Digital Signal Processors: fundamentals & system design

Lecture 1

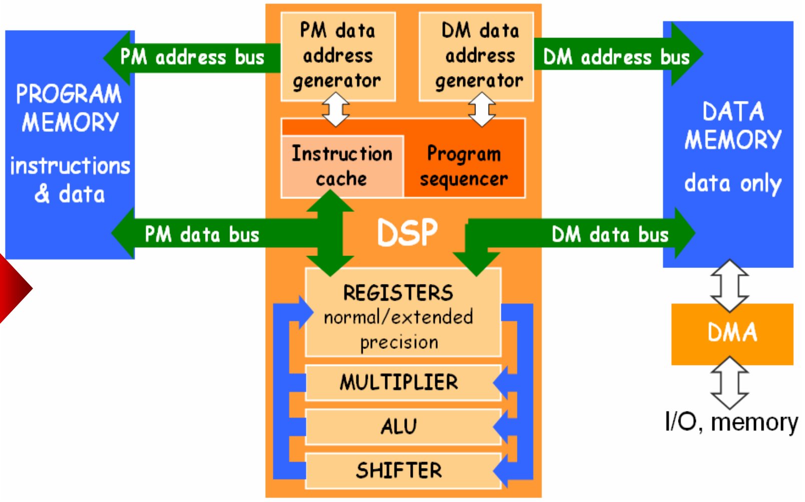
Maria Elena Angoletta
CERN

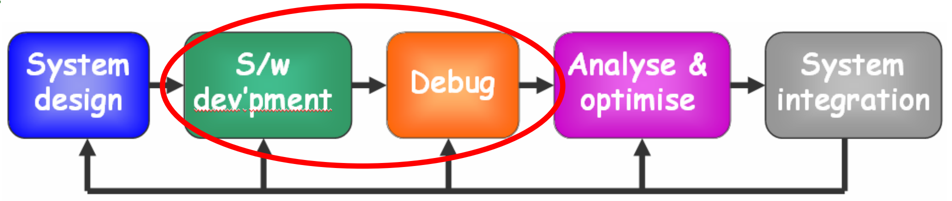
Topical CAS/Digital Signal Processing
Sigtuna, June 1-9, 2007

Lectures plan

Lecture 1 (now!)

introduction, evolution, DSP core + peripherals



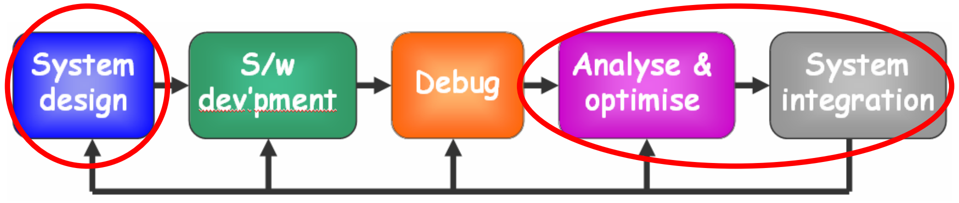


Lecture 2

DSP peripherals (cont'd), s/w dev'pment & debug.

Lecture 3

System optimisation, design & integration.



Lecture 1 - outline

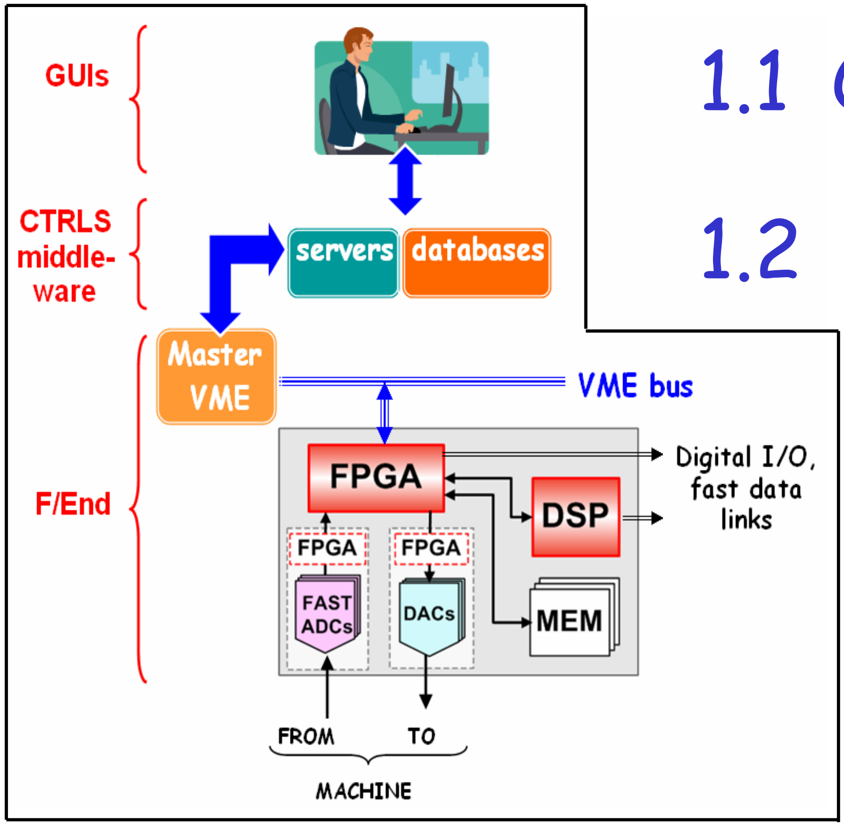
Chapter 1: Introduction

Chapter 2: DSP evolution & current scenery

Chapter 3: DSP core architecture

Chapter 4: DSP peripherals

Introduction



1.1 Overview

1.2 Use in accelerators

1.1 Overview

Digital Signal Processor (DSP):

μ-processor for DSPing applications.

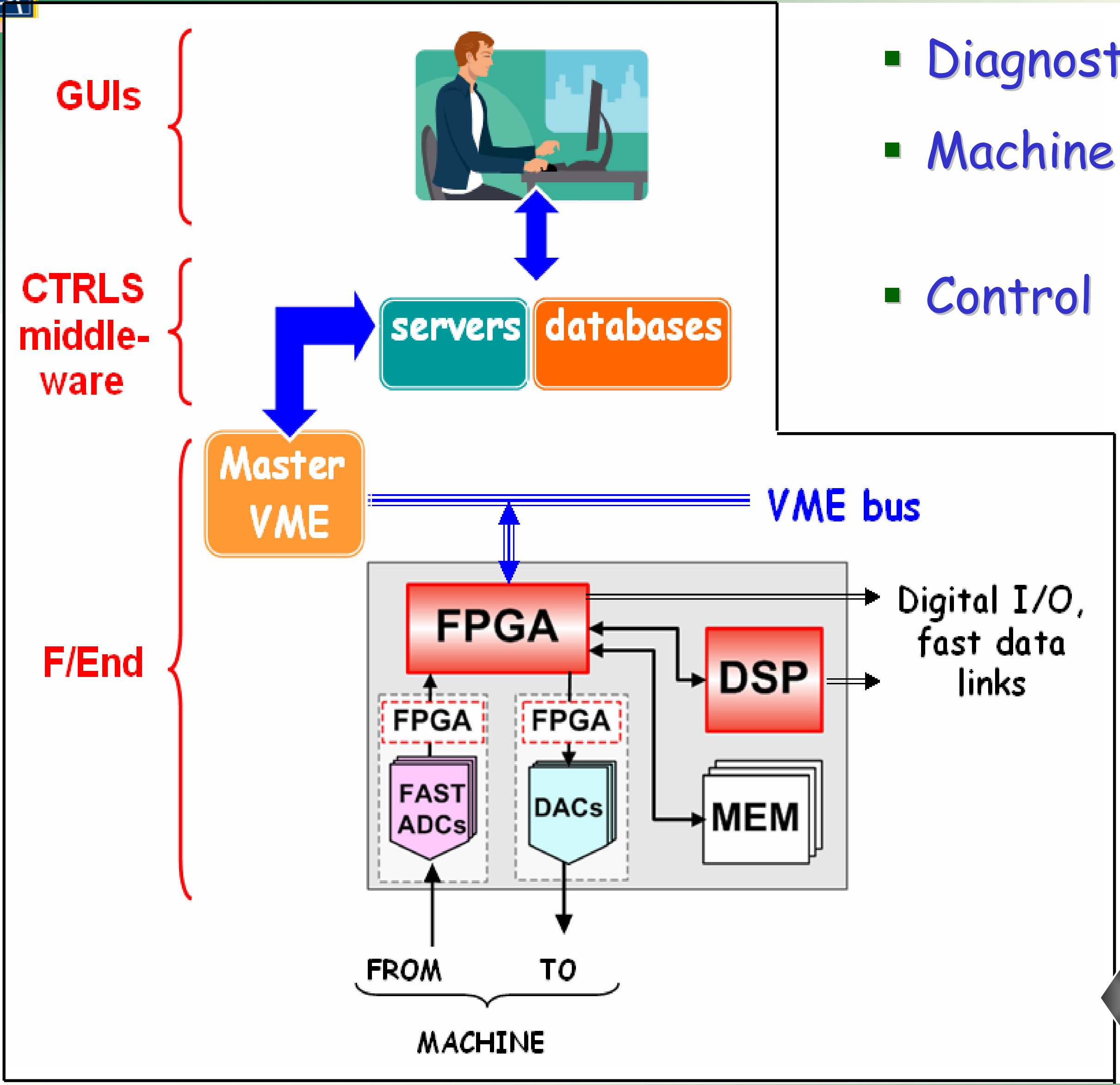
Characteristics

- **Real-time** data processing
- High **throughput**
- **Deterministic** operation
- **Re-programmable** by s/w

Key enabling technology for many electronics products:

- **Comms**: broadband & wireless
- **Automotive**: audio, driver assistance…
- **Consumer**: security, entertainment, toys
- **Instrumentation**: medical, test/measurement …
- **Military/aerospace**: radar, target detection …

1.2 Use in accelerators



- Diagnostics
- Machine protection
- Control
  - beam [LLRF]
  - power supplies
  - motors

DSP system example: classical controls structure.

Chapter 2 topics

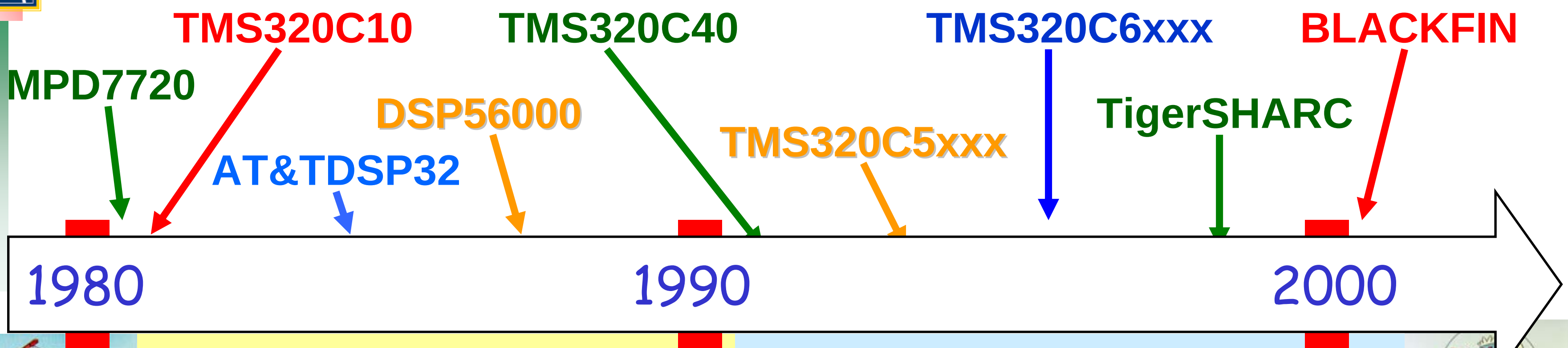
DSP evolution & current scenery

2.1 DSP evolution

2.2 Current mainstream DSPs

Summary

2.1 DSPs evolution: h/w features



DEVELOPMENT

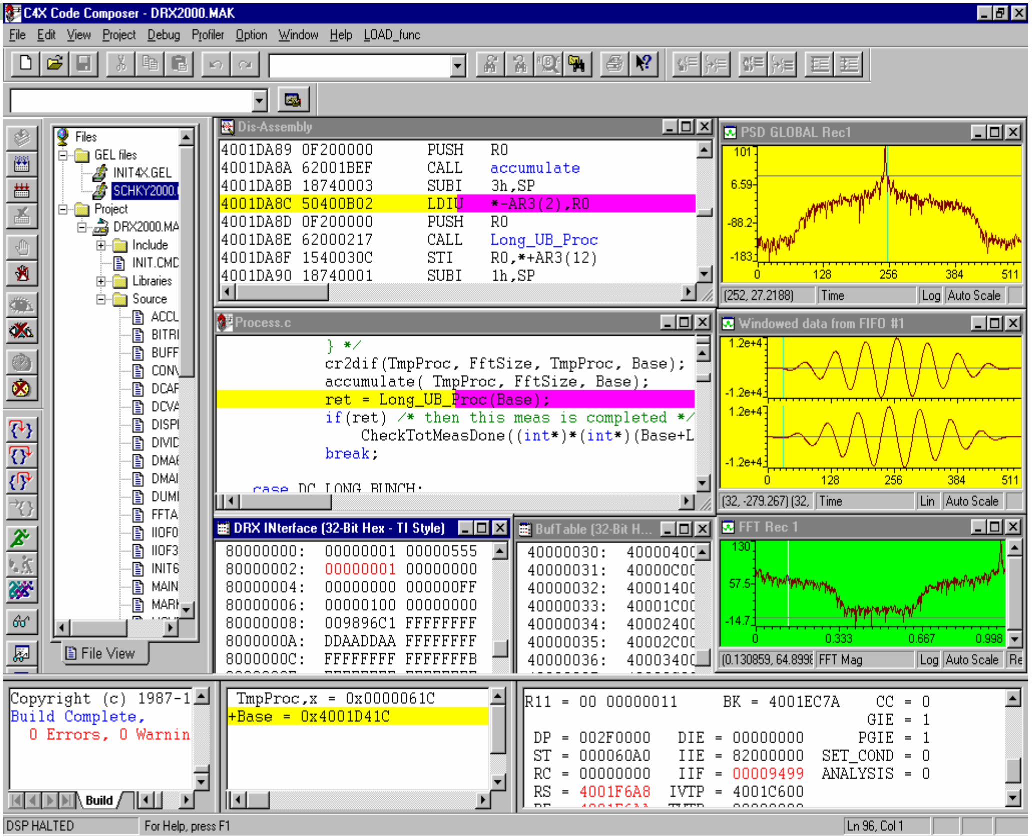
- Harvard architecture
- Data format:
  - early '80s: fixed point
  - late '80s: floating point (often *non* IEEE).
- DMA
- Fixed-width instruction set

CONSOLIDATION

- Parallel architectures
- Fewer manufacturers
- Multiprocessing support
- Late '90s: improved debug capabilities (ex: TI RTDX)
- Trend: wider/few families for code compatibility.

2.1 DSP evolution: s/w tools

- Spectacular evolution!
- Advanced compilers
  - Deal with h/w complexity
  - Efficient high-level languages
- Graphical programming
  - MATLAB
  - NI LabVIEW DSP Module (Hyperception RIDE)



```
4001DA89 0F200000      PUSH   R0
4001DA8A 62001BEF      CALL   accumulate
4001DA8B 18740003      SUBI   3h,SP
4001DA8C 50400B02      LDIU   *-AR3(2),R0
4001DA8D 0F200000      PUSH   R0
4001DA8E 62000217      CALL   Long_UB_Proc
4001DA8F 1540030C      STI    R0,*+AR3(12)
4001DA90 18740001      SUBI   1h,SP
```

```
            } */
            cr2dif(TmpProc, FftSize, TmpProc, Base);
            accumulate( TmpProc, FftSize, Base);
            ret = Long_UB_Proc(Base);
            if(ret) /* then this meas is completed */
               CheckTotMeasDone((int*)*(int*)(Base+L
            break;

      case DC_LONG_BUNCH:
```

```
80000000:  00000001 00000555
80000002:  00000001 00000000
80000004:  00000000 000000FF
80000006:  00000100 00000000
80000008:  009896C1 FFFFFFFF
8000000A:  DDAADDAA FFFFFFFF
8000000C:  FFFFFFFF FFFFFFFB
```

```
40000030:  4000040C
40000031:  40000C0C
40000032:  4000140C
40000033:  40001C0C
40000034:  4000240C
40000035:  40002C0C
40000036:  4000340C
```

```
Copyright (c) 1987-1
Build Complete,
  0 Errors, 0 Warnin
```

```
 TmpProc,x = 0x0000061C
+Base = 0x4001D41C
```

```
R11 = 00 00000011      BK = 4001EC7A      CC = 0
                                         GIE = 1
 DP = 002F0000   DIE = 00000000         PGIE = 1
 ST = 000060A0   IIE = 82000000     SET_COND = 0
 RC = 00000000   IIF = 00009499     ANALYSIS = 0
 RS = 4001F6A8  IVTP = 4001C600
```

```
DSP HALTED     For Help, press F1                    Ln 96, Col 1
```

Code Composer for TI 'C40 DSPs (CERN, 1999)

- High-performance simulators, emulators & debugging facilities
  - High-visibility into target (~no interferences)
  - Multiple DSP development & debugging in same JTAG chain.

2.1 DSPs evolution: device integration

	1980	1990	2000	≥ 2010
Die size [mm]	50	50	50	5
Technology [µm]	3	0.8	0.1	0.02
MIPS	5	40	5000	50000
MHz	20	80	1000	10000
RAM [Bytes]	256	2000	32000	1 million
Price [$]	150	15	5	0.15
Power [mW/MIPS]	250	12.5	0.1	0.001
Transistors	50000	500000	5 million	60 million
Wafer size [in.]	3	6	12	12

PROJECTED VALUES

2.1 DSPs evolution: device integration [2]

TI 'C6713

- Technology : 0.13 µm/6-level metal CMOS.
- Voltages: core supply = 1.26 V; I/O supply = 3.3 V.
- Core frequency: up to 225 MHz.
- Packages: 256-pin ball grid array or 208-pin plastic quad flatpack.

- **Operating voltage decrease** (5 V → 1.5 V):
  - Lower power consumption
  - Faster logic level transitions

Electromigration: higher speed increases performance ***but*** decreases chip durability

2.2 Current mainstream DSPs

3 main manufacturers: **Texas Instruments** (TI), **Analog Devices** (ADI) & **Freescale semiconductor** (formerly Motorola).

Mostly used for accelerators – TI & ADI

TI DSP families: TMS320Cxxxx

- **'C2x**: digital signal controller.
- **'C5x**: power-efficient.
- **'C6x**: high-performance.

ADI DSP families:

- **SHARC**: first ADI family (now 3 generations).
- **TigerSHARC**: high-performance for multiprocessor systems.
- **Blackfin**: high-performance, low power.

2.2 Current mainstream DSPs [2]

Low-cost fixed point DSPs

Applications: Audio, controls, power supplies, consumer.

Chip	Price [$] (*)	Data format [bit]	Max freq. [MHz]	On-chip memory [KByte]	Rivals	Notes
Freescale DSP563xx	4-47	24	275	24 K-648 K	SHARC, 'C67	Only mainstream DSP with 24-bit fixed point.
Freescale DSP5685x	3-20	18	120	20 K-612 K	'C28x	
TI 'C24x / 'C28x	2-8 3-14	16 32	40 150	13 K-69 K 40 K-294 K	DSP5685x	Hybrid µcontroller-DSP
TI 'C55x	4-17	16	300	80 K-376 K	Blackfin	

(*): Q3 2006 price for 10K units.

2.2 Current mainstream DSPs [3]

High-performance fixed point DSPs

Applications: Telecom infrastructure, automotive, video.

Chip	Price [$] (*)	Data format [bit]	Max freq. [MHz]	On-chip memory [Byte]	Rivals	Notes
ADI Blackfin	5-60	16	750	84 K-328 K	'C55x, 'C64x	
Freescale MSC71xx/ MSC81xx	13-184	16	300 1000	88 K–472 K 10.7 M	'C64x, Blackfin	Most chips are quad-core.
TI 'C64x / 'C64x+	15-208 180-260	16	1000	160 K–2112 K	MSC81xx/ MSC71xx	Adds 4- or 8-MAC features to 'C62x

(*): Q3 2006 price for 10K units.

2.2 Current mainstream DSPs [4]

Floating point DSPs

Applications: Military, imaging, audio.

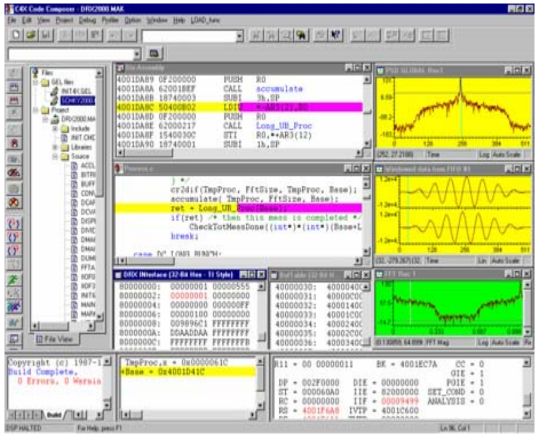
Chip	Price [$] (*)	Data format [bit]	Max freq. [MHz]	On-chip memory [Byte]	Rivals	Notes
ADI TigerSHARC	130-205	32 float. & 16 fixed point	600	512 K-3 M	'C67x, SHARC	VLIW + SIMD. On-chip DRAM.
ADI SHARC	5-15	32	200	512 K-768 K	'C67x	SIMD + multiprocessing
TI 'C67x/'C67x+	12-30	32	300	16 K-288 K	TigerSHARC, SHARC	Floating point version of 'C62x

(*): Q3 2006 price for 10K units.

Chapter 2 summary

- DSPs born early '80s: fast, real-time, deterministic processing.

- *Spectacular* evolution in software tools.
  - Compilers allow efficient high-level.
  - Graphical DSP programming (rapid prototyping).


C4X Code Composer - DFDC2000.MAK
File Edit View Project Debug Profiler Option Window Help LOAD_func
DSP HALTED
For Help, press F1

```
4001DA89 0F200000    PUSH   R0
4001DA8A 62001BEF    CALL   accumulate
4001DA8B 18740003    SUBI   3h,SP
4001DA8C 50400B02    LDI    *-AR3(2),R0
4001DA8D 0F200000    PUSH   R0
4001DA8E 62000217    CALL   Long_UB_Proc
4001DA8F 1540030C    STI    R0,*+AR3(12)
4001DA90 18740001    SUBI   1h,SP
```

```
    } */
    cr2dif(TmpProc, FftSize, TmpProc, Base);
    accumulate( TmpProc, FftSize, Base);
    ret = Long_UB_Proc(Base);
    if(ret) /* then this meas is completed */
        CheckTotMessDone((int*)*(int*)(Base+L
    break;
```

```
80000000:  00000001 00000555
80000002:  00000001 00000000
80000004:  00000000 000000FF
80000006:  00000100 00000000
80000008:  009896C1 FFFFFFFF
8000000A:  DDAADDAA FFFFFFFF
8000000C:  FFFFFFFF FFFFFFFB
```

```
40000030:  400004C
40000031:  40000C0C
40000032:  4000140C
40000033:  40001C0C
40000034:  4000240C
40000035:  40002C0C
40000036:  4000340C
```

```
Copyright (c) 1987-1
Build Complete,
  0 Errors, 0 Warnin
```

```
TmpProc,x = 0x0000061C
+Base = 0x4001D41C
```

```
R11 = 00 00000011     BK = 4001EC7A      CC = 0
                                        GIE = 1
DP = 002F0000   DIE = 00000000        PGIE = 1
ST = 000060A0   IIE = 82000000    SET_COND = 0
RC = 00000000   IIF = 00009499    ANALYSIS = 0
RS = 4001F6A8  IVTP = 4001C600
```

- Great evolution in device integration.
  - Operating voltage decrease: lower power consumption & faster logic.
  - Core frequency increase.
  - Beware: ***electromigration***.

- Main families for accelerator applications: ADI & TI.

DSP core architecture

3.1 Introduction

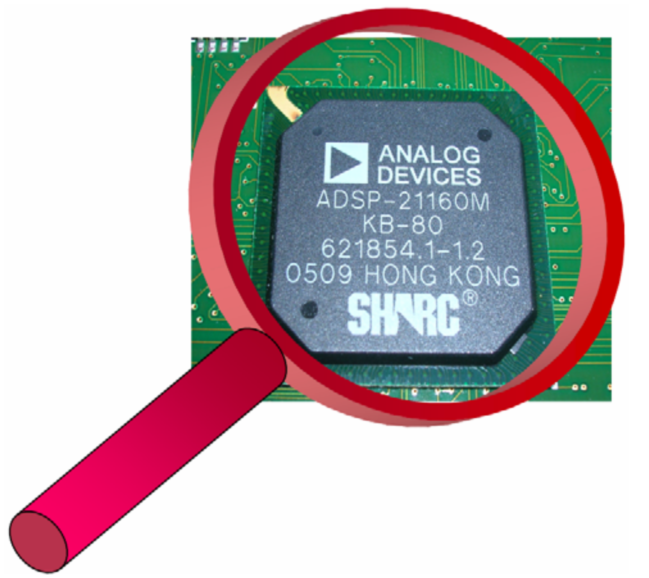
3.2 Fast data access

3.3 Fast computation

3.4 Numerical fidelity

3.5 Fast-execution control

Summary



3.1 DSP core architecture: intro

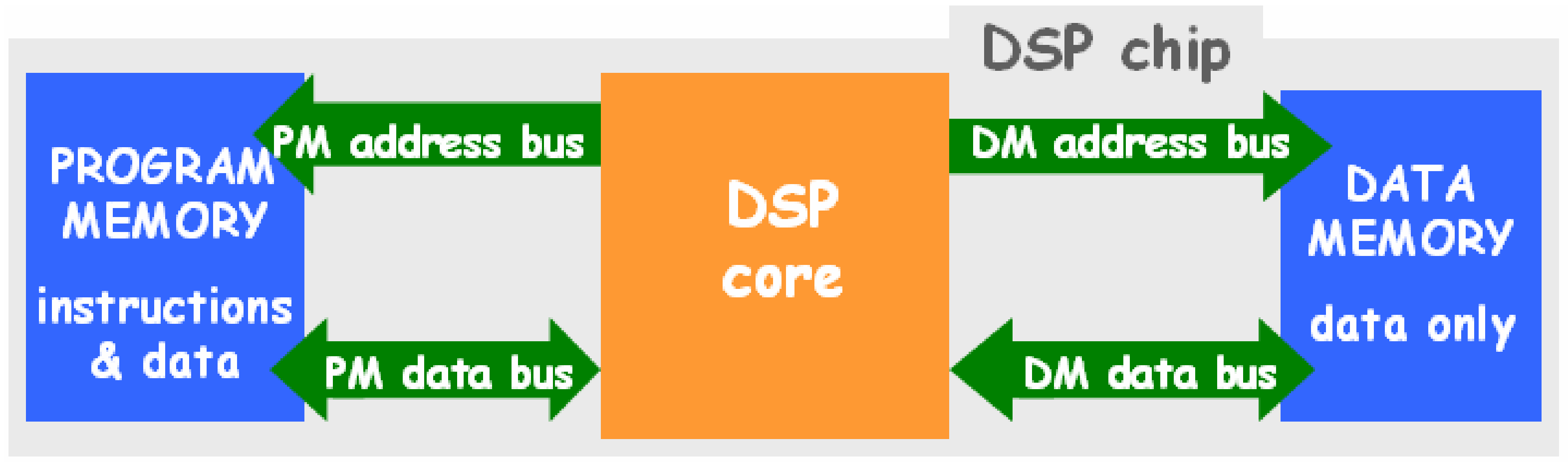
Shaped by **predictable real-time DSPing** !

ex: FIR $y(n) = \sum_{k=0}^{M} a_k \cdot x(n-k)$

Requirements	How
3.2 Fast data access	High-BW memory architectures. Specialised addressing modes. Direct Memory Access (DMA).
3.3 Fast computation	MAC-centred. Pipelining. Parallel architectures (VLIW, SIMD).
3.4 Numerical fidelity	Wide accumulator regs, guard bits ..
3.5 Fast-execution control	H/w assisted zero-overhead loops, shadow registers …

3.2 Fast data access

a) High-BW memory architectures



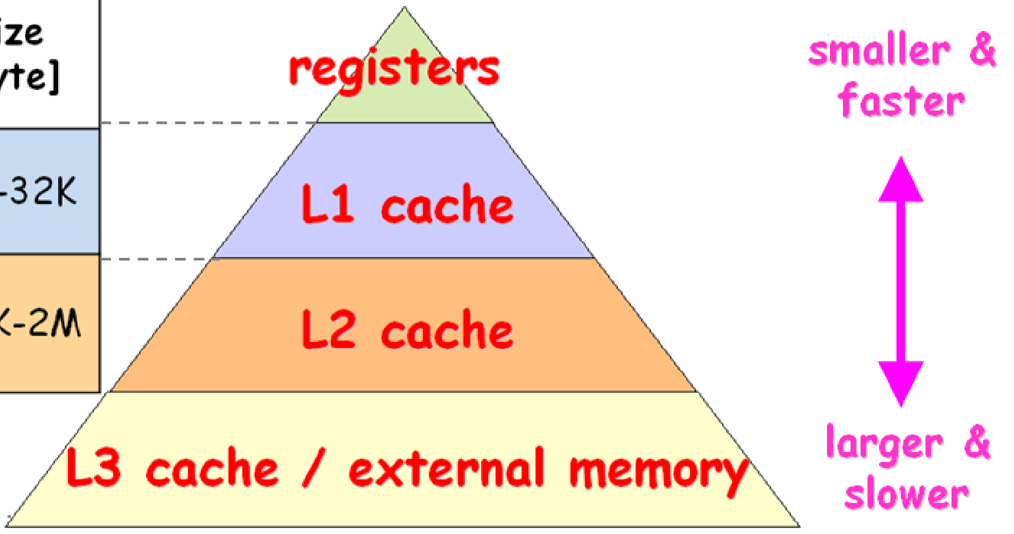
Harvard architecture

- Builds upon Harvard architecture & improves throughput.
- More buses than for Von Neumann: efficient *but* expensive.
- Instruction cache: loops instructions pre-fetched & buffered.
  - → memory BW used for data fetch.
- Data cache on newer DSPs.

3.2 Fast data access

a) High-BW memory architectures – cont'd.

Access time [ns]	h/w	Size [Byte]
1	~5 transistor /cell	16K-32K
5-10	~2 transistor /cell	512K-2M
10-50		



Hierarchical memory architecture

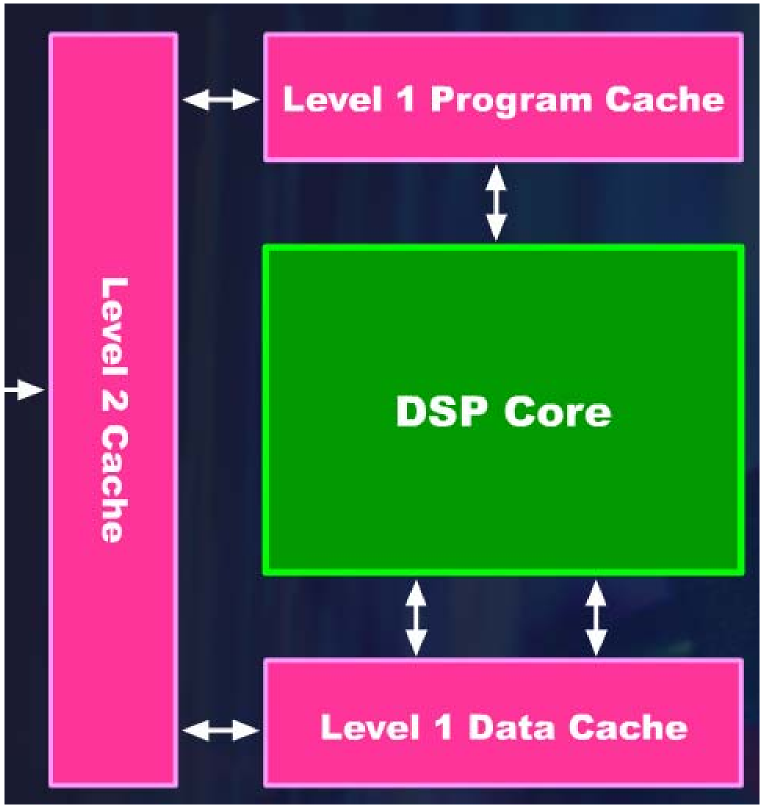
Cache limitations: unpredictability of cache hits → difficult worst case scenario prediction.

NB: user may lock cache for deterministic execution of critical sections.

3.2 Fast data access example: 'C6713

'C6713 cache

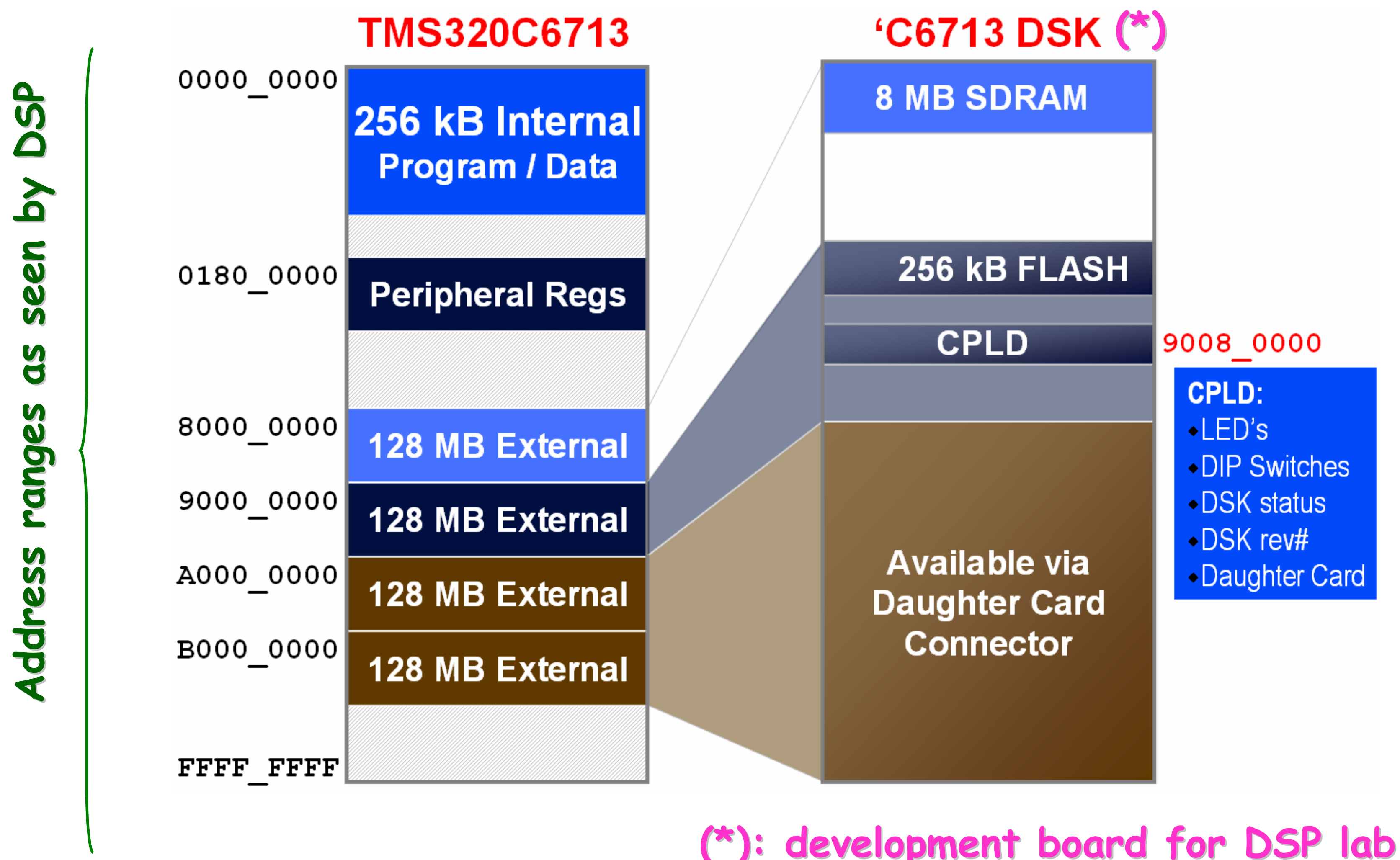
- 2-level cache architecture, Level1 (L1) & Level2 (L2).
- L1 cache: 8 KByte total
  - 4 KByte program cache (L1P)
  - 4 KByte data cache (L1D)
- L2 cache: 256 Kbyte total
  - 64 KByte *dual cache:* mapped memory, cache or combination of the two.
  - 192 KByte mapped SRAM.



TI 'C6713 cache architecture

3.2 Fast data access example: 'C6713 [2]

'C6713 memory mapping

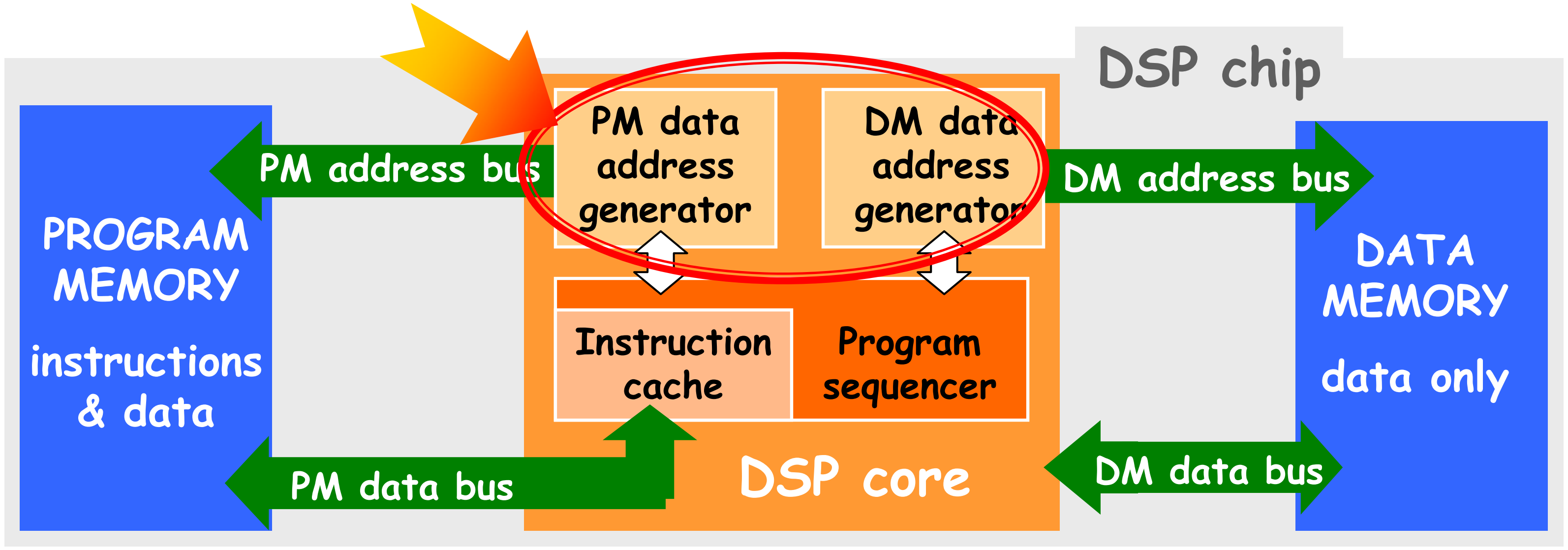


(*): development board for DSP lab.

3.2 Fast data access

b) Specialised addressing modes

Data Address Generator (**DAG**) manages address update in DSP-common patterns.



- Addressing examples:

Circular : read & write pointers managed by h/w.

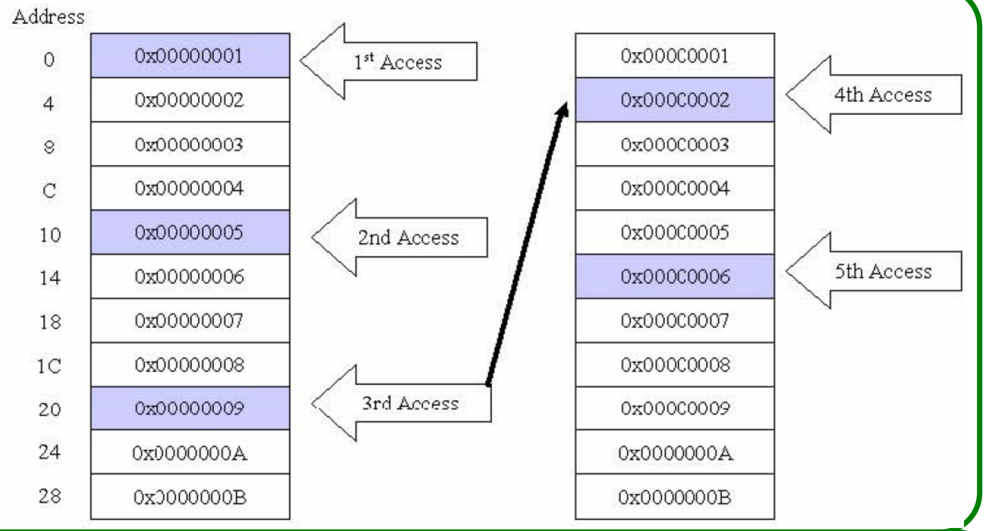
Bit-reversed : FFT-computation.

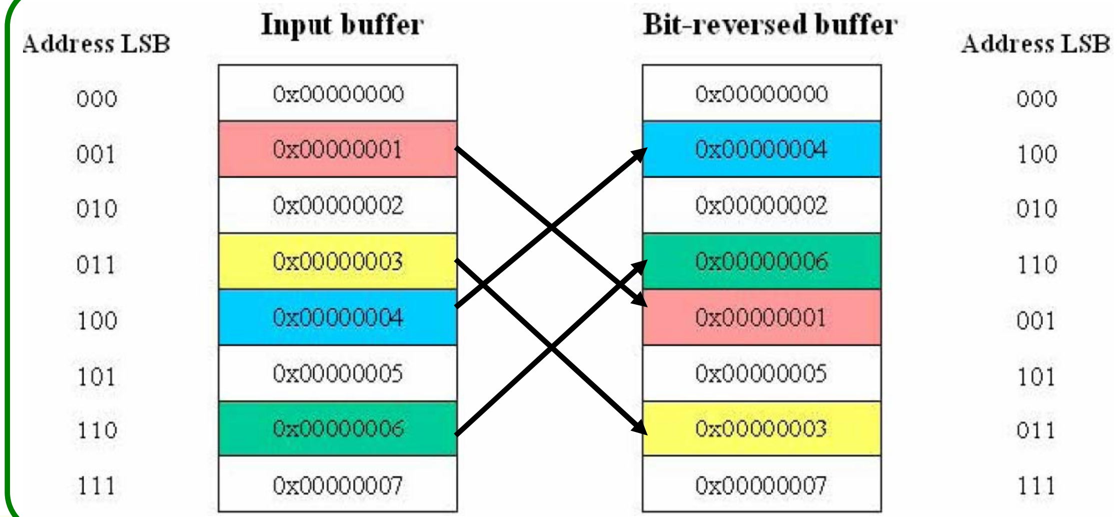
3.2 Fast data access

b) Specialised addressing modes – cont'd

Circular addressing example:

buffer wrap-around



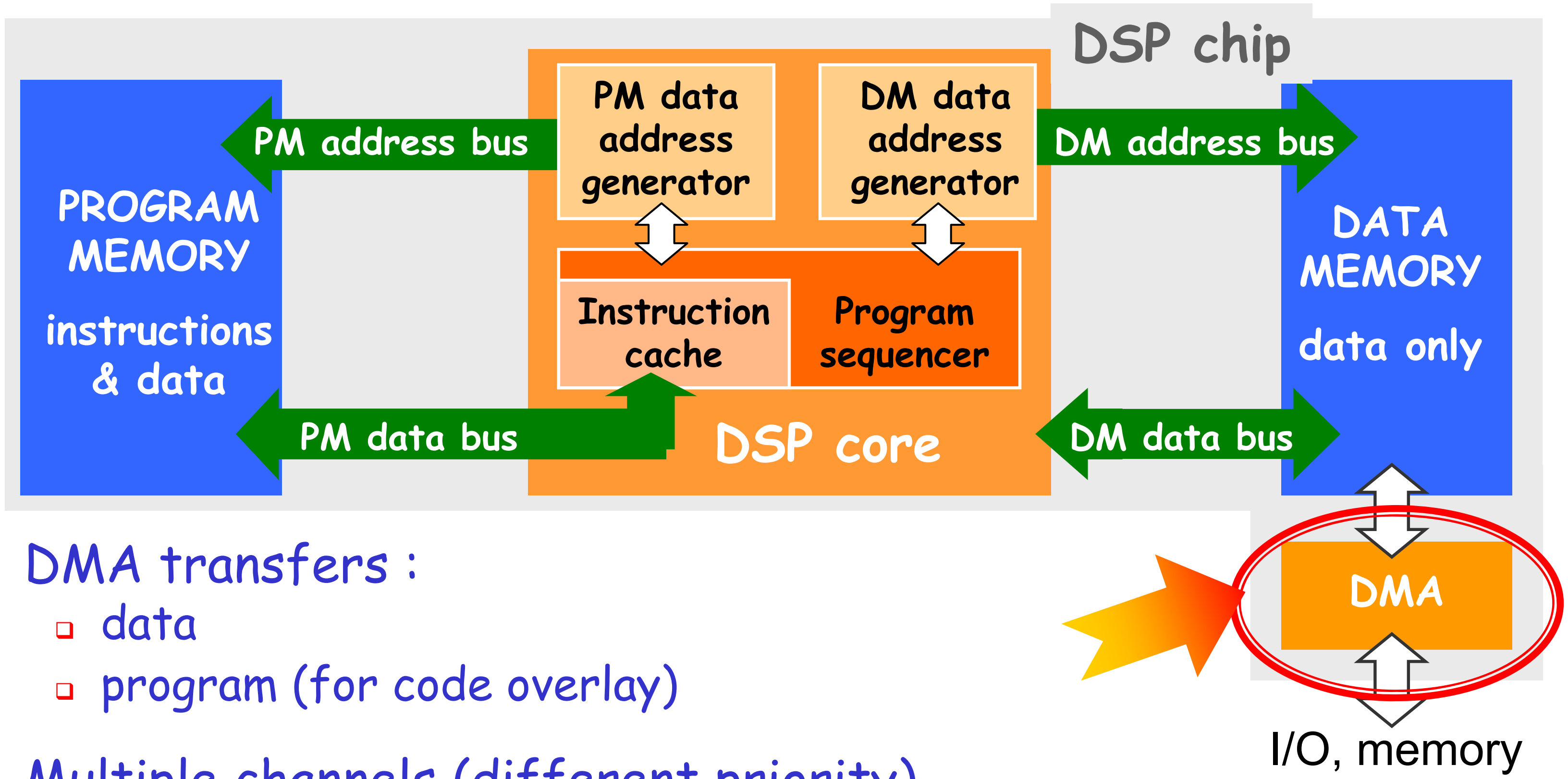


Bit-reversing addressing: data flow

3.2 Fast data access

c) Direct Memory Access (DMA)

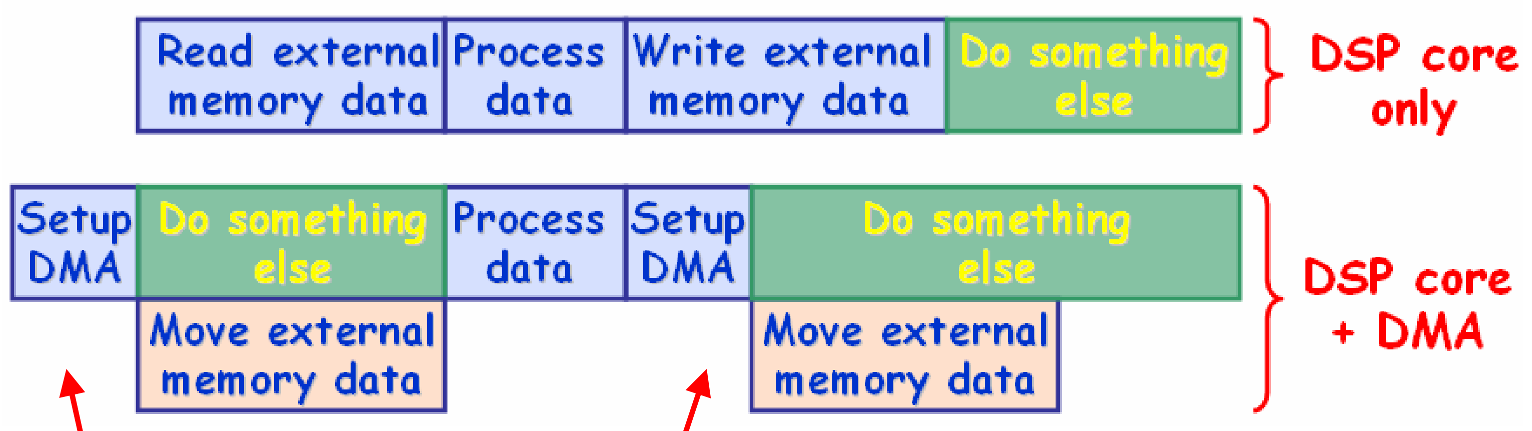
DMA coprocessor: memory transfers without DSP core intervention



- DMA transfers :
  - data
  - program (for code overlay)
- Multiple channels (different priority).
- Arbitration DSP core-DMA for colliding memory access.

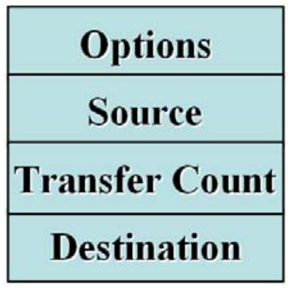
3.2 Fast data access

c) Direct Memory Access (DMA) – cont'd.



DMA setup

- **register-** or **RAM-based**.
- typical info for setup



- DMA transfer: triggered by **DSP core** or by **event**.
- DMA can generate interrupt when transfer completed.

3.2 Fast data access

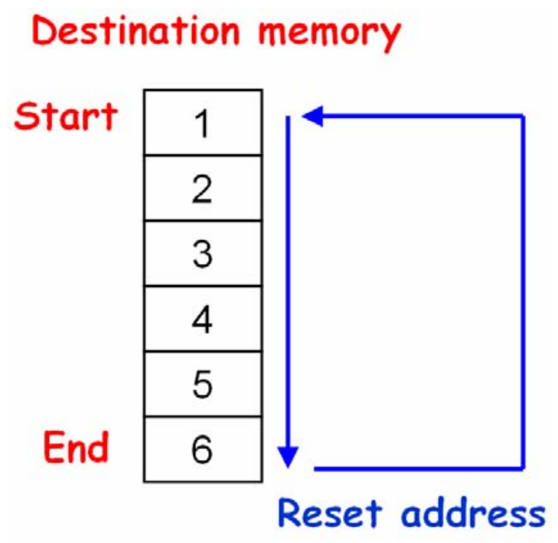
c) Direct Memory Access (DMA) – cont'd.

FLEXIBLE & POWERFUL CONFIGURATIONS

Examples:

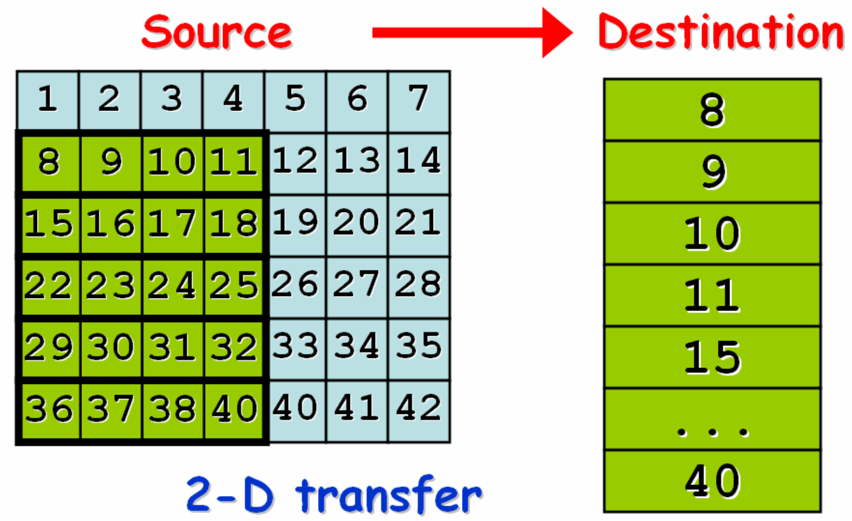
Chained DMA

DMA transfer completion starts new transfer.



Data formatting

Multi-dimensional data transfers available



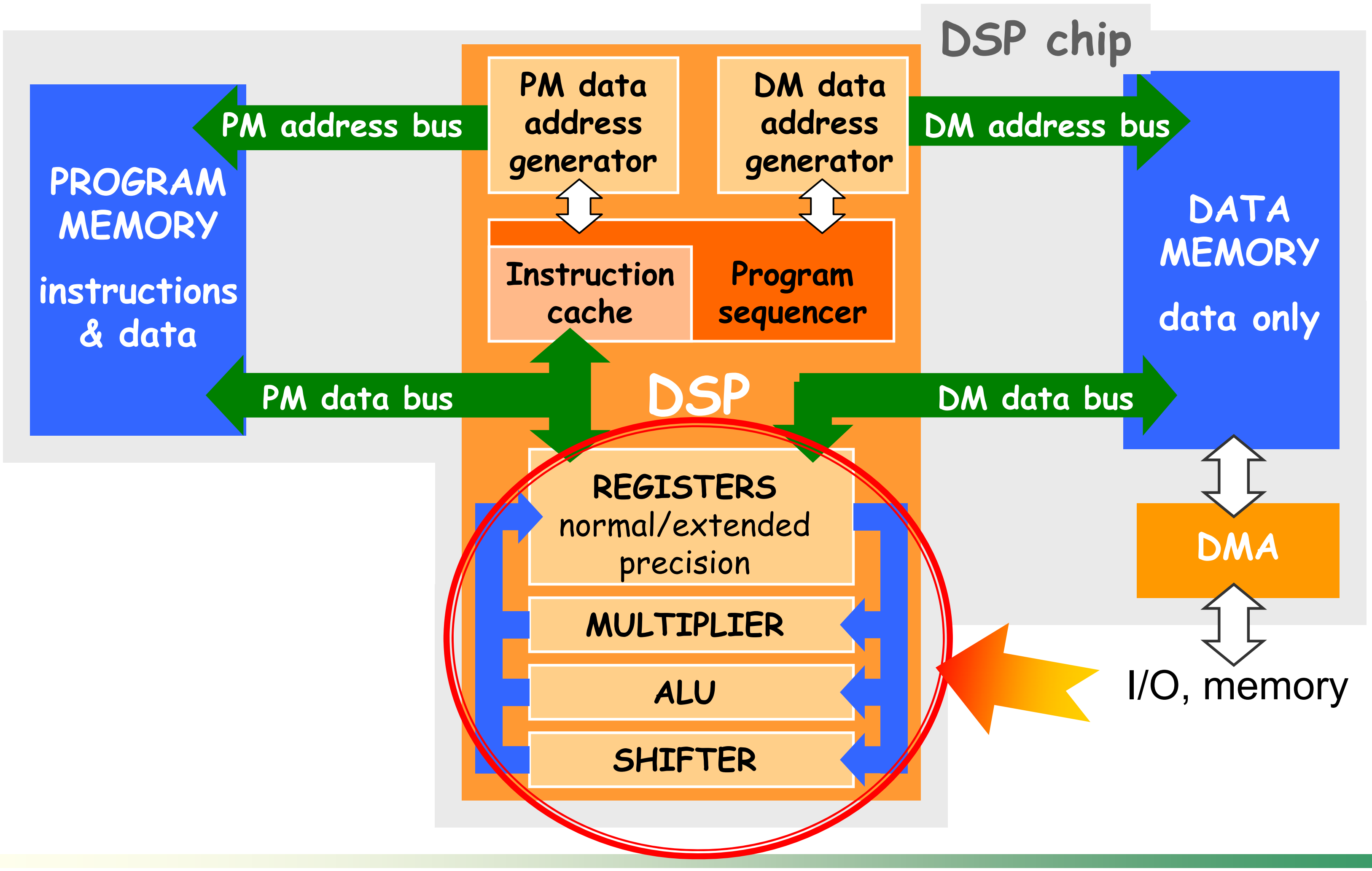
Shaped by **predictable real-time DSPing** !

ex: FIR $y(n) = \sum_{k=0}^{M} a_k \cdot x(n-k)$

Requirements	How
3.2 Fast data access	High-BW memory architectures. Specialised addressing modes. Direct Memory Access (DMA).
3.3 Fast computation	MAC-centred. Pipelining. Parallel architectures (VLIW, SIMD).
3.4 Numerical fidelity	Wide accumulator regs, guard bits ..
3.5 Fast-execution control	H/w assisted zero-overhead loops, shadow registers …

3.3 Fast computation

a) MAC-centered



3.3 Fast computation

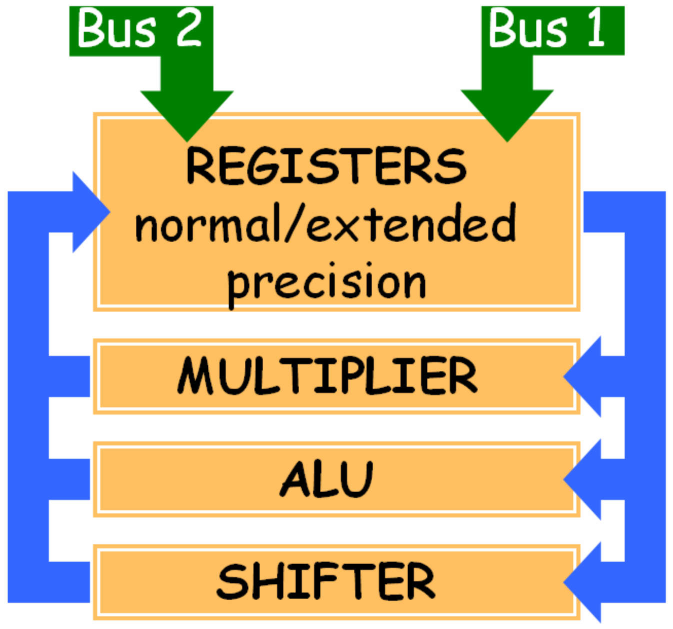
a) MAC-centered cont'd

Registers: intermediate & final results, counters. Shadow registers in some DSPs (ex: ADI SHARC) for fast context switch.

Multiplier: one instruction cycle multiplication (& accumulation).

Arithmetic Logic Unit (ALU): one instruction cycle ops. (basic arithmetic & logical).

Shifter (simple / barrel): shifts (scaling) & rotates.



3.3 Fast computation

b) Pipelining

Pipelining

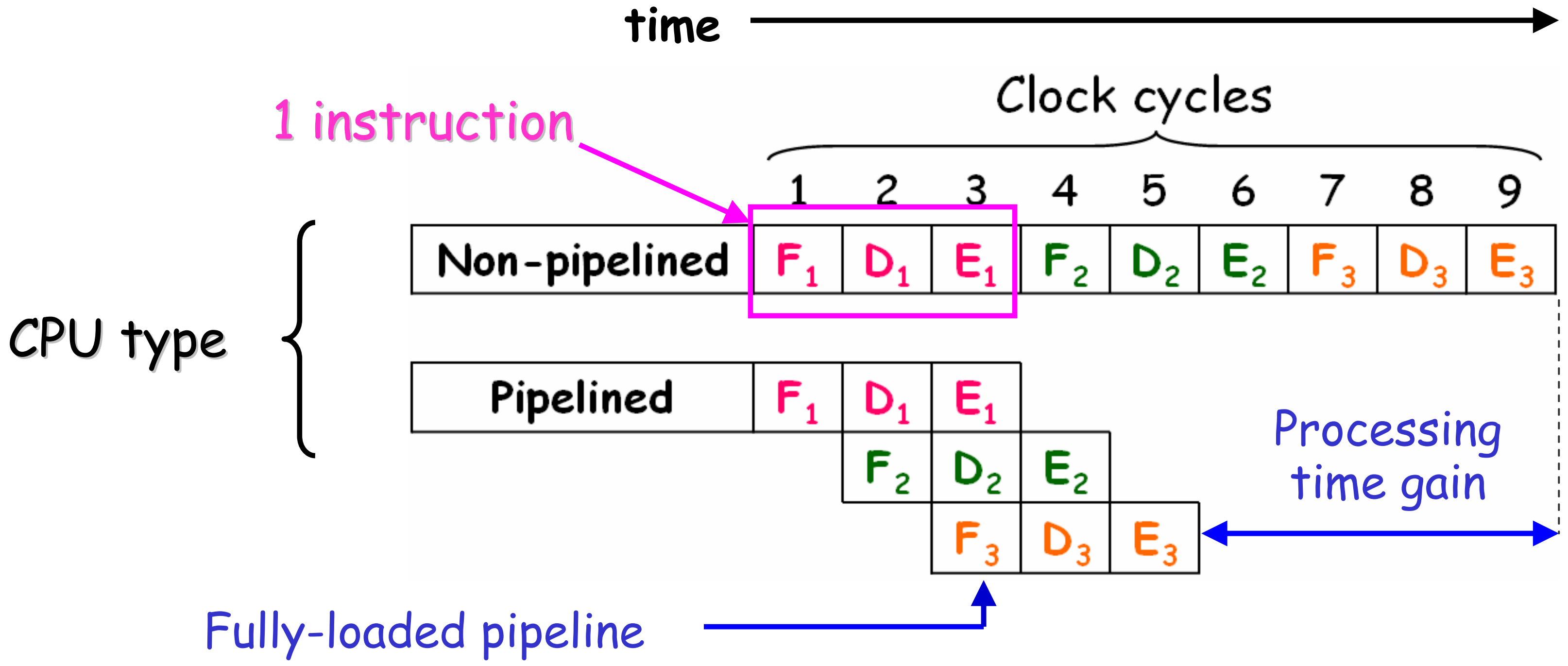
- instruction execution divided into stages.
- execution of stages for different instructions overlapped in time.

Basic pipeline stages

Fetch	▪ Generate program fetch address ▪ Read opcode
Decode	▪ Route opcode to functional unit ▪ Decode instruction
Execute	▪ Execute instruction

3.3 Fast computation

b) Pipelining – cont'd



- Pipeline full → fast instruction fetch → L1 & L2 cache.

3.3 Fast computation

b) Pipelining – cont'd

- Processors may add more sub-stages.
  - Smaller steps → faster processor clock speed

C6713 pipelining $\left\{ \begin{array}{l} \text{4 fetch stages. Fetch-packet = 8 instructions.} \\ \text{2 decode stages.} \\ \text{Up to 10 execution stages.} \end{array} \right.$

Pipelining limitations:

- Efficient *BUT* complex to program → compiler/scheduler effort .
- **Branch effects**: flow change (*branch/interrupt*) → pipeline flush
- **Resource conflicts**: two or more instructions need same h/w.
- **Data hazards**: instruction needs result of previous instruction.

→ *difficult worst case scenario prediction.*

3.3 Fast computation

c) Parallel architectures

Increased parallelism improves performance.

Very Long Instruction Words (VLIW):

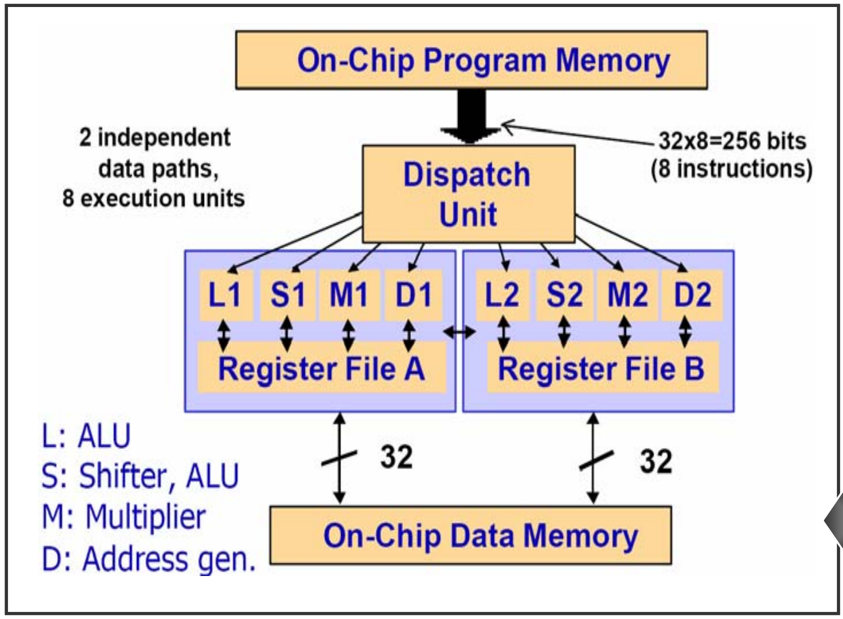
- **Instruction-level parallelism (ILP)**: multiple execution units, each executes its own instruction.
- Innovative architecture – first used in 'C62xx (1997) VelociTI.
- "***Multi issue***" DSPs: many instructions issued @same time.

Single Input Multiple Data (SIMD):

- **Data-level parallelism (DLP)**: one instruction performs same operation on multiple data sets.
- Technique used within other architectures (ex: VLIW).
- "***Single-issue***": one instruction issued @same time.

3.3 Fast computation

c) Parallel architectures: VLIW



- Simple & regular instructions set.
- Wide "global" instructions.
- **Deterministic** behaviour: scheduling @compile-time (i.e. static) *NOT* @processing-time.

TI 'C6000 VLIW architecture

Plus

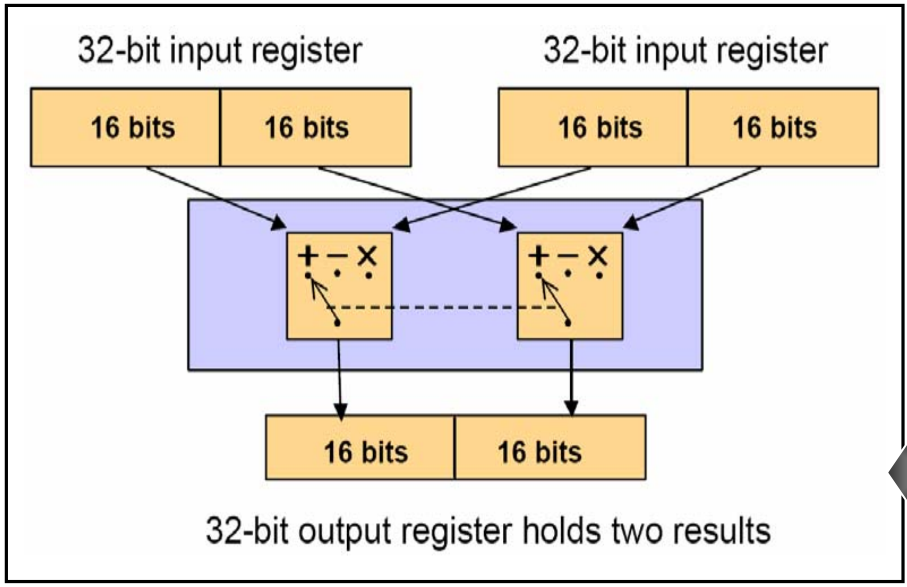
- Increased performance for many algorithms
- Scalable.
- Good compiler target.

Minus

- Deep pipelines & long latencies: peak performance elusive.
- High memory use/power consumption.
- Assembly: complex to hand-optimise.

3.3 Fast computation

c) Parallel architectures: SIMD



- Each instruction performs *lotsa* work.
- May support multiple data width.
- SIMD can be ON/OFF

SIMD typical architecture: SHARC "Hammerhead"

Plus

- Effective on large data blocks.
- Applicable to other architectures, ex: TigerSHARC (VLIW + SIMD).

Minus

- Parallel algorithms only → reorganisation penalties.
- High program-memory usage.

Shaped by **predictable real-time DSPing** !

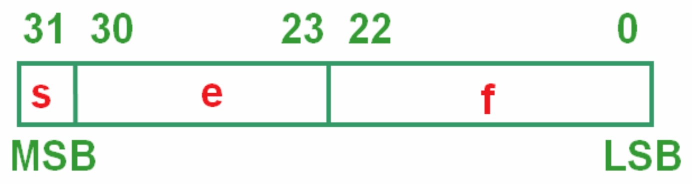
ex: FIR $y(n) = \sum_{k=0}^{M} a_k \cdot x(n-k)$

Requirements	**How**
3.2 Fast data access	High-BW memory architectures. Specialised addressing modes. Direct Memory Access (DMA).
3.3 Fast computation	MAC-centred. Pipelining. Parallel architectures (VLIW, SIMD).
3.4 Numerical fidelity	Wide accumulator regs, guard bits ..
3.5 Fast-execution control	H/w assisted zero-overhead loops, shadow registers …

3.4 Numerical fidelity

- Wide accumulators/registers for precision & overflow avoidance: *guard bits*.
- Overflow/underflow flags.
- Saturated arithmetic when overflowing.
- **Floating point arithmetic**: high dynamic range/precision.

IEEE 754 standard (normalised single precision)



e = exponent, offset binary, $-126 < e < 127$

s = sign, 0 = pos, 1 = neg

f = fractional part, sign-magnitude + hidden bit

$$\boxed{\text{Coded number } x = (-1)^{s} \cdot 2^{e} \cdot 1.f}$$

$$\boxed{\text{Dynamic range}_{\text{dB}}} = 20 \log_{10}\left[\frac{\text{largest value}}{\text{smallest value}}\right]$$

Fixed point ~ 180 dB

Floating point ~1500 dB

32 bits

Shaped by **predictable real-time DSPing** !

ex: FIR $y(n)=\sum_{k=0}^{M} a_k \cdot x(n-k)$

Requirements	How
3.2 Fast data access	High-BW memory architectures. Specialised addressing modes. Direct Memory Access (DMA).
3.3 Fast computation	MAC-centred. Pipelining. Parallel architectures (VLIW, SIMD).
3.4 Numerical fidelity	Wide accumulator regs, guard bits ..
3.5 Fast-execution control	H/w assisted zero-overhead loops, shadow registers …

3.5 Fast-execution control

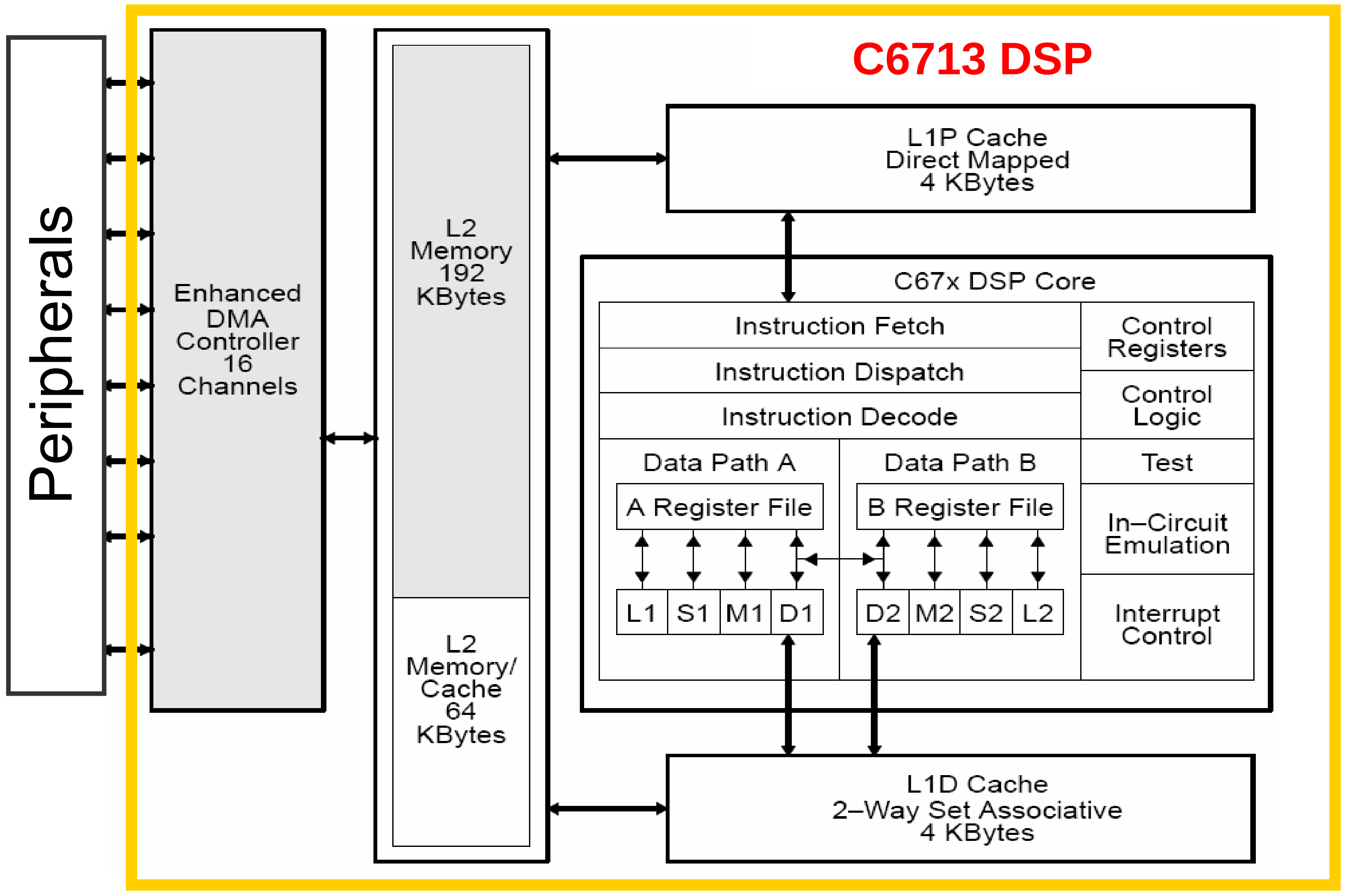
- Zero-overhead looping via specialised instructions (ex: RPTB).
- Fast/deterministic interrupt servicing. Important as DSP systems often interrupt-driven.
  - ***Interrupts*** : internal/external (DSP pins).
  - ***Latency*** = delay [interrupt pin active → first ISR instruction executed].
  - DSP stops current activity (if higher priority interrupt) & starts ISR.
  - DSP must save info related to previous activity (***context***).
  - Often many user-selectable interrupt dispatchers (→ saved context).

Example: SHARC (ADSP21160M, 80 MHz, 12.5 ns cycle)

ISR uses secondary register set → Super-fast

Interrupt dispatcher	Cycles before ISR	Cycles after ISR
Normal	183	109
Fast	40	26
Super-fast	34	10
Final	24	15

3.6 DSP core example: TI 'C6713



Yellow box: C6713 DSP core. **White boxes**: parts common to all C6000 devices. **Grey boxes**: additional features on the C6713 DSP.

Chapter 3 summary

- DSP core characteristics shaped by predictable real-time DSPing.
- Fast data access
  - High-BW memory architecture
  - Specialised addressing modes
  - Direct Memory Access (DMA)
- Fast computation
  - MAC-centred
  - Pipelining
  - Parallel architectures (VLIW, SIMD)
- Numerical fidelity
  - wide accumulator registers, guard bits…
- Fast-execution control
  - h/w assisted zero-overhead loops, shadow registers…

Chapter 4 topics

DSP peripherals

4.1 Introduction	Today
4.2 Interconnect & I/O	Today
4.3 Services	Today
4.4 C6713 example	Today
4.5 Memory interfacing	Tomorrow
4.6 Data converter interfacing	Tomorrow
4.7 DSP Booting	Tomorrow
Summary	Tomorrow

4.1 Introduction

- Available peripherals: important factor for DSP choice.
  - Interconnect & I/O. Potential bottleneck!
  - Services: timers, PLL, power management, booting logic.
- Embedded peripherals:

☺ Fast performance	☹ Less flexible across applications
☺ Low power consumption	☹ Unit cost may be higher

- Terrific evolution
  - Few parallel & serial ports initially.
  - Now support for audio/video streaming applications.
- Often not enough pins on DSP chip → multiplexed! Select desired peripherals @ DSP boot **[→ C6713]**.

4.2 Interconnect & I/O

Some connectivity:

- General Purpose I/O (GPIO)
- Host Processor Interface (HPI)
- PCI

Some serial interfaces:

- Serial Peripheral Interface (SPI)
- Universal Asynchronous Receiver-Transmitter (UART)
- Controller Area Network (CAN)
- Multichannel Buffered Serial Ports (McBSP)
- Multichannel Audio Serial Port (McASP)

} Many transmit/receive data sizes

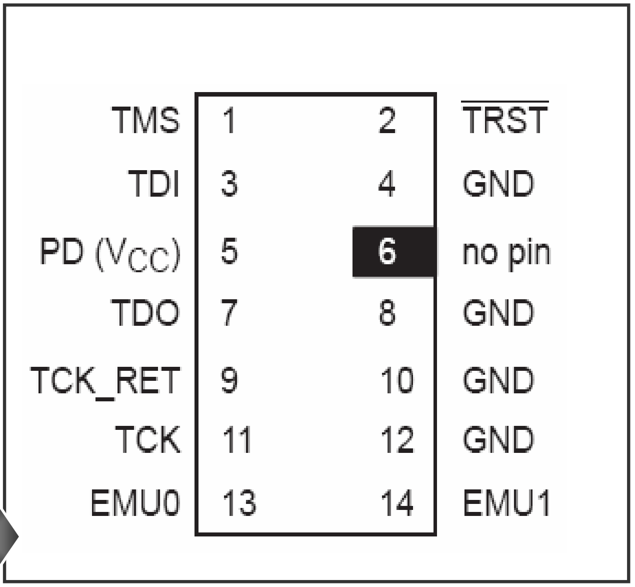
Some parallel interfaces:

Linkports [ADI]: DSP-DSP or DSP-peripheral communication

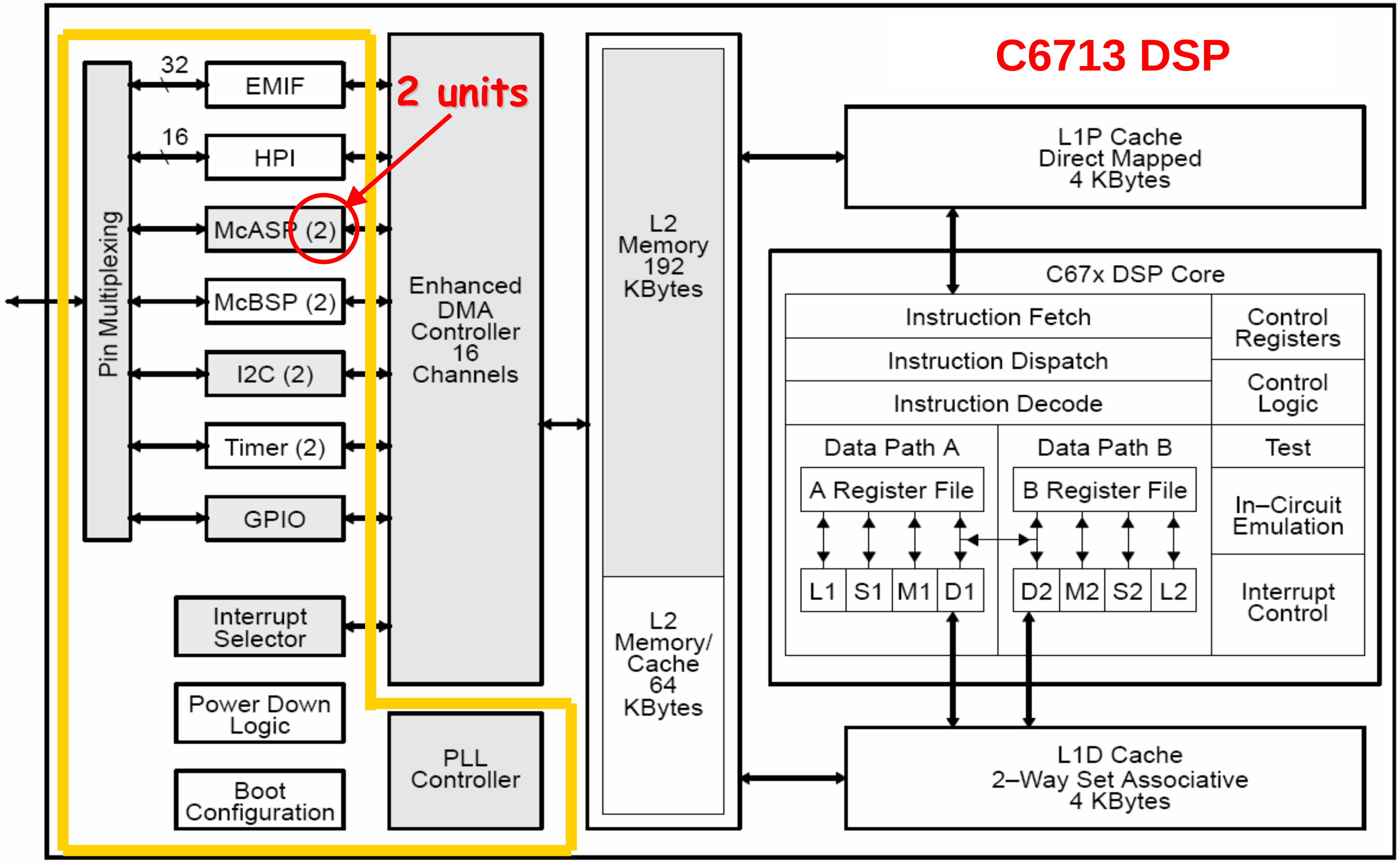
4.3 Services

- Timers: pulses / interrupts generation.
- Power management: reduces clock to reduce power consumption.
- DSP booting mode [→next lecture] & configuration logic
- PLL controller: generates clock for DSP core & peripherals from internal/external clock.
- Interrupt selector: Selects interrupts to send to the CPU. Can also change their polarity.
- JTAG: allows emulation & debug [→ **chapter 7**].

JTAG 14-pin header



4.5 Example: C6713 DSP



Yellow box: C6713 DSP peripherals. **White boxes**: parts common to all C6000 devices. **Grey boxes**: additional features on the C6713 DSP.

Chapter 4 (partial) summary

- Peripherals: wide range & important parameters for DSP choice.
- Interconnect & data I/O: serial + parallel interfaces.
- Services: PLL, timers, JTAG, power management…

TO BE CONTINUED TOMORROW