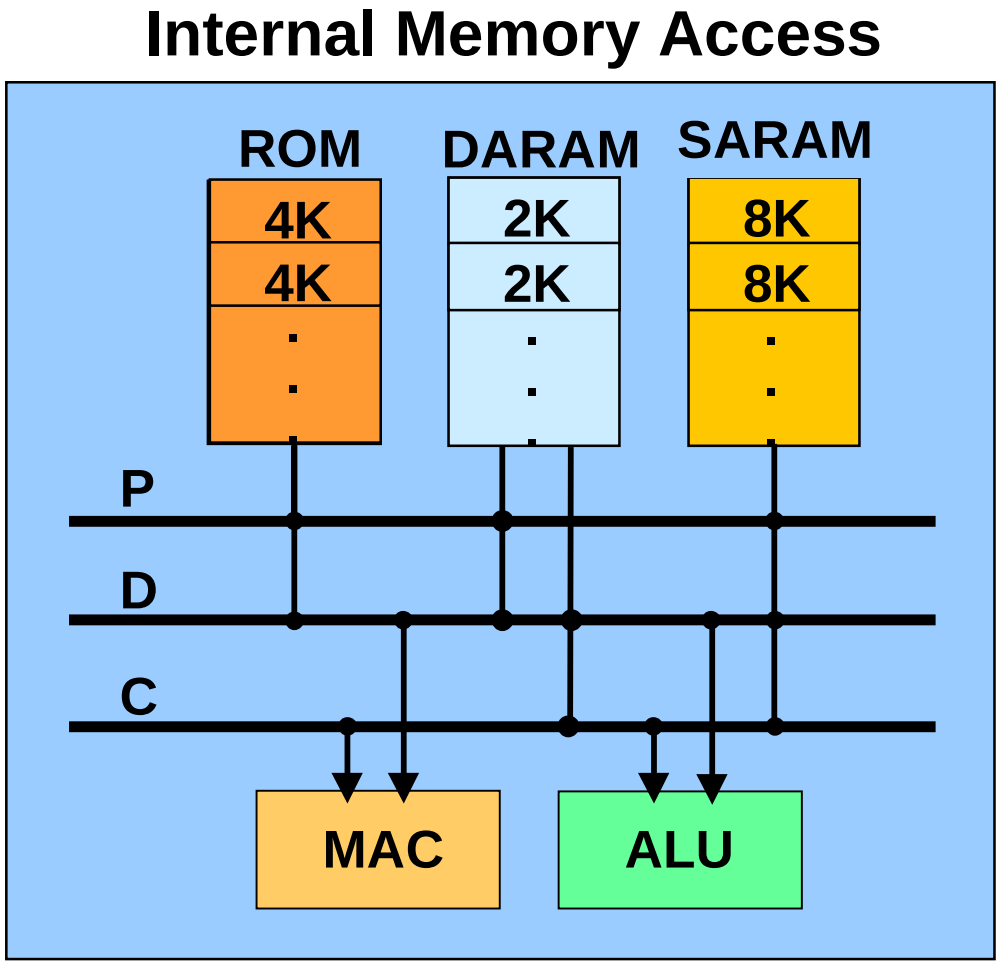
Variety of sources for easy interfacing

- ✔ **Internal Memory**
- ✔ **External Memory Interface (EMIF)**
- ✔ **Serial Ports:**
  - **Standard Serial Port**
  - **Buffered (BSP)**
  - **Multi-channel (McBSP)**
- ✔ **Host Port Interface (HPI)**
- ✔ **Direct Memory Access (DMA)**

Interfacing: Internal memory access

ROM & RAM:
One access per block per cycle

DARAM:
Two accesses per block per cycle

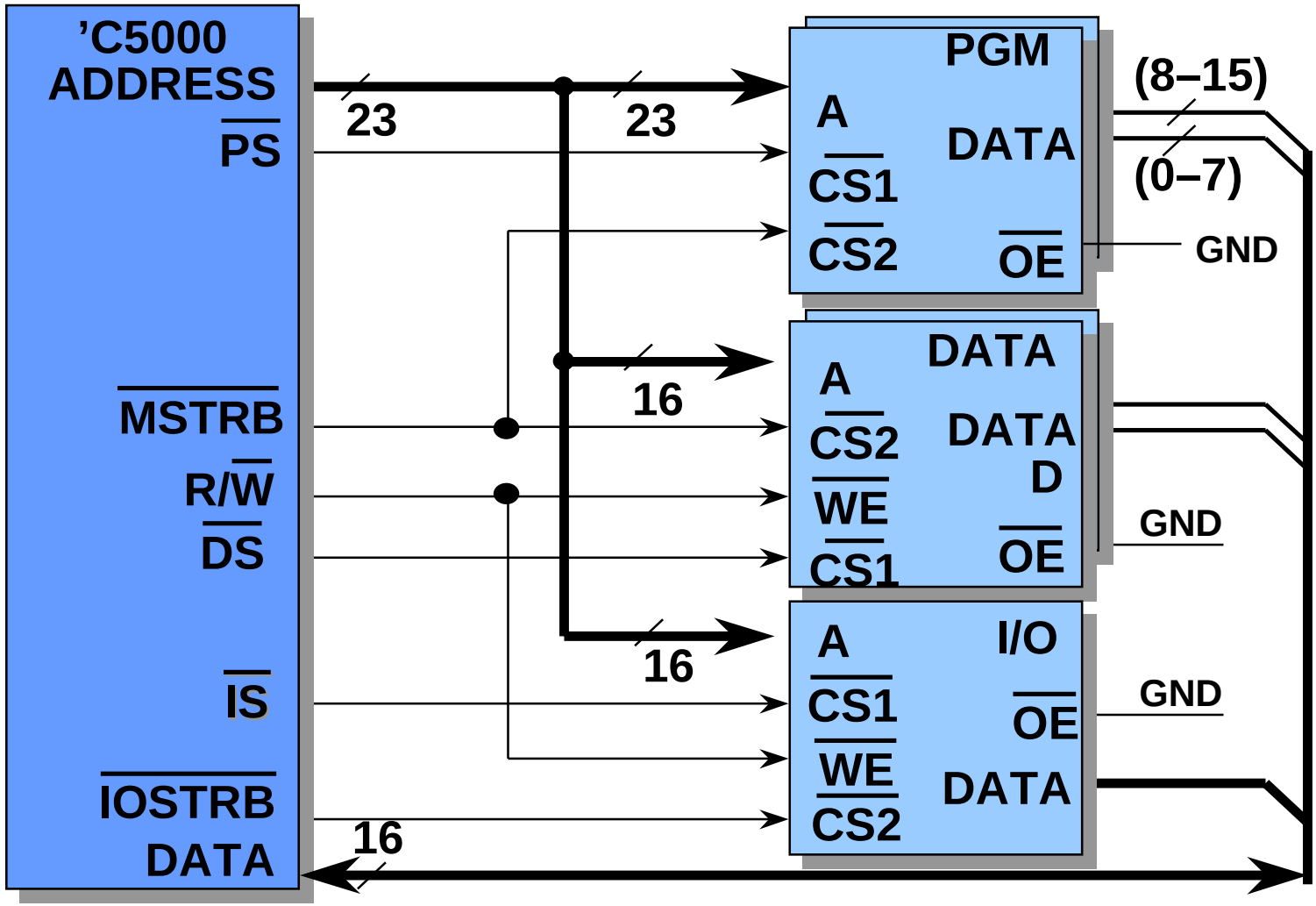


Interfacing: Memory

'C5000	DARAM	SARAM	ROM
'C5420	32 K	168 K	--
'C5402	16 K	--	4 K
'C5410	8 K	56 K	16 K
'C549	8 K	24 K	16 K
'C548	8 K	24 K	2 K
'C545/6	6 K	--	48 K
'C542/3	10K	--	2 K
'C541	5 K	--	28 K

16-bit words

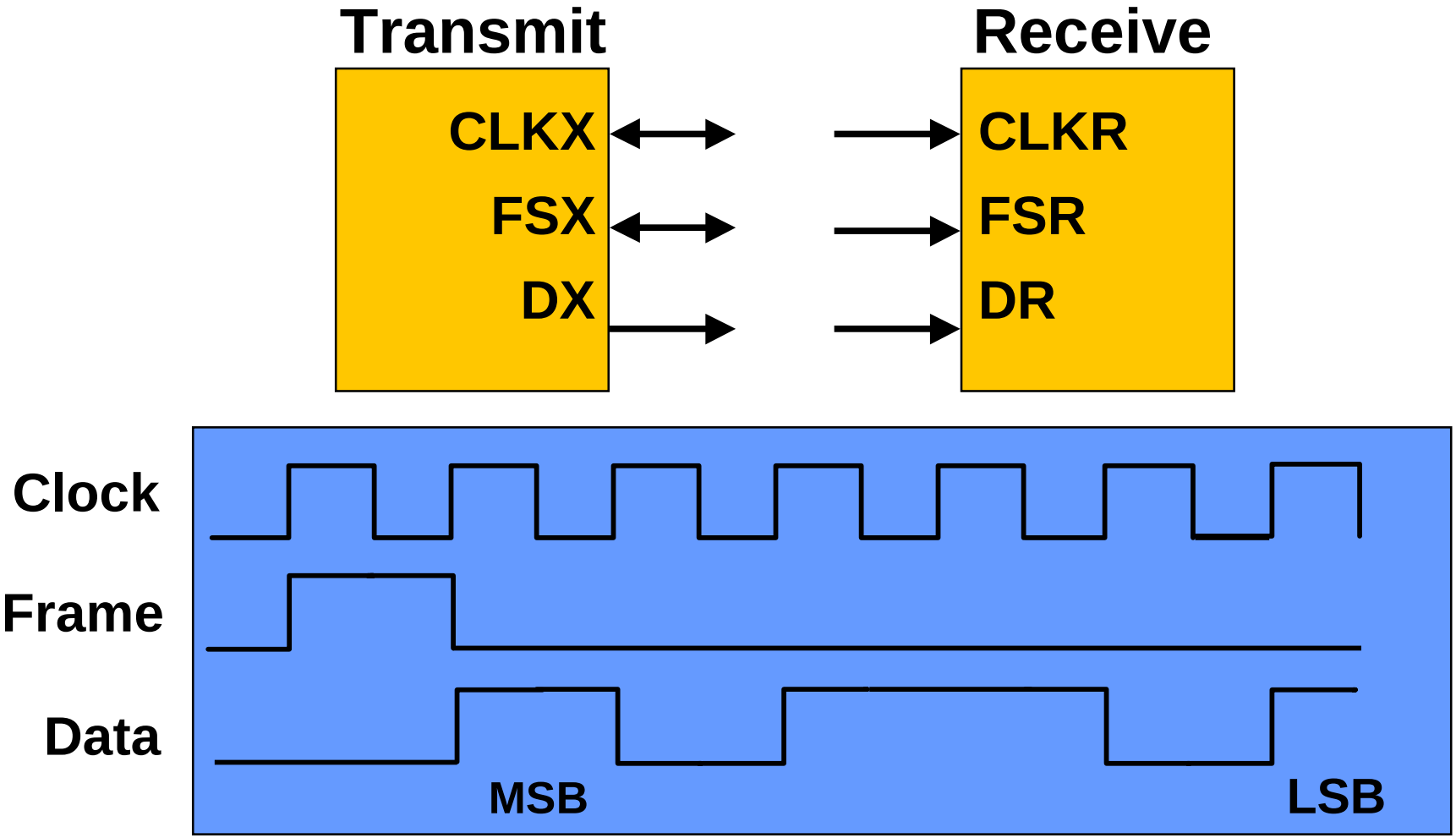
Interfacing: External Memory I/F



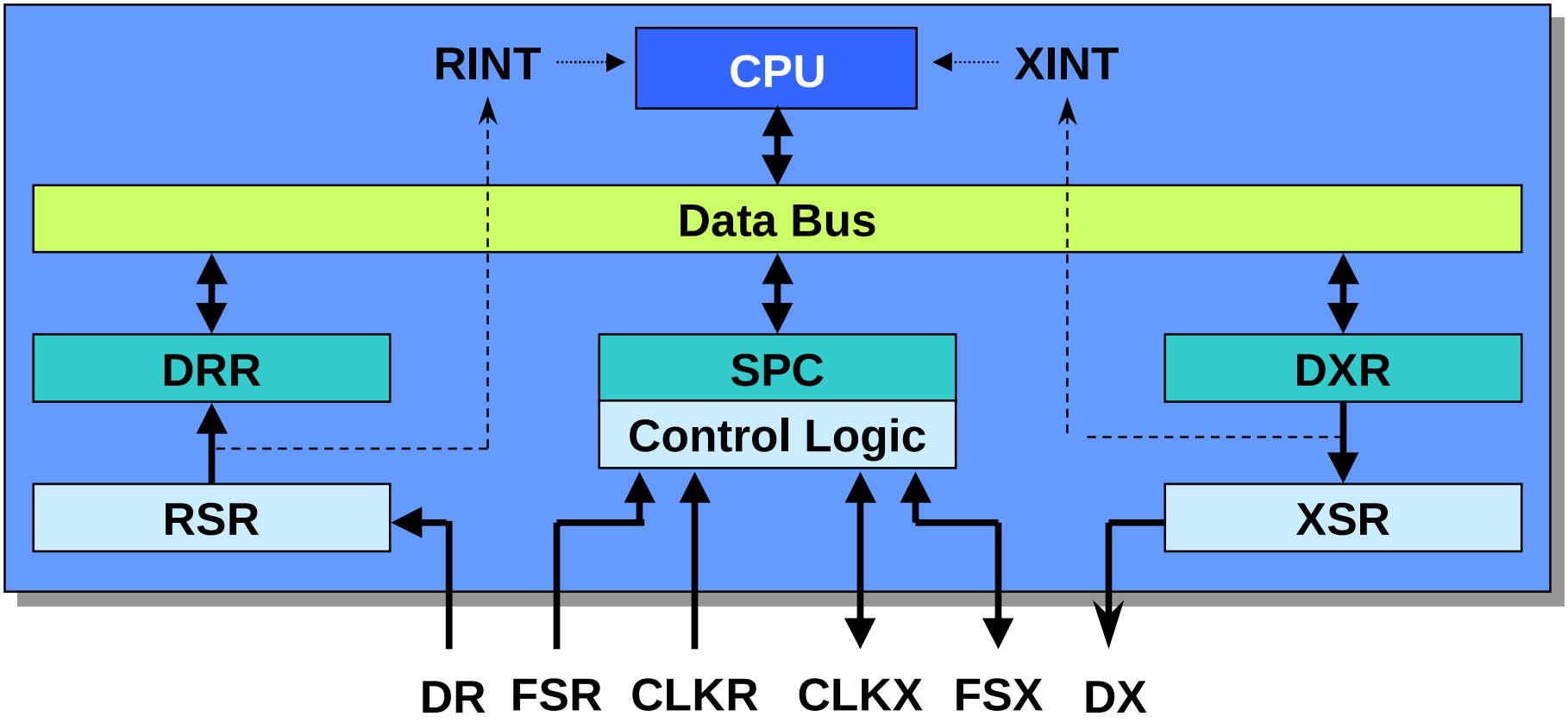
EMIF: Flexible memory I/F

- Typical asynchronous parallel interface
- Separate strobes for program, data and I/O spaces
- 23-bit address range for external program space
- 0-14 software programmable wait states
- Simplified bank switching -- programmable ability to insert wait states when crossing bank boundaries

Standard Serial Port: Pins & signals

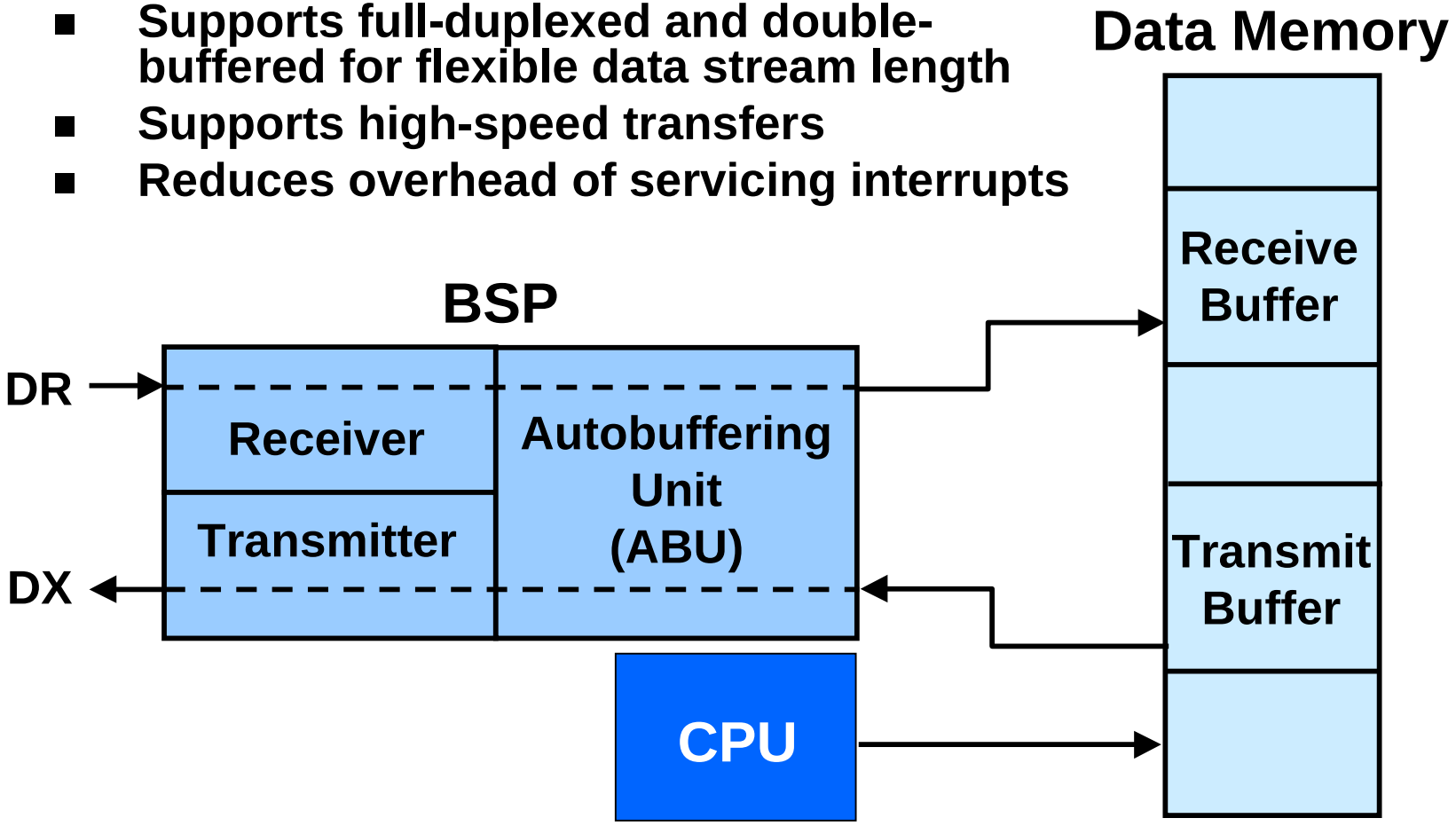


Standard Serial Port: Architecture

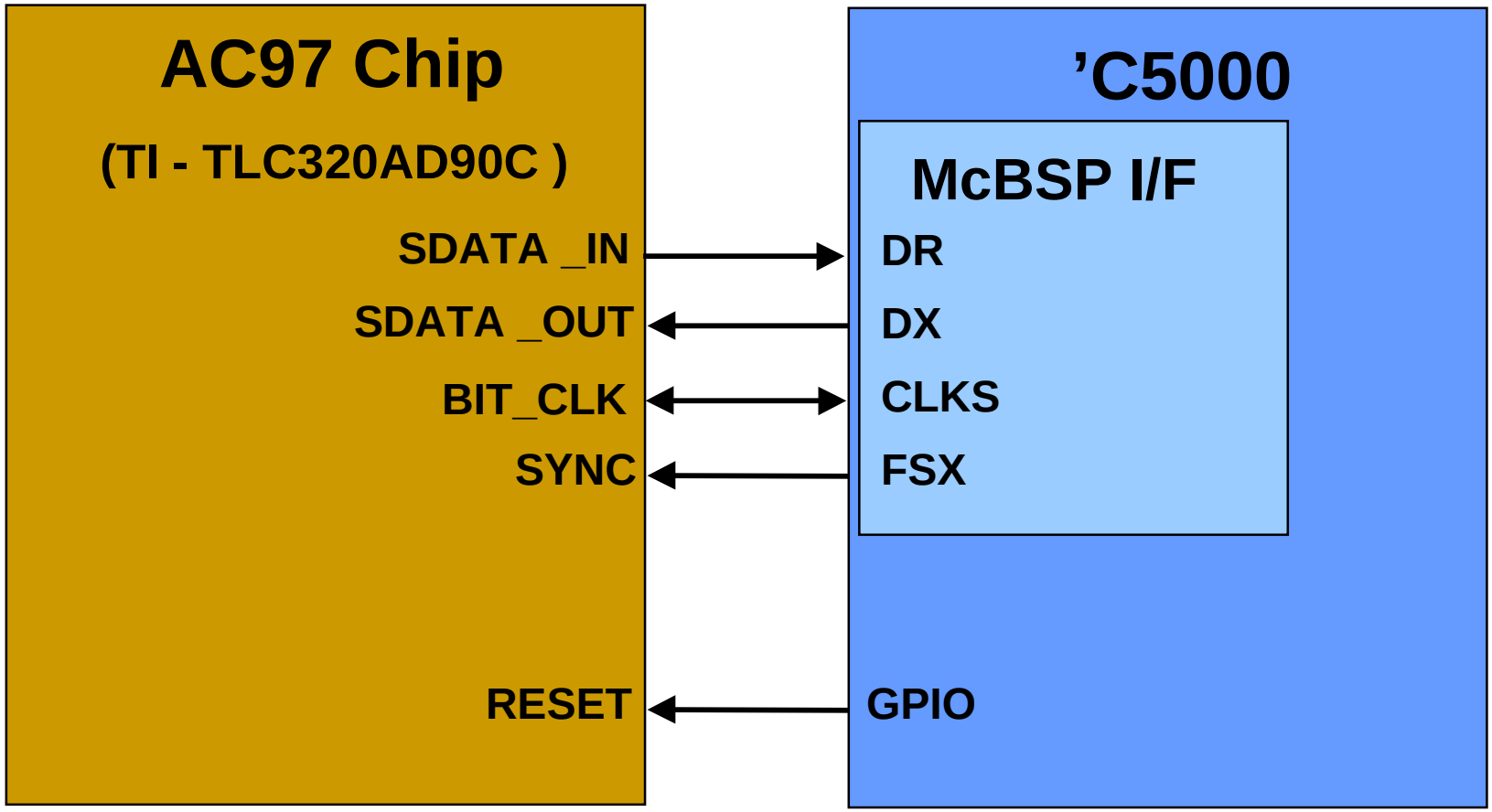


Buffered Serial Port: Autobuffering

- Clocks at full CLKOUT rate
- Supports full-duplexed and double-buffered for flexible data stream length
- Supports high-speed transfers
- Reduces overhead of servicing interrupts



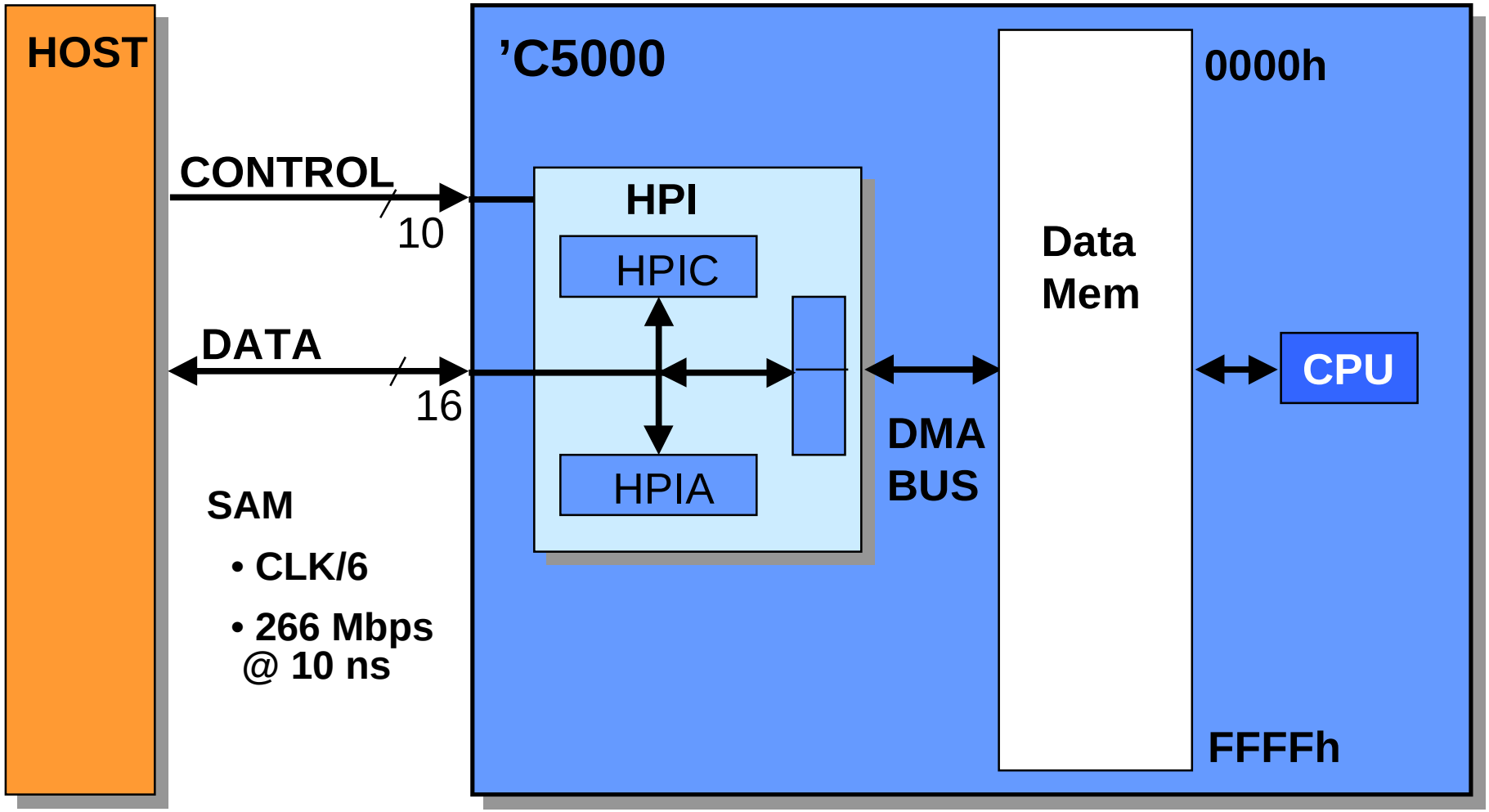
McBSP: Glueless I/F with AC97



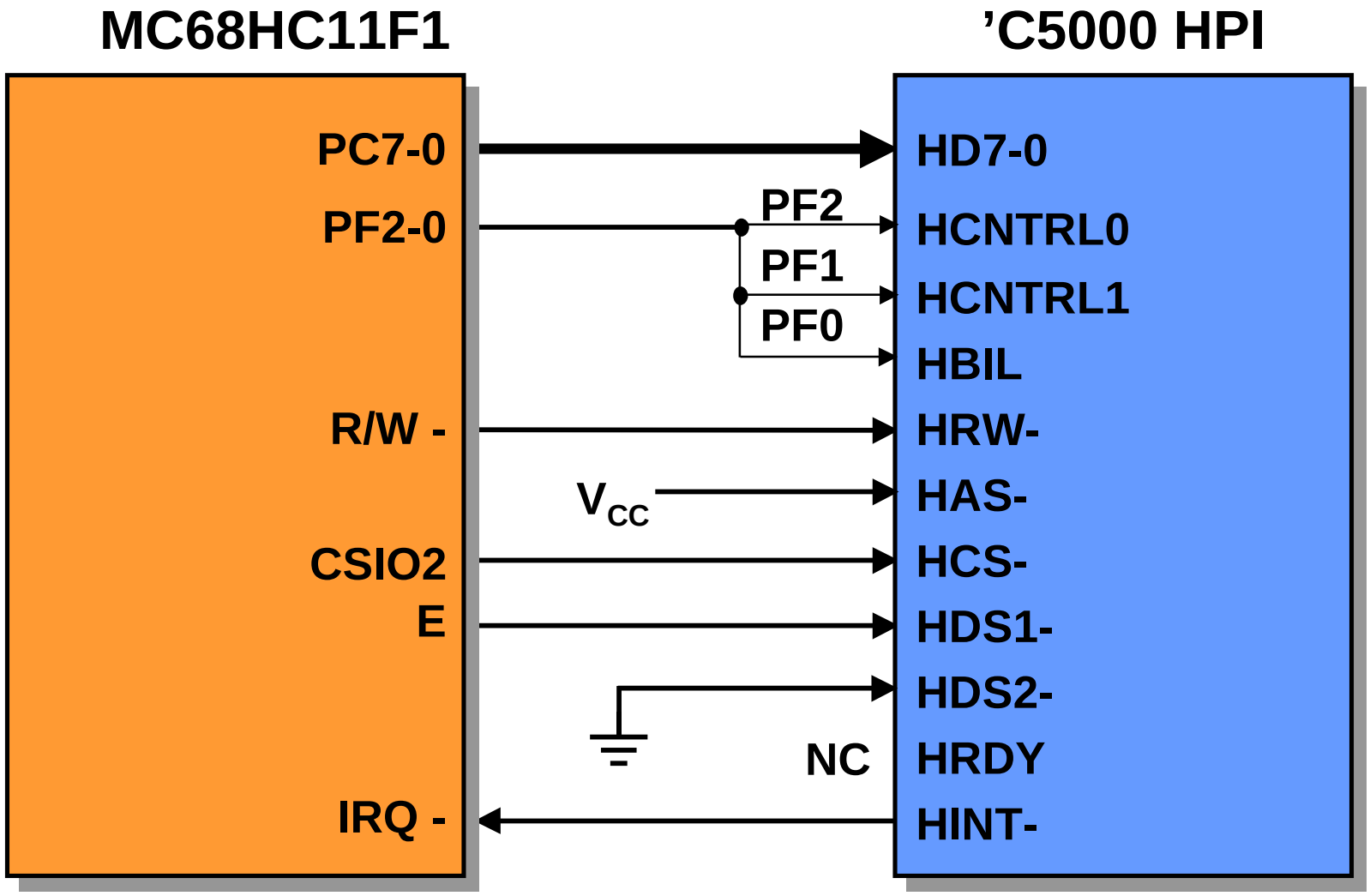
Interfacing: Serial ports

'C5000	McBSP	BSP	TDM	Standard
'C5420	6	--	--	--
'C5402	2	--	--	--
'C5410	3	--	--	--
'C548/9	--	2	1	--
'C545/6	--	1	--	1
'C542/3	--	1	1	--
'C541	--	--	--	2

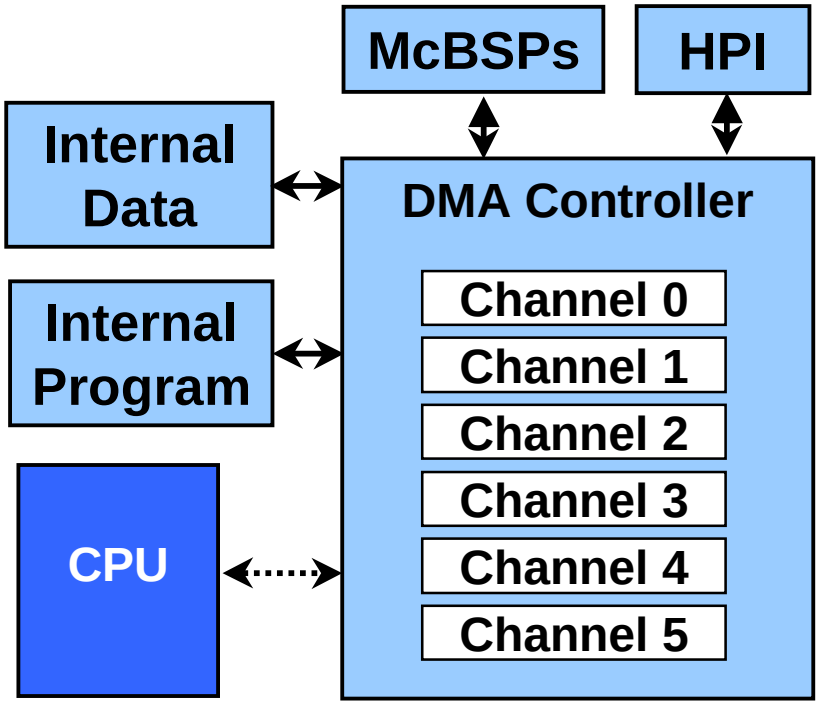
Interfacing: The HPI concept



Motorola 68HC11F1 to ’C5000 HPI



DMA: Transfers transparent to CPU



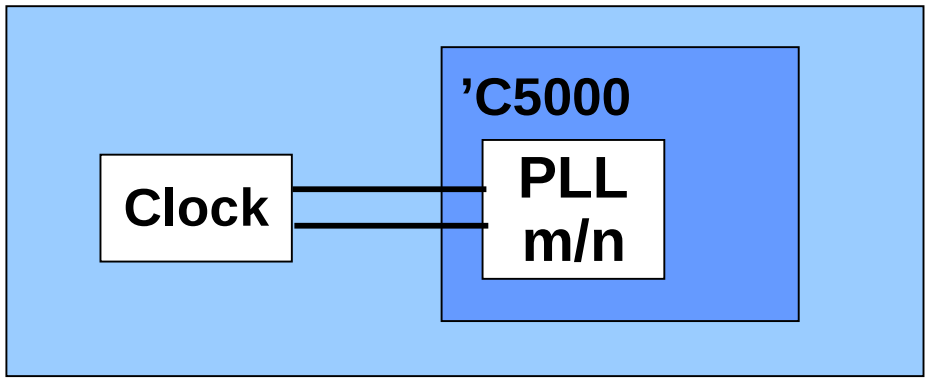
- ***High performance:***
  - Six DMA channels
  - Data moves from / to peripherals and memory
  - DMA higher priority than CPU
- **Programmable:**
  - Data widths
  - Priorities
- **Auto-initialization**

'C5000: Peripherals

'C5000	RAM/ROM	HPI	DMA	Serial Port
'C5420	200 K / --	16-bit	12-ch	6 McBSP
'C5402	16 K / 4 K	8-bit	6-ch	2 McBSP
'C5410	64 K / 16 K	8-bit	6-ch	3 McBSP
'C548/9	32 K / 16 K	8-bit	--	2 BSP/ 1 TDM
'C545/6	6 K / 48 K	8-bit	--	1 BSP/ 1 Std
'C542/3	10K / 2 K	8-bit	--	1 BSP/ 1 TDM
'C541	5 K / 28 K	8-bit	--	2 Std

Interfacing: PLL clock

- Instruction rate clock can be derived from slower external clock.
- ’C548 and above are programmable on the fly (32 ratios possible; no device reset required)
- PLL clock reduces EMI issues by lowering board-level clock rate.
- Lower cost/frequency oscillator (crystal) are multiplied internally.
- Device supports programmable delay for PLL lock time.



How do I work with TI's 'C5000?

How do I get my performance?

What performance can I expect?

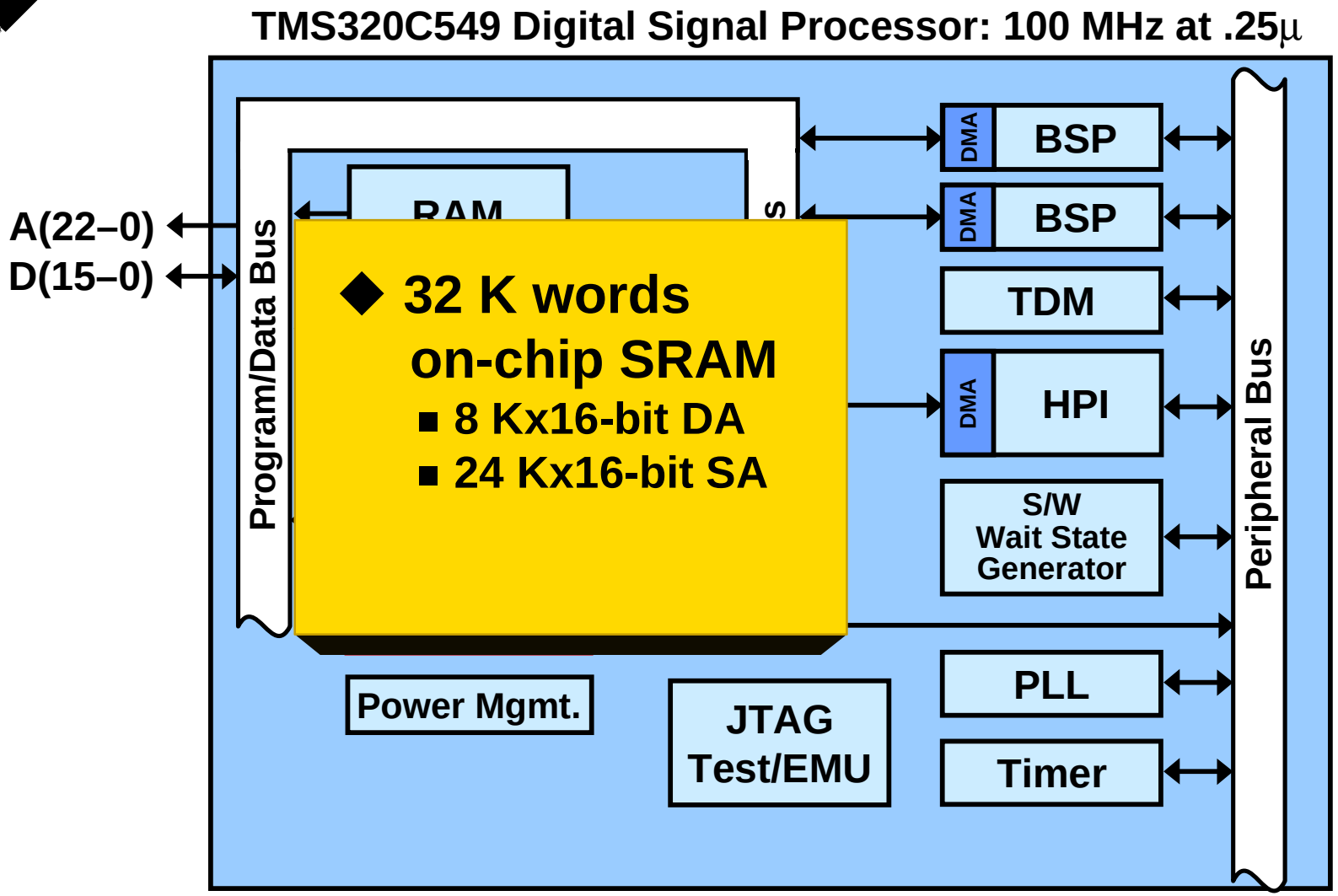
How do I interface easily?

What are the new 'C5000 devices?

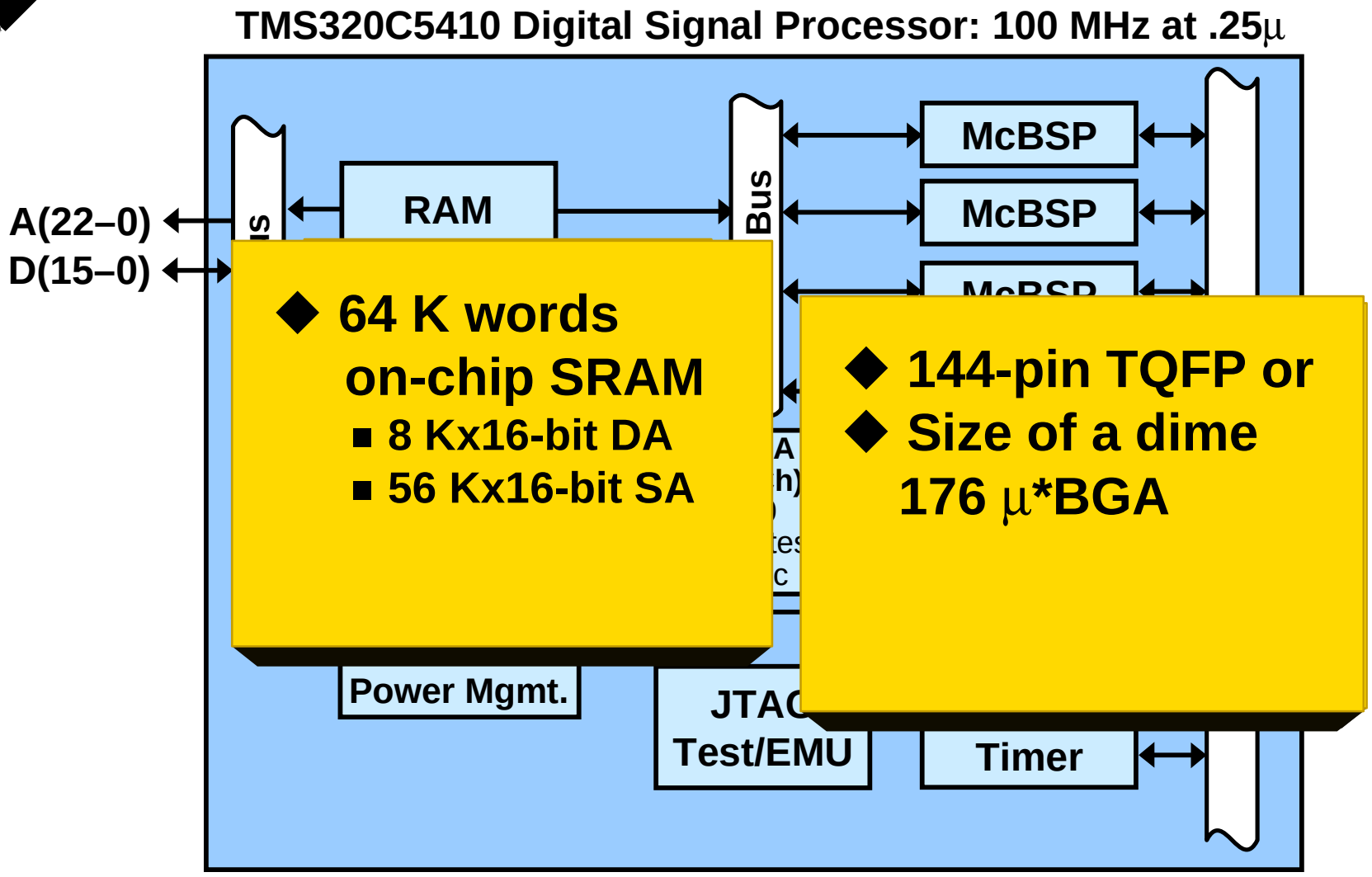
How do I minimize power consumption?

How does TI enable power-efficient performance at lower cost?

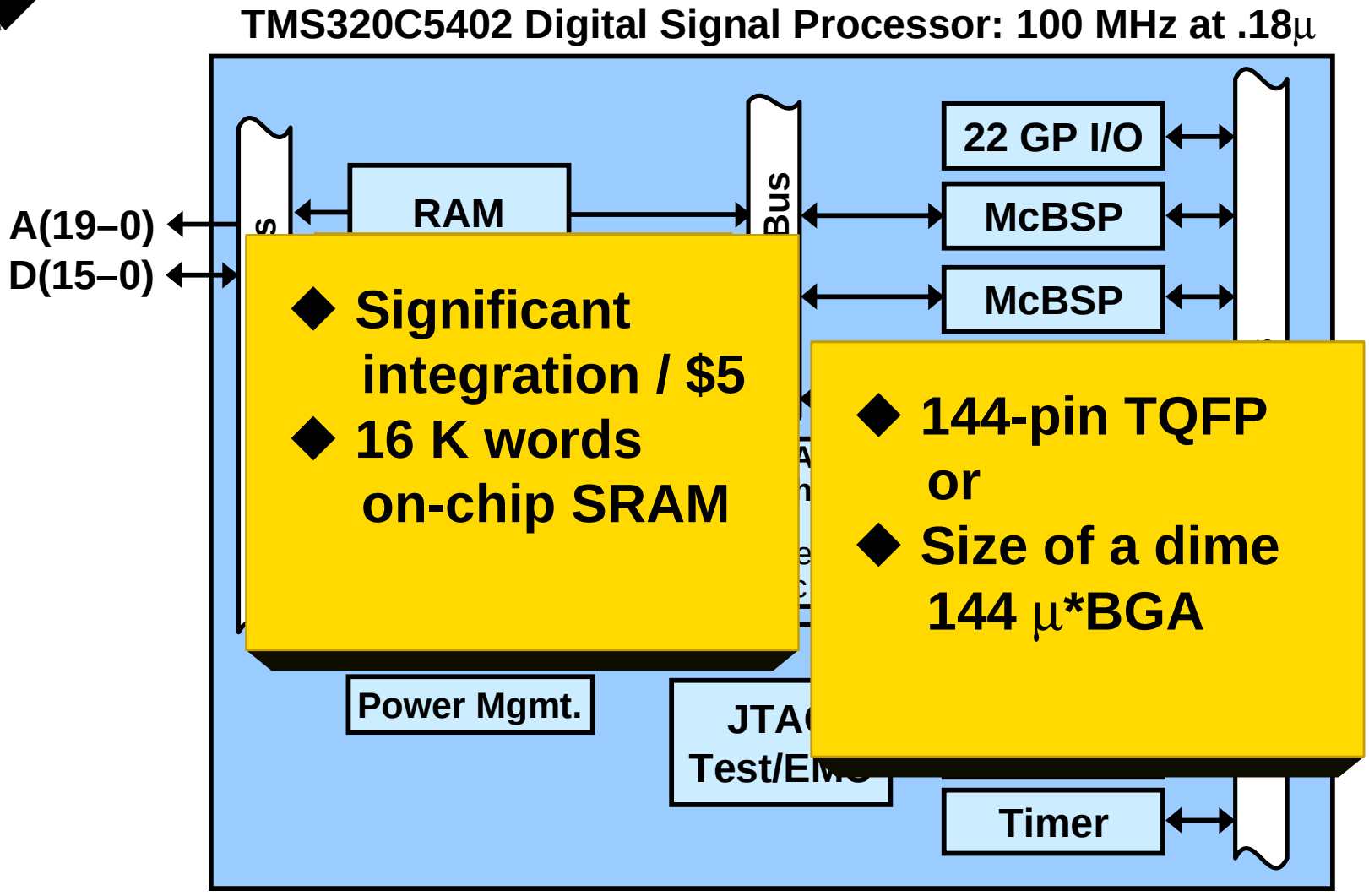
'C549: 100 MIPS in production



'C5410: Increasing on-chip integration

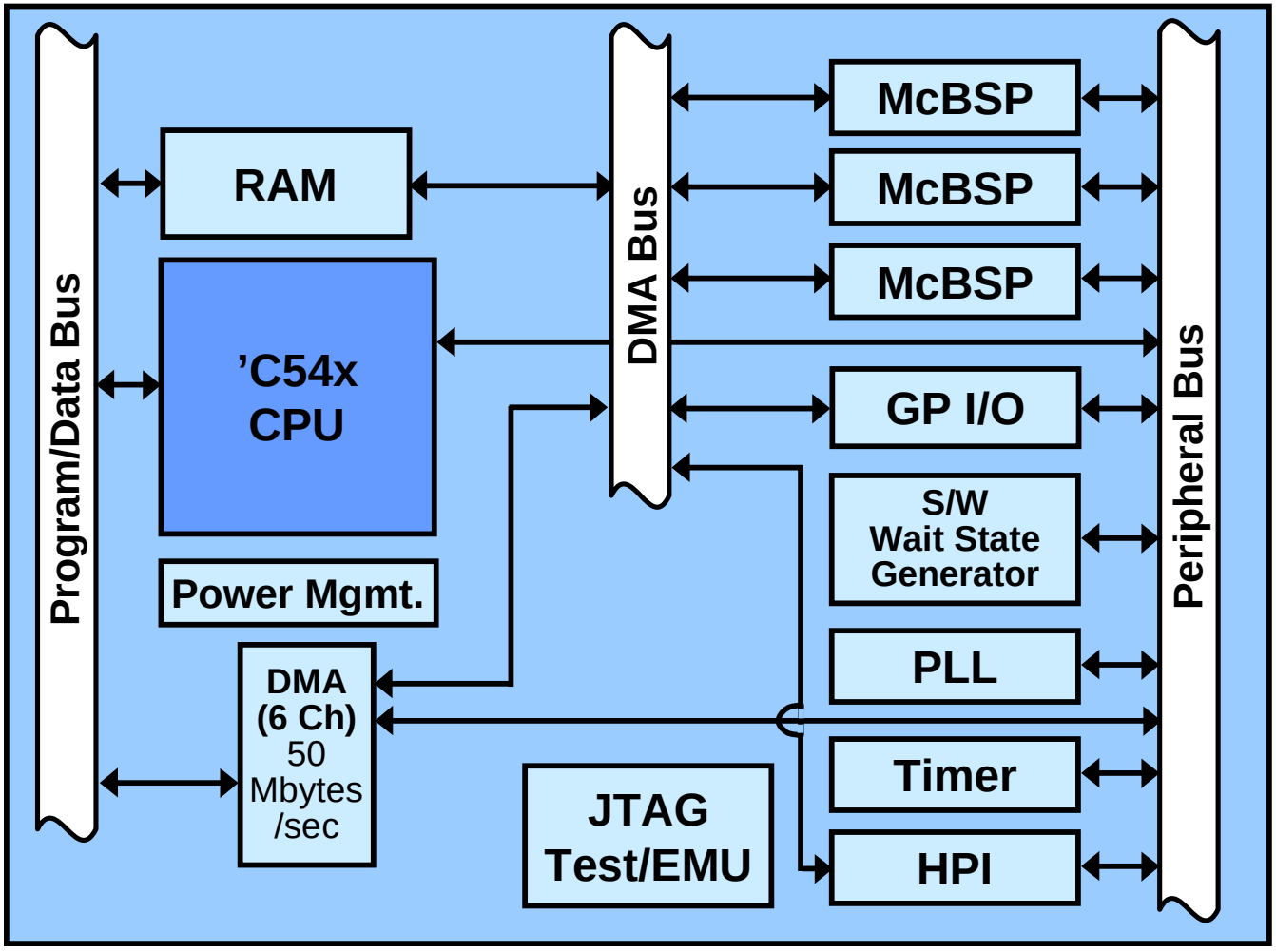


'C5402: 100 MIPS at $5

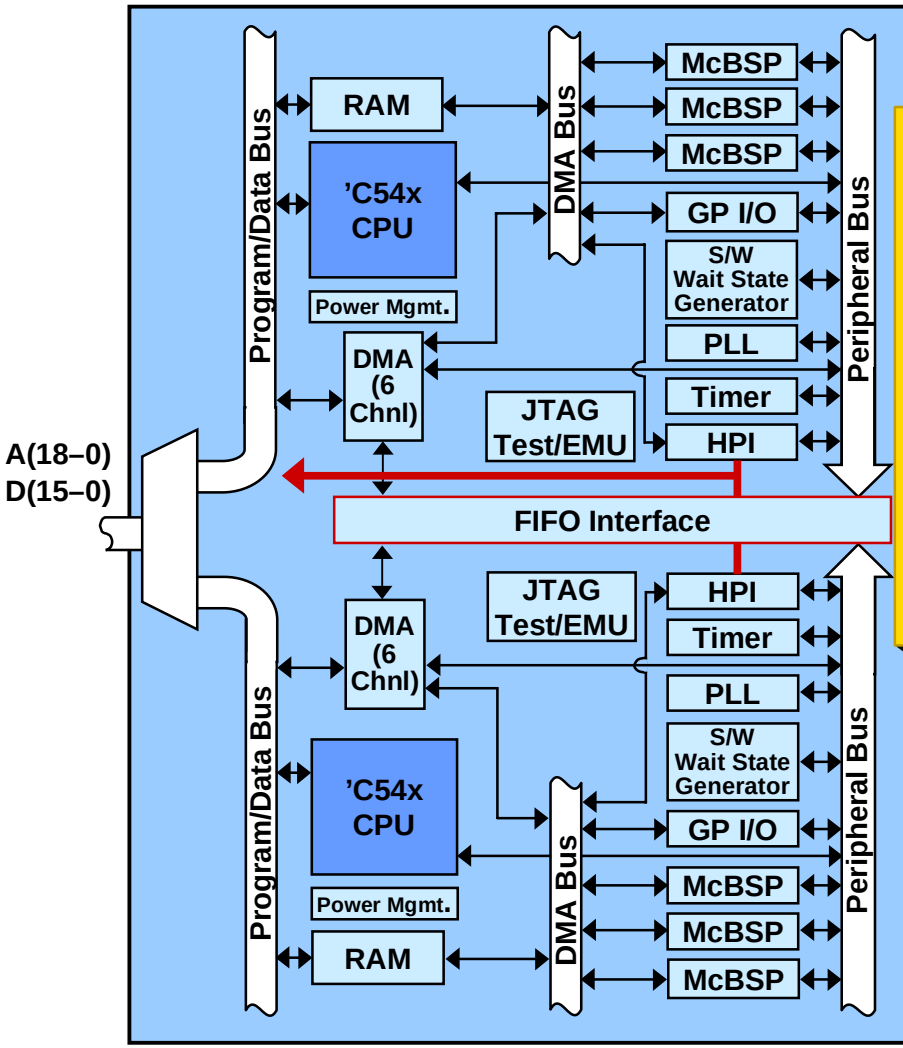


'C5420: Dual core 200 MIPS

TMS320C5420 Digital Signal Processor: 200 MHz at .18µ



'C5420: Dual core 200 MIPS



- 128-pin TQFP
  or
- Size of a dime
  144 μ*BGA

How do I work with TI's 'C5000?

How do I get my performance?

What performance can I expect?

How do I interface easily?

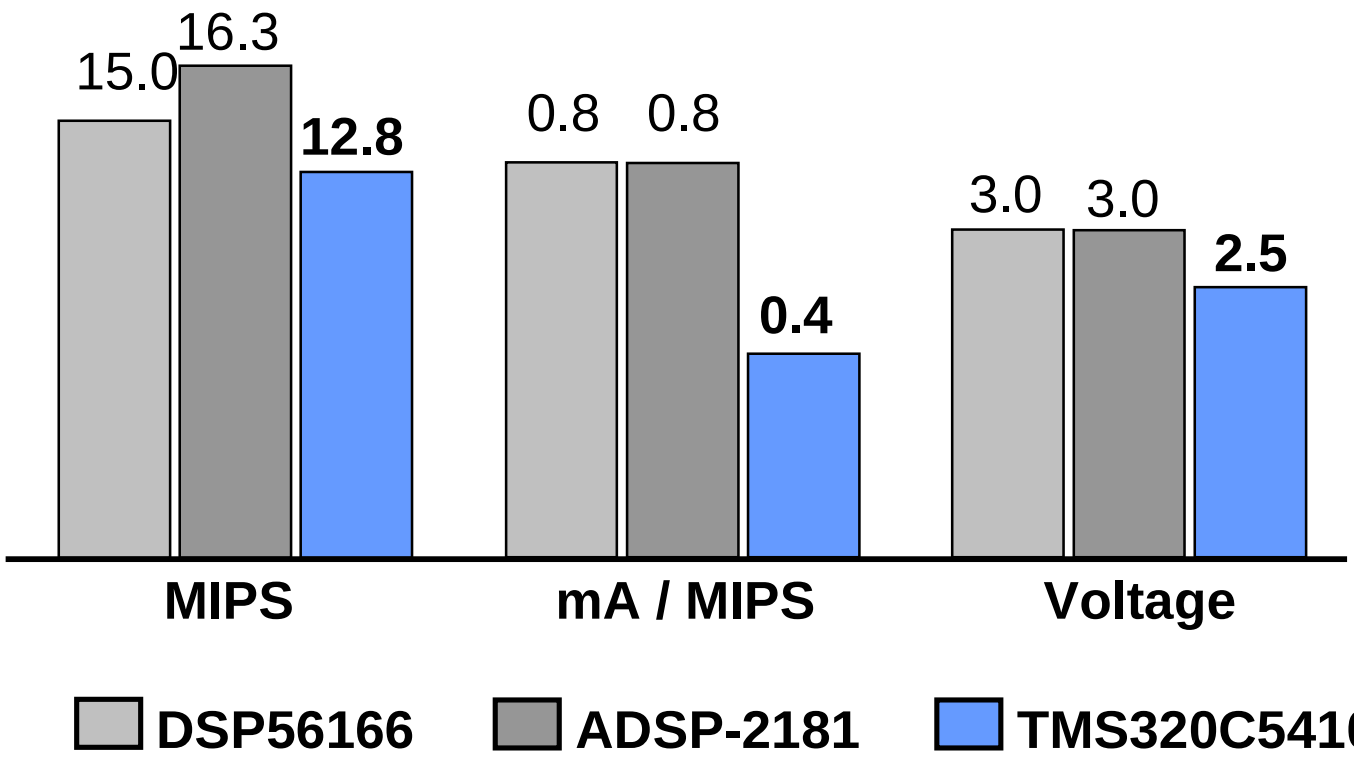
What are the new 'C5000 devices?

How do I minimize power consumption?

How does TI enable power-efficient performance at lower cost?

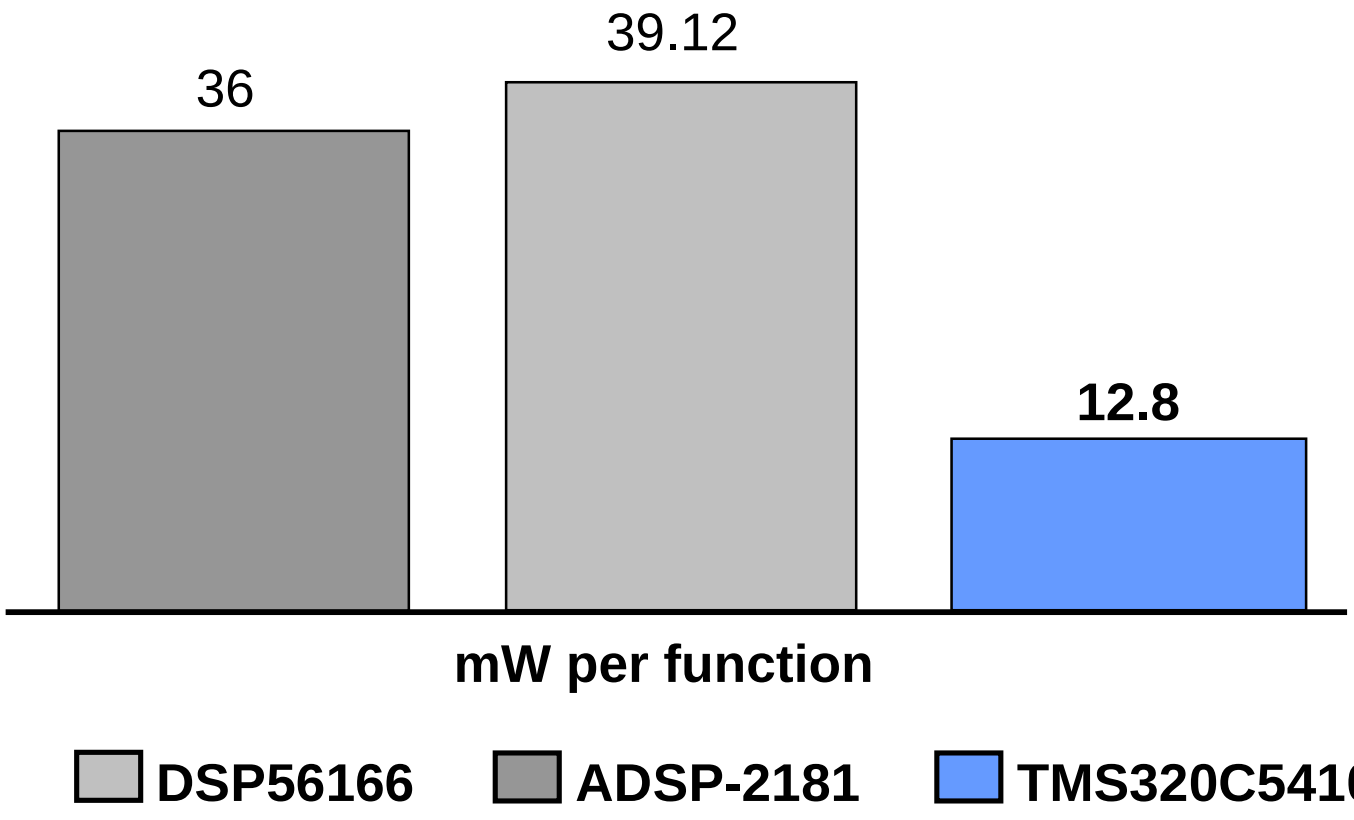
'C5000: mW per IS-54 VSELP

True measure of power consumption is:
(MIPS/algorithm) x (mA /MIPS) x (V) = mW / function

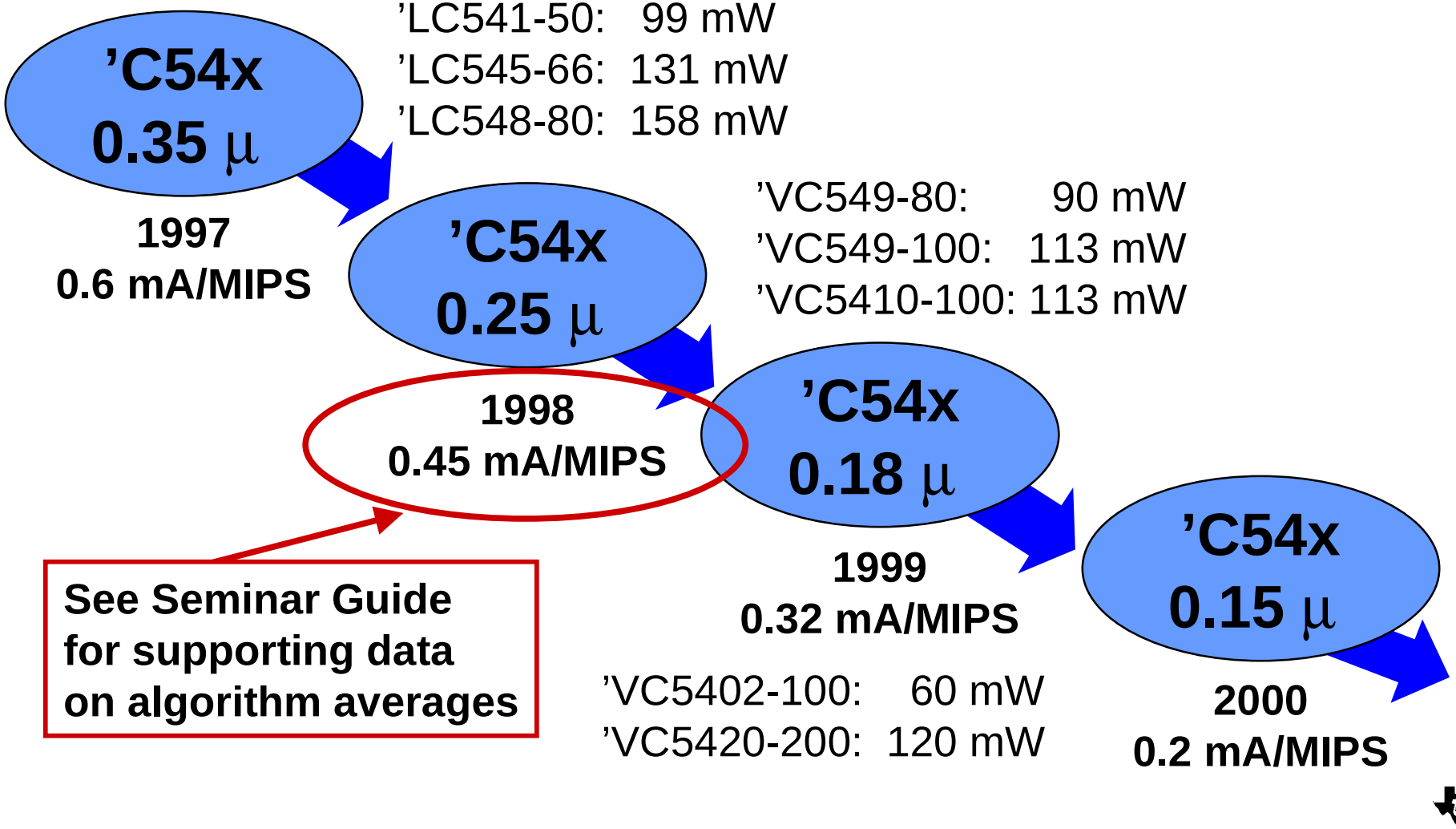


'C5000: mW per IS-54 VSELP

True measure of power consumption is:
(MIPS/algorithm) x (mA /MIPS) x (V) = mW / function



'C5000: Process technology



'C5000: Lowest power consumption

Mechanisms used on the 'C5000:

- **Bus keepers / Holders**
  maintain state of external bus
- **External Bus off control**
  disables the external bus
- **Static design**
  lower clock to DC
- **IDLE 1, 2, 3 modes**
  drop into various power-down modes
- **PLL options**
  use lower system clock
- **MIPS efficiency**
  requiring fewer MIPS

How do I work with TI's 'C5000?

How do I get my performance?

What performance can I expect?

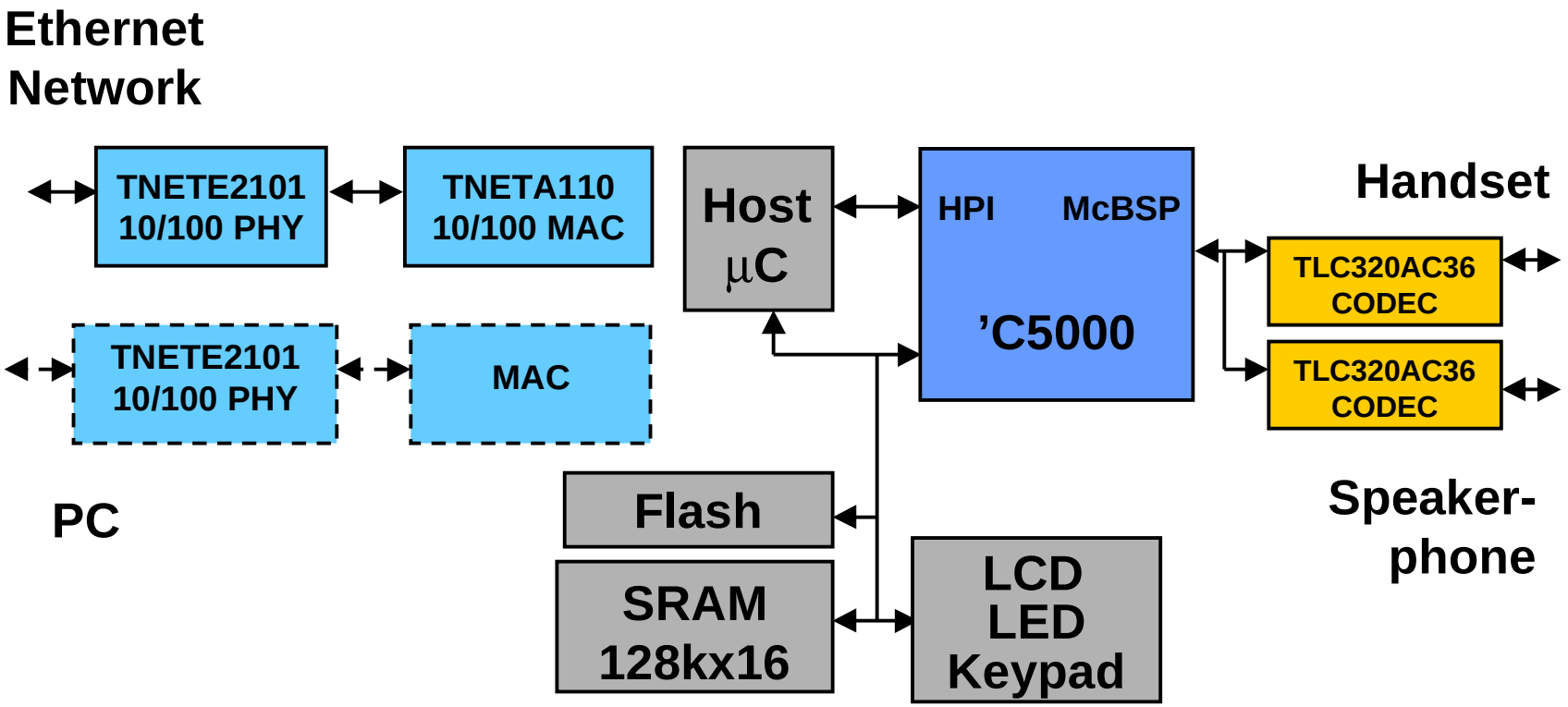
How do I interface easily?

What are the new 'C5000 devices?

How do I minimize power consumption?

How does TI enable power-efficient performance at lower cost?

Value per $: Internet phone example



'C5402: Internet phone example

100 MIPS @$5.00*

'C5402

OR

50 MIPS @$10.00**

ADI 2186

G.723.1 Vocoder
Echo Cancellation
DTMF
Voice Activity Detection
Full Duplex Speakerphone
Voice Packet Management

TOTAL 40 MIPS

* 25 Ku pricing. **Source: Minimum pricing on www.ednmag.com

'C5402: Internet phone example

100 MIPS @$5.00*

'C5402

OR

50 MIPS @$10.00**

ADI 2186

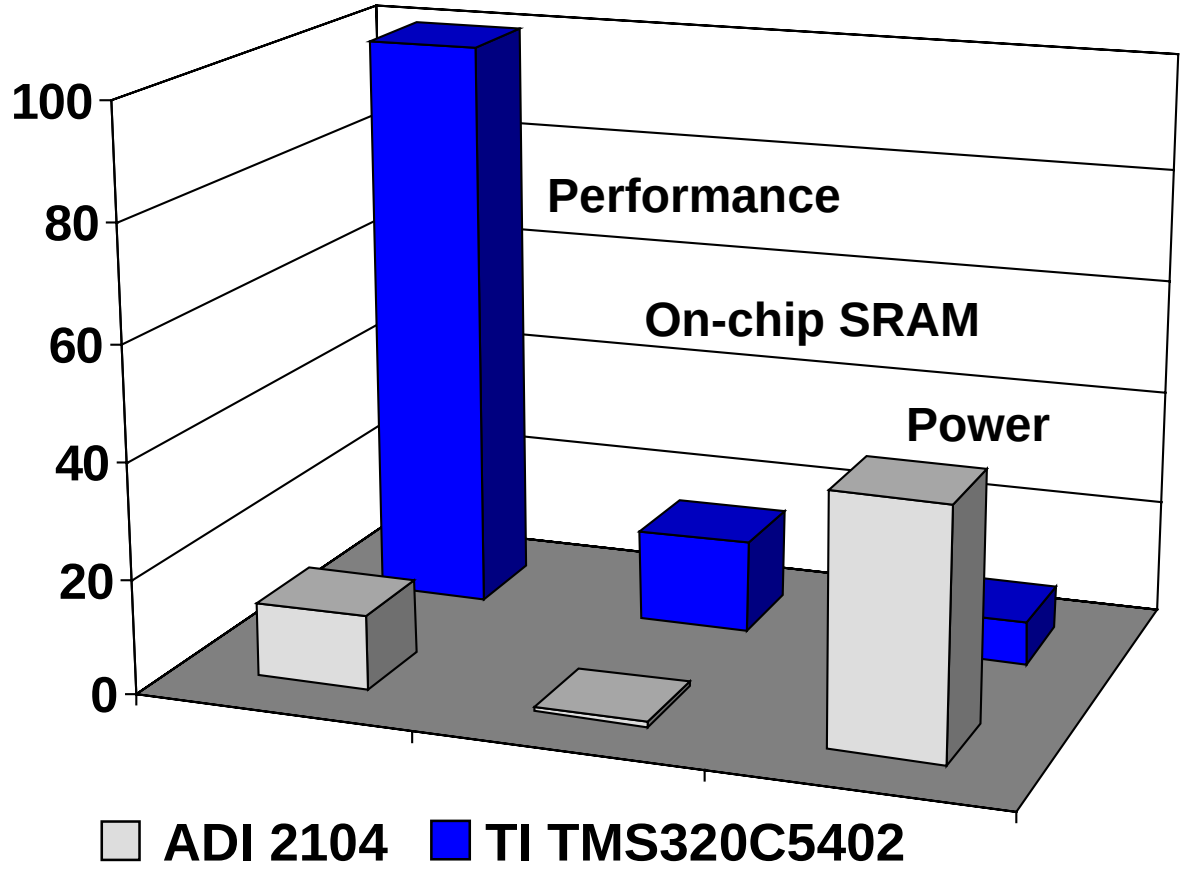
Plus... at half the cost, 'C5402 delivers:

- **Lower power, less space**
- **Glueless I/F to μC/Host, to multiple CODECs, and to fast or slow SRAM and Flash**
- **Expandable system for product differentiation, increased application functionality and seamless upgrades**

* 25 Ku pricing. **Source: Minimum pricing on www.ednmag.com

Competitor: What do you get for $5?

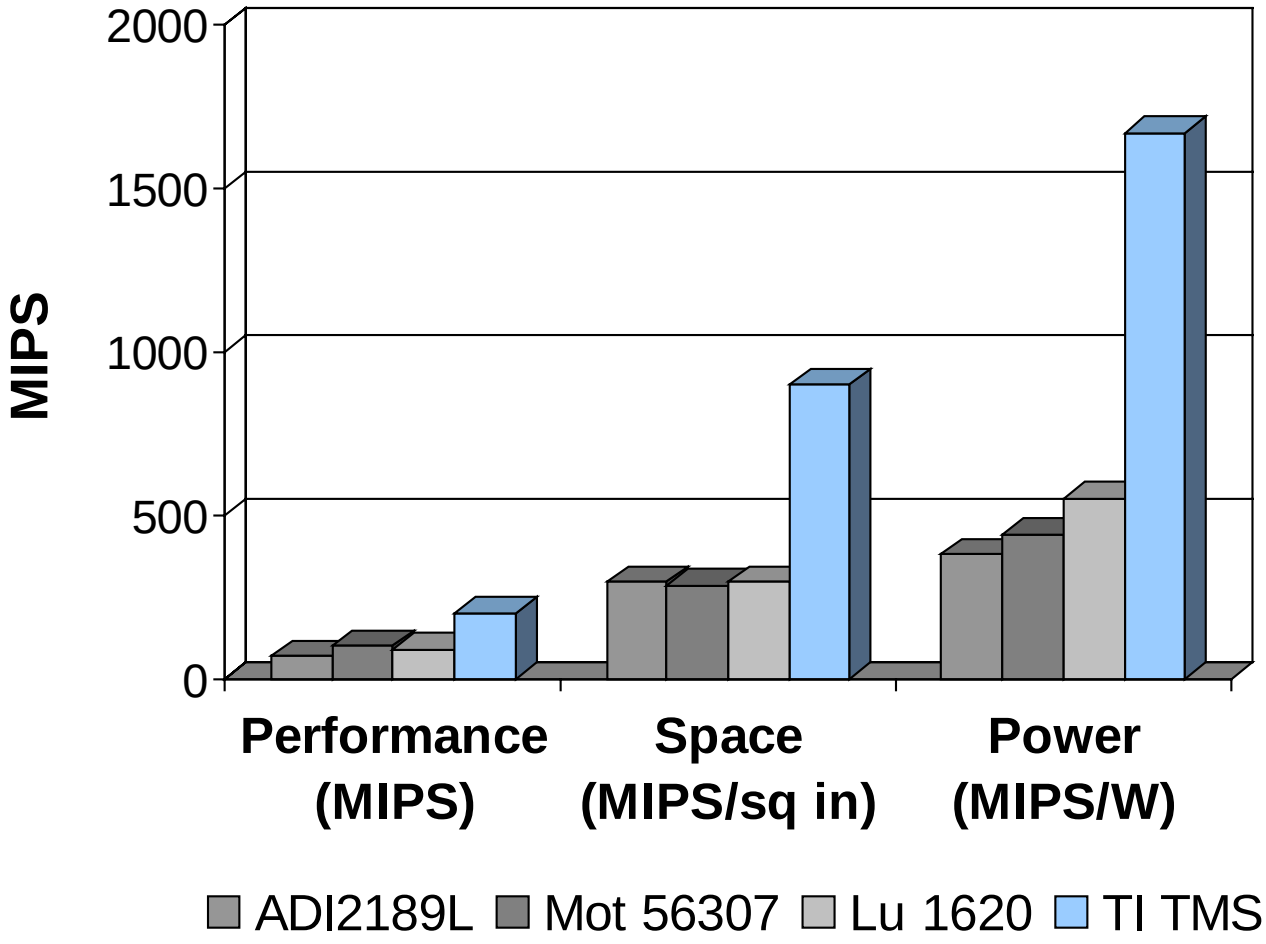
1/7th the performance, 1/20th on-chip SRAM, 6x the power consumption



Source: Analog Devices data sheet and web site

'C5420: More MIPS per careabout

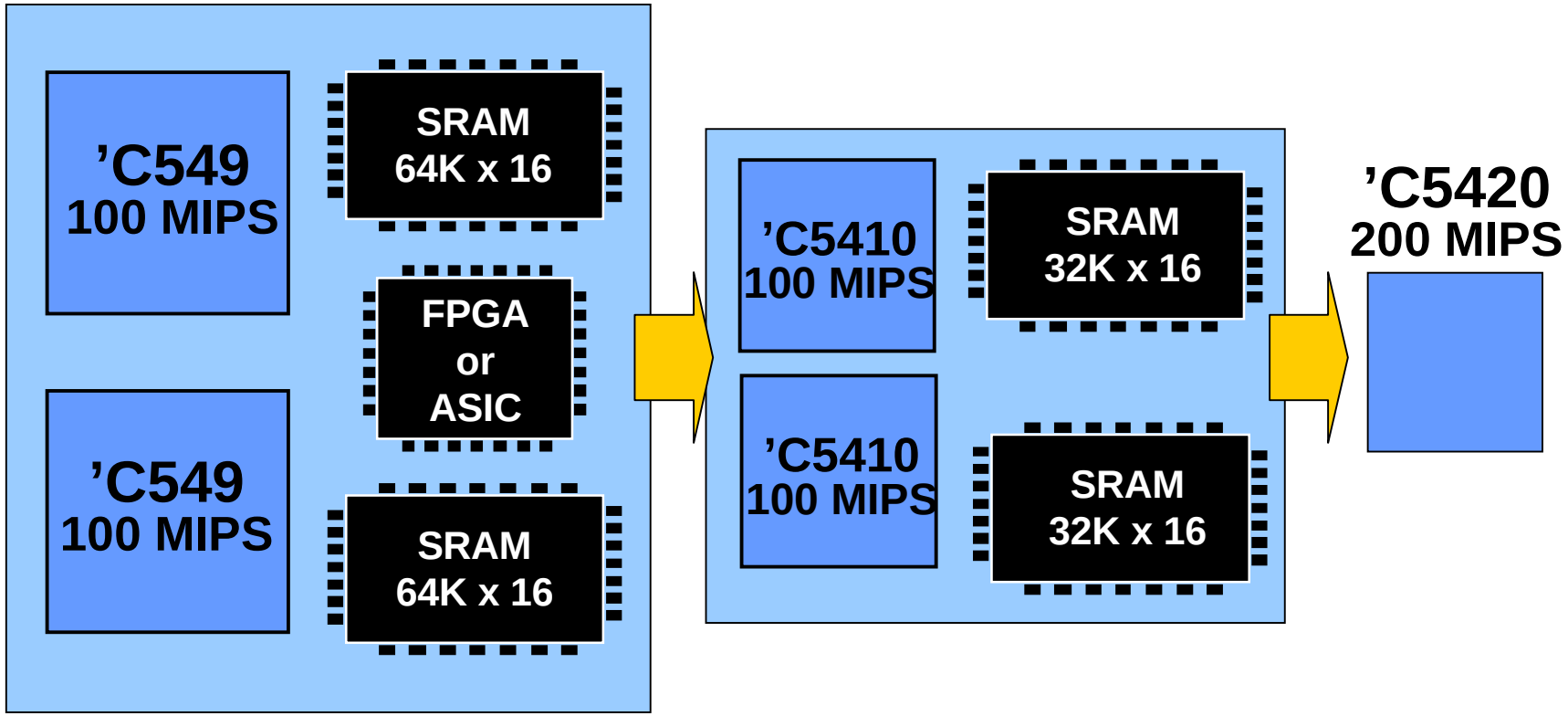
Up to 3x performance per Watt and 3x performance per square inch



Based on application of BDTImark to competitor data sheets, press releases, and web sites.

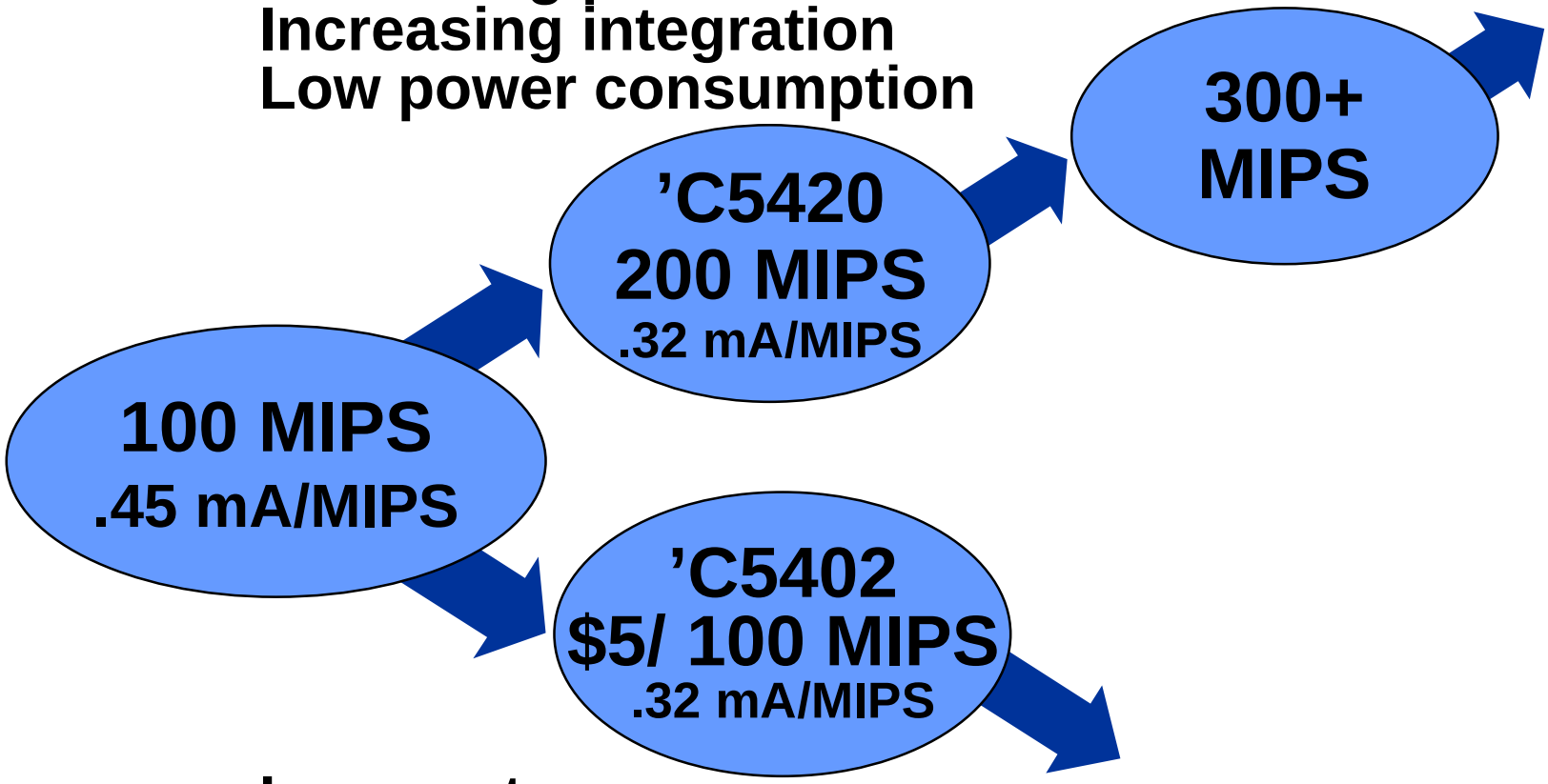
'C5420: Decreasing system costs

...through board space reduction



'C5000: Future directions

Increasing performance
Increasing integration
Low power consumption



Low cost
Power-efficient performance