Signal Processors Lecture Notes, Pt 2

Institut für Technische Informatik, DI Dr. Eugen Brenner

Signal Processors, 448.032

03.04.2018

Chapter Overview

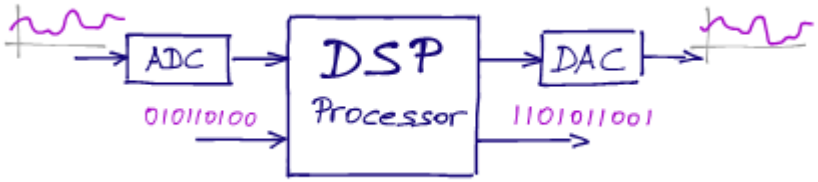
2. Fundamentals of DSP

- Motivation
- Examples of DSP Applications
- Characteristics of DSP Algorithms
- HW Features of DSPs
- Components of DSP Architectures
- Characteristics of DSP Architectures
- Selected DSP Implementations
- Number Formats
- DSPs vs. General Purpose CPUs
- Conclusion

Motivation

Structure of Digital Systems

- Processing chain
  - Transformation and re-transformation in digital representation



- Mathematical description of signals/systems
  - Continuous domain:
    Fourier- and Laplace transform
  - Discrete domain:
    discrete Fourier and Z-transform

Analog vs. Digital Processing

- Reproducibility
  - Digital: systems with equal functionality are simple to produce (copy SW)
  - Analog: device tolerances, calibration causes higher costs
- Long-term stability
  - Digital: inherent
  - Analog: temperature drift, deterioration of units, higher costs due to compensation

Analog vs. Digital Processing

- Re-programmability
  - Digital: modification of functions occurs by modification of SW. Adaptive systems are possible
  - Analog: modifications of function requires modification of HW. Adaptive systems are more difficult to realize
- Data transfer and storage
  - High SNR at digital functions: great SNR, small bit error rate
  - Digital transmission and storage: error detection and error recovery possible
    (encoding)

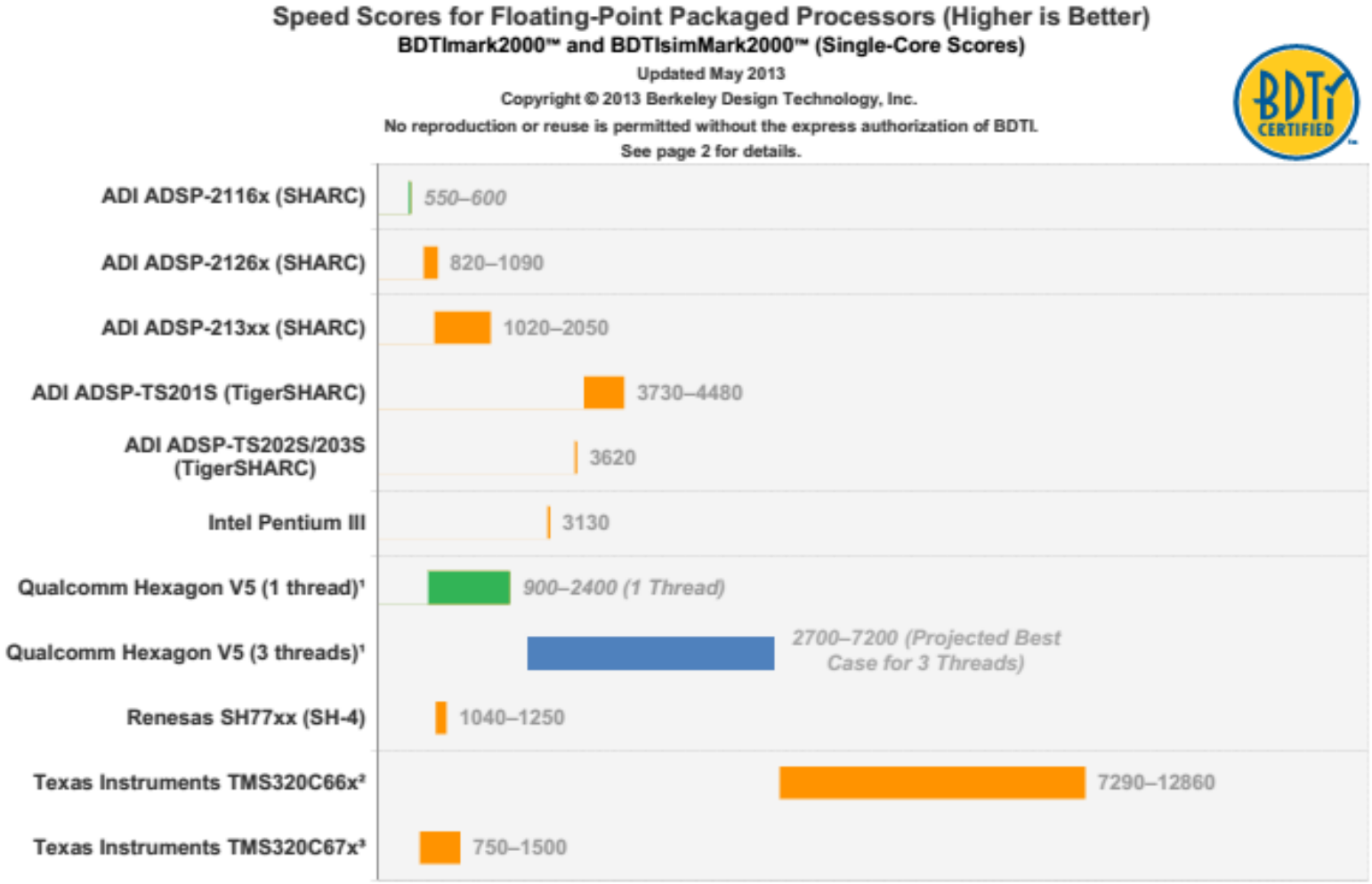
Analog vs. Digital Processing

- Data Compression
  - For economic use of transmission lines
  - Digital: loss-less as well as lossy compression possible (e.g.: DCT vs. Huffman coding in JPEG)
  - Analog: compression always lossy (e.g.: bandwidth limitations)
- Analog unrealizable functions
  - Filters with linear phase response
  - Loss-less compression
  - FIR filter

Limitations of Digital Processing

- Computing power
  - Available computing power limits functionality (and sampling rates, respectively)
  - But: technological advances steadily pushes the boundaries
- Economic feasibility
  - Potentially higher HW-costs than with analogue system
  - But: with higher integration and mass production more applications are feasible

Advantage through Specialization



Examples of DSP Applications

Wide Range of Applications

Examples of DSP Applications

- Real-time data acquisition
  - Data pre-processing (filtering)
  - Instrumentation, spectral analyser
- High-speed control
  - Motor control, hard disks (reading head)
  - Robotics (sensors and actuators)

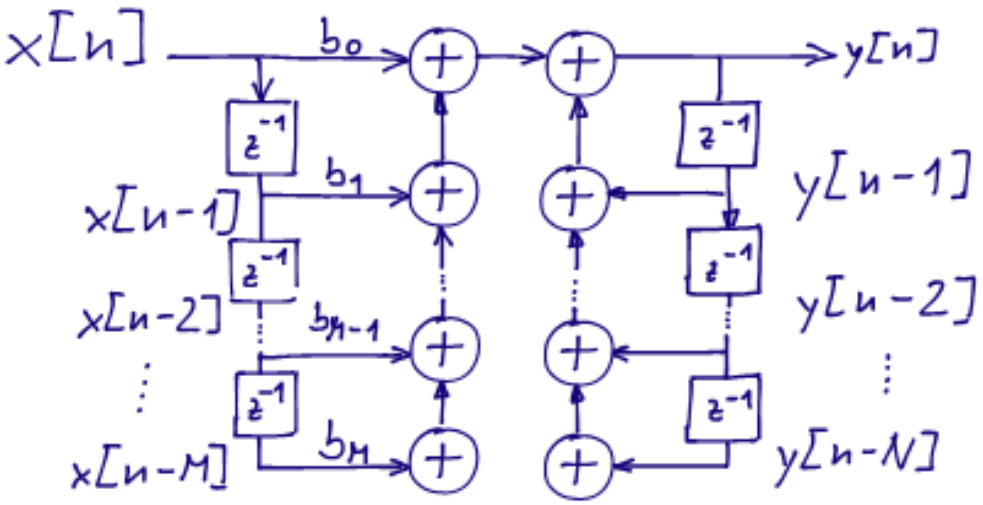
Examples of DSP Applications

- Digital audio
  - Analysis, compression, enhancements, watermarking, surround sound
- Communication
  - Modulation, de-modulation, coding, equalizer, error correction, noise cancellation
- Image processing
  - Image enhancement (filtering, noise suppression), video processing, digital TV

Characteristics of DSP Algorithms

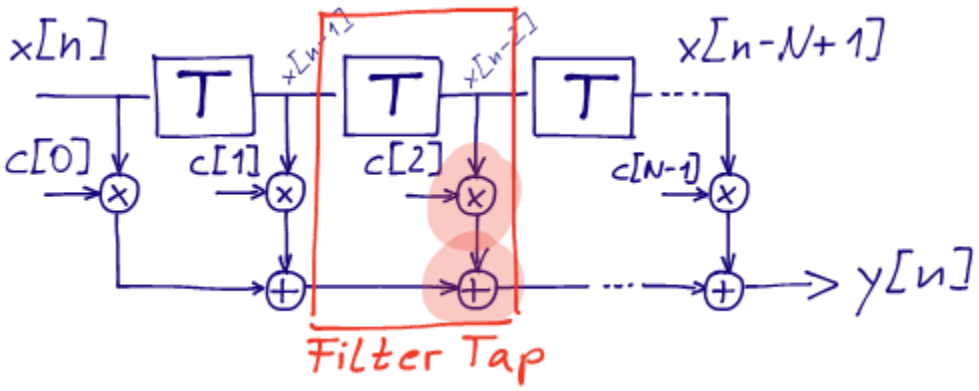
A Linear Time-Invariant Discrete-Time System

$$y[n] - \sum_{k=1}^{N} a_k\, y[n-k] = \sum_{k=0}^{M} b_k\, x[n-k]$$



$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}}$$

Example: Finite Impulse Response Filter



- Mathematical description:

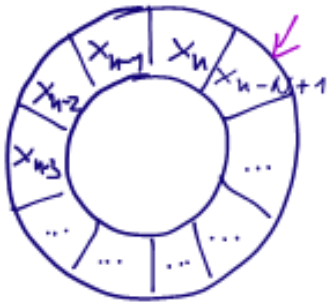
$$y_n = x_n \cdot c_0 + x_{n-1} \cdot c_1 + \ldots + x_{n-N+1} \cdot c_{N-1}$$

  - y_k output value at time k
  - x_k input value at time k
  - c_i filter coefficient

Example: Finite Impulse Response Filter

$$y_n = \sum_{i=0}^{N-1} x(n-i) * c_i$$

- Use a ringbuffer to store previous inputs



- Computation of output value: N taps
  - N x multiply
  - N - 1 x add
  - 2N operands

Characteristics of DSP Algorithms

- Process data from input-buffer => output-buffer

```
x[oldest] = input();
Y = 0;
for(k = 0; i < N; k++)
   y += h[k] * x[(oldest + k) % N];
oldest = (oldest + 1) % N
output(y)
```

  - => modulo Adressierung

Characteristics of DSP Algorithms

- Handling overflows
  - Ceiling
  - Convergent
  - Floor (truncate)
  - Nearest
- Shifting
  - Logic shift: fill with zeros
  - Arithmetic shift: consider sign bit
  - Rotate: wrap around shifted bit
  - Shift/rotate arbitrary number of bits

Characteristics of DSP Algorithms (in general)

- Characteristics
  - Many arithmetic operations (ADD, MULT)
  - Continuous operations on multidimensional fields
  - Few branches but with well-predictable targets
  - High concurrency
  - High data volume (throughput)
- DSP-Algorithms are commonly used in:
  - Embedded systems
  - Real-time systems
  - “streaming applications”

HW Features of DSPs

Characteristics of DSP Architectures

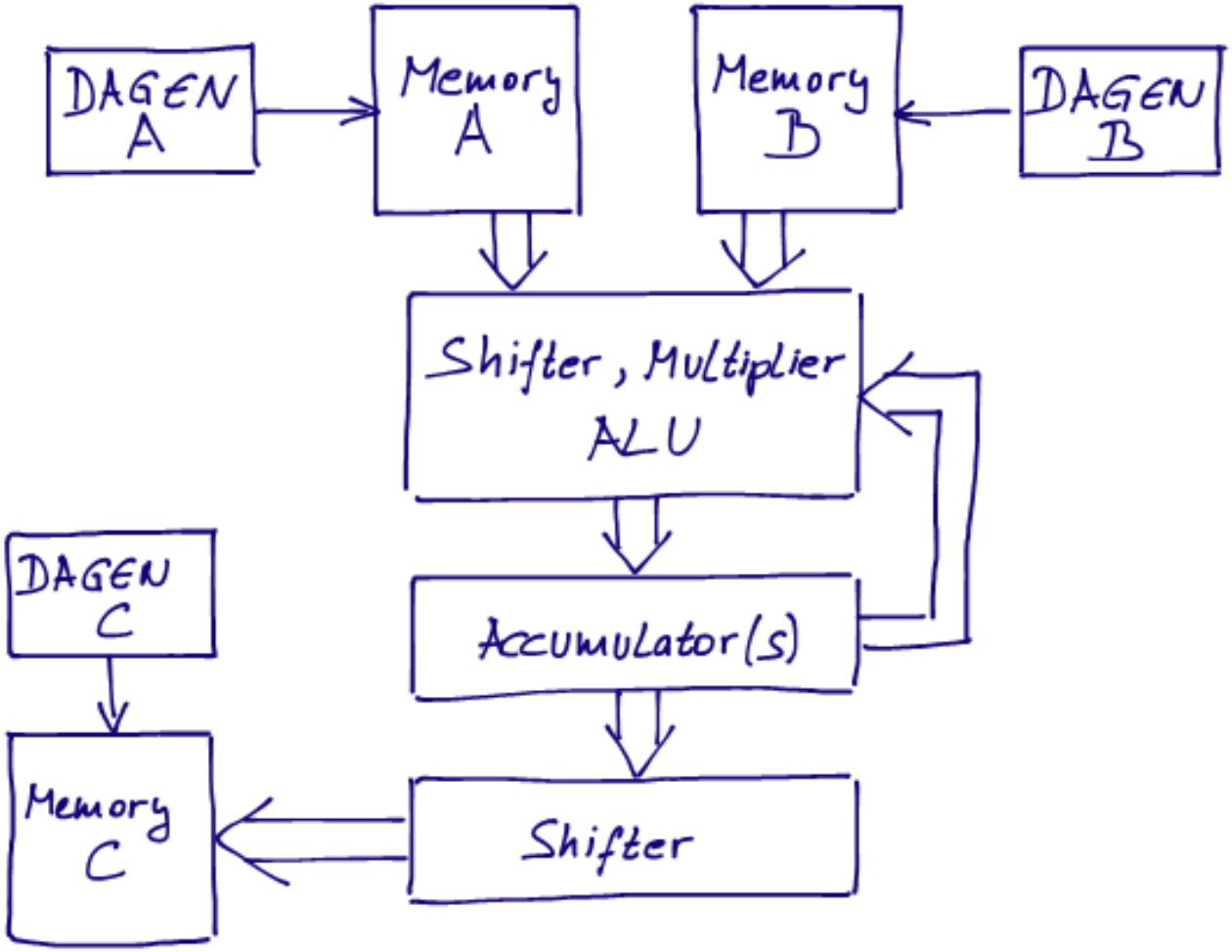
- Efficient execution of DSP algorithms
  - with special HW-components
- Example: FIR-Filter
  - Objective: compute one filter-tap in a single cycle

Characteristics of DSP Architectures

- Complexity of a single FIR-Filter Tap
  - Multiplication, addition
    => Multiply-Accumulate
  - Access to 3 operands
    => Multiple internal busses
  - Fetch instruction
    => Separate bus for instructions, Harvard architecture
  - Loop control
    => Special instructions, Pragmas
  - In-/output buffer management
    => Special addressing modes

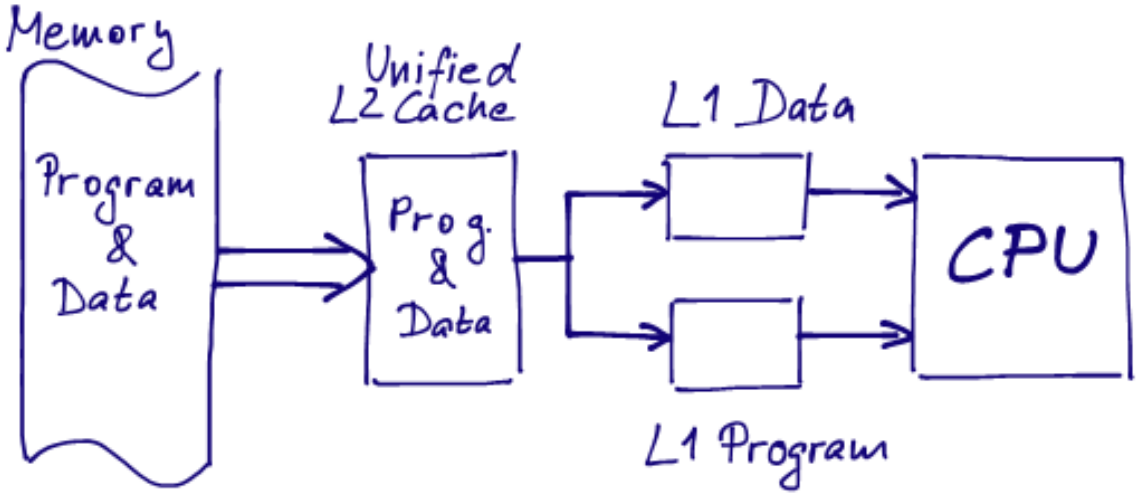
Components of DSP Architectures

Generic DSP-Architecture



CPU Architecture

- (Modified) Harvard architecture



- Access data and instructions concurrently

Components of DSP Architectures

- Multiply-Accumulate (MAC)
  - Multiplications are expensive
  - DSPs have a dedicated (parallel) HW multiplier
  - Multiplication and addition in a single cycle
  - Accumulator with additional Guard Bits
- ALU & Shifter
  - AND, OR, NOT, Shift
  - (Barrel-) Shifter for scaling
- Pipelined arithmetic processing

Components of DSP Architectures

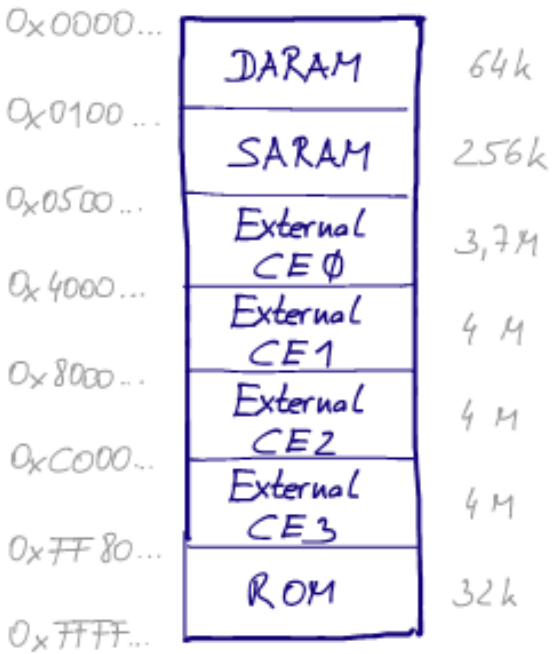
- Multiple buses
  - Several internal data/address-buses
    (e.g., loading two operands, simultaneous DMA and data-transfer)
- Multiple memories
  - Separate memories for parallel access
  - Dual-/multiport memory
  - Faster, internal (on-chip) memory
  - Slower, external memory (larger capacity)
  - Mapping of memory segments to different areas possible
    (e.g., stack, variables, code, . . .)

Components of DSP Architectures

- Data Address GENeration (DAGEN)
  - Support different addressing modes
  - Sequential addressing
  - Modulo addressing
  - Bit-reversed addressing
  - Increment/decrement of address pointer

Components of DSP Architectures

- Memory Space
  - Mapping of different memories into single address space
- Cache
  - L1 cache close to the processor, fast
  - L2 cache, larger, not so fast
  - Separate caches for program & data
  - Example: C55x has 64 bytes instruction queue, can be used as cache
  - Performance vs. complexity, determinism

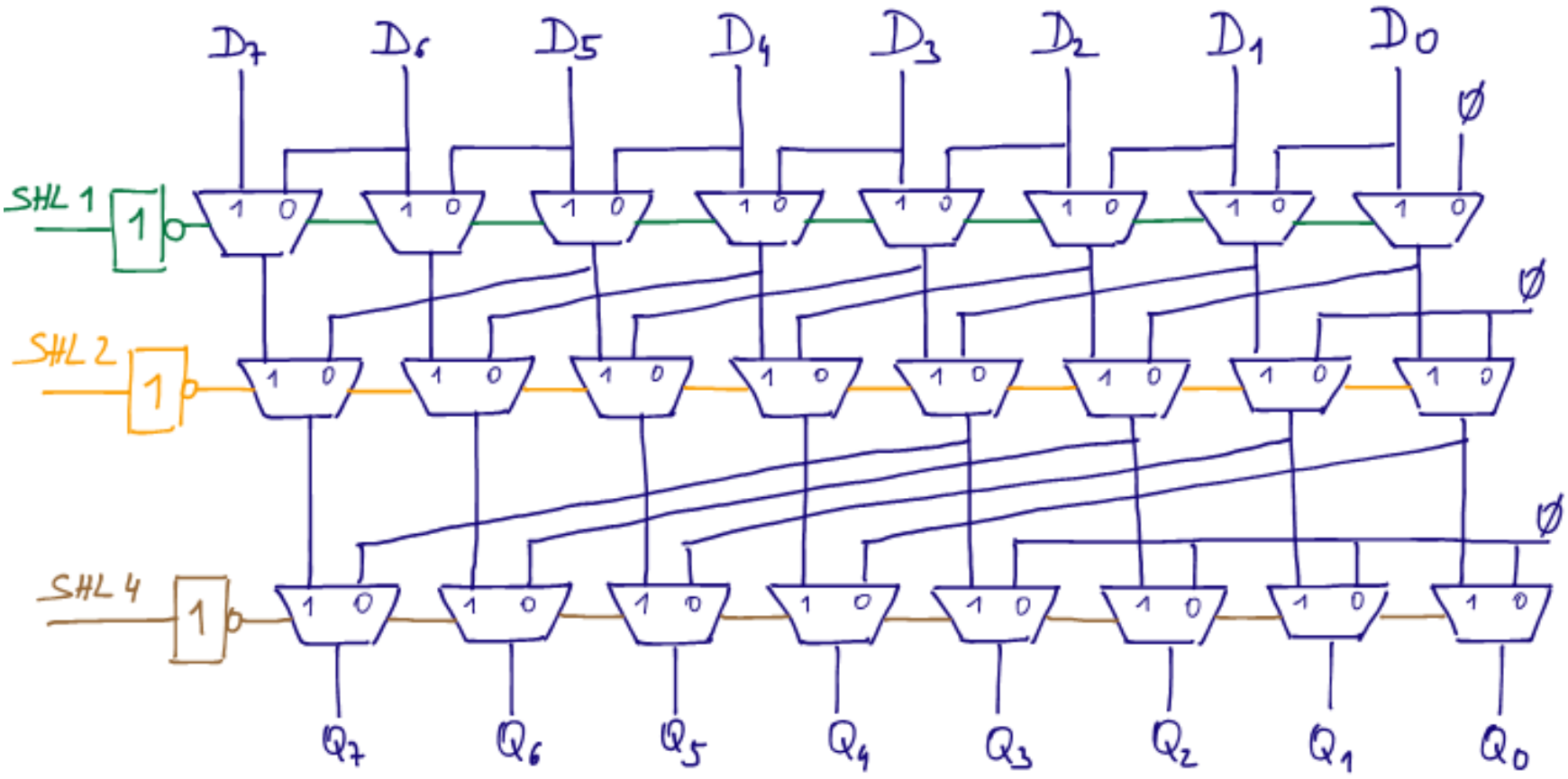


Components of DSP Architectures

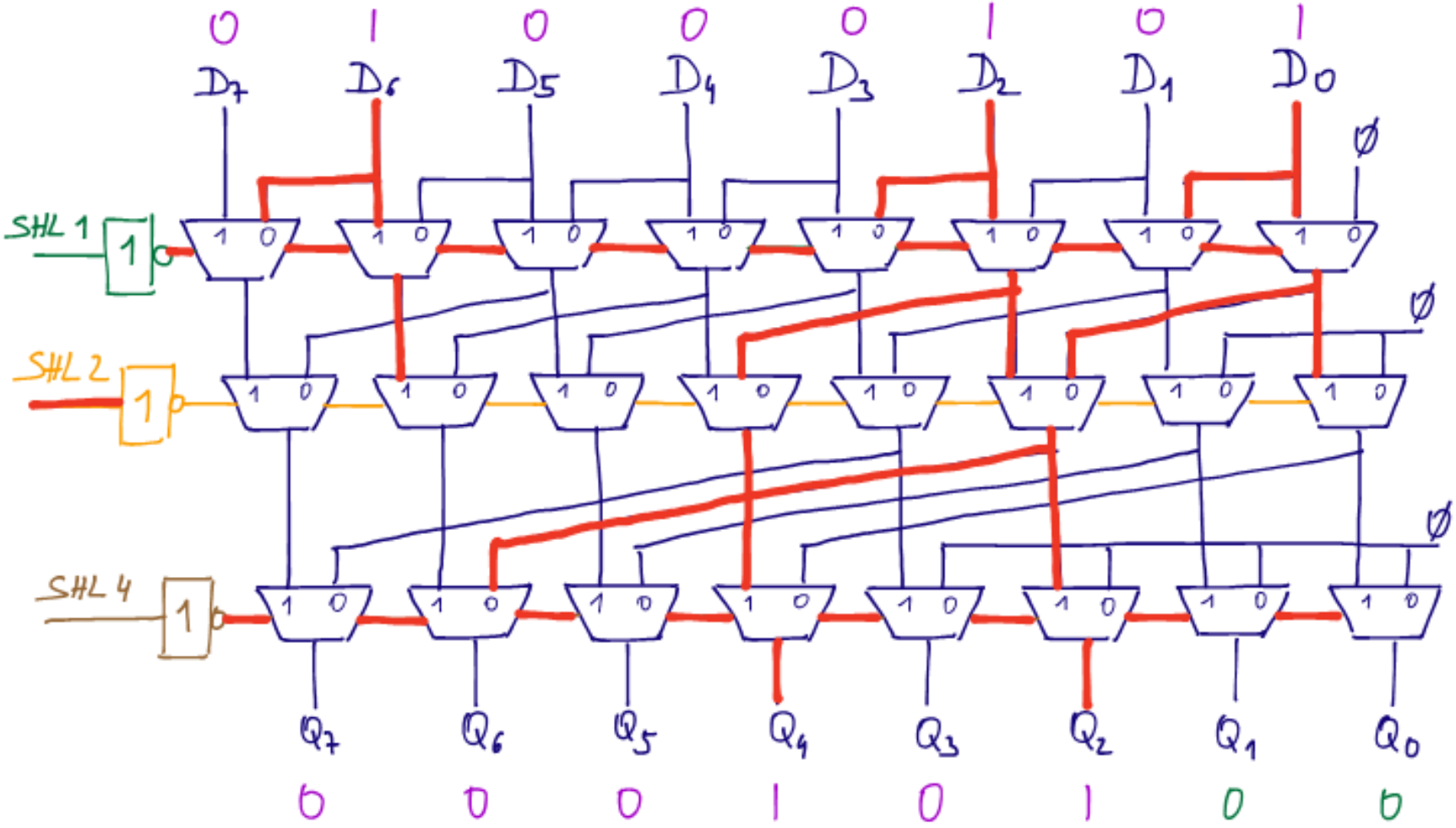
- Peripherals
  - Serial port
    - Standard serial port
    - Buffered serial port (BSP): high-speed full-duplex
    - Time-division multiplexing serial port: communication with multiple peers
    - Multi-channel BSP (McBSP), e.g. TMS320 processors
  - Host-Port Interface (HPI)
    - Interface to a host processor (e.g. microprocessor)
  - Parallel port
  - Hardware timer
  - Clock generation
  - . . .

Binary Shifting

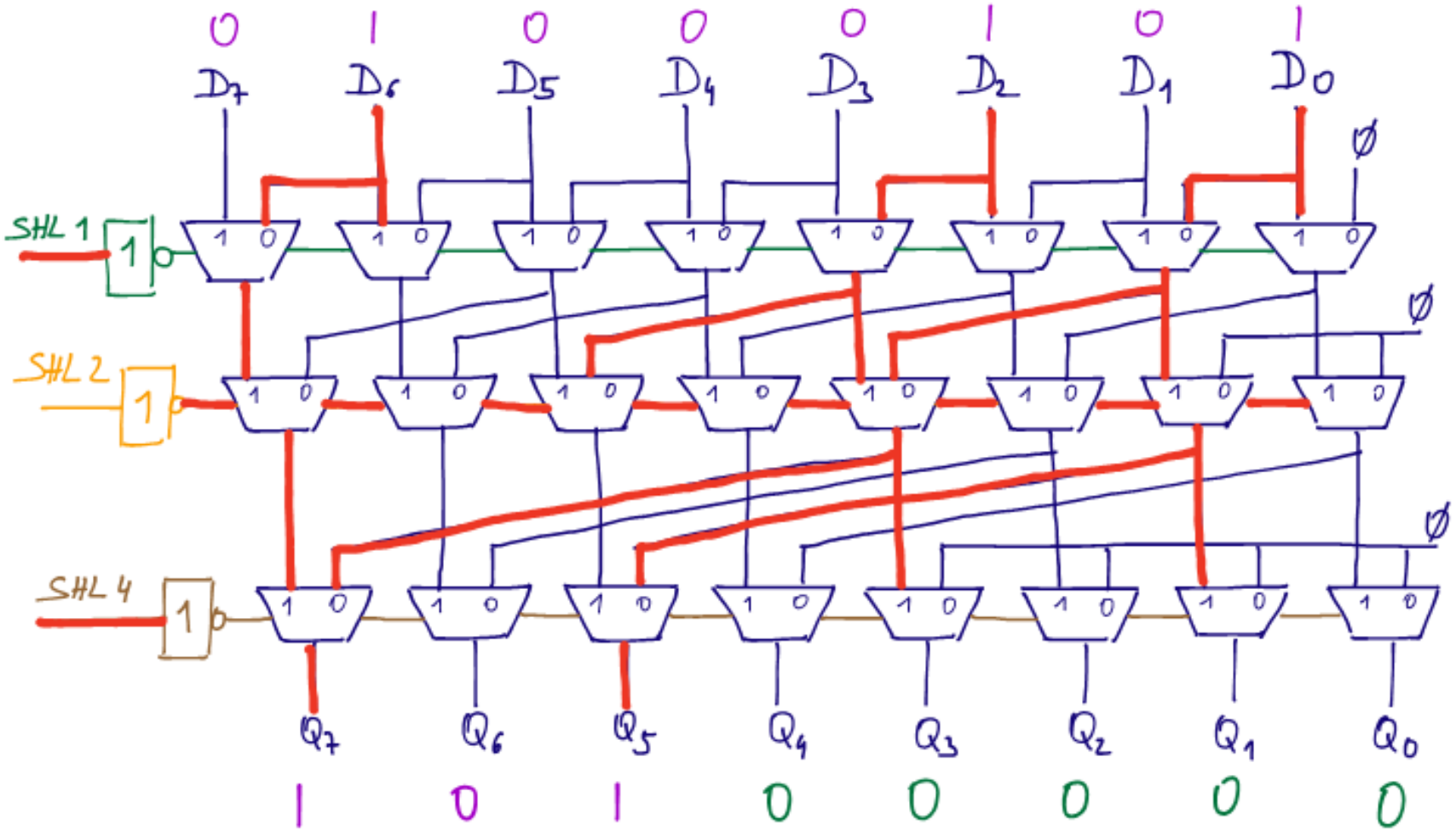
Barrel Shifter



Barrel Shifter

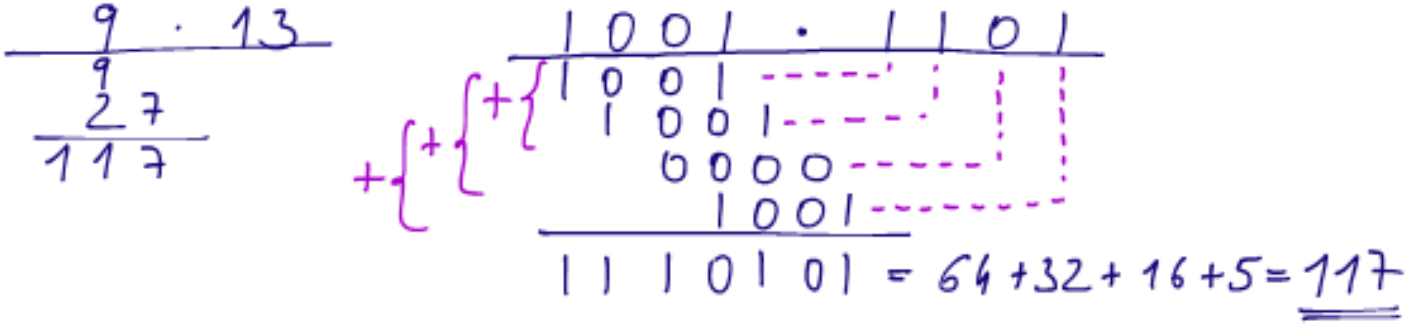


Barrel Shifter

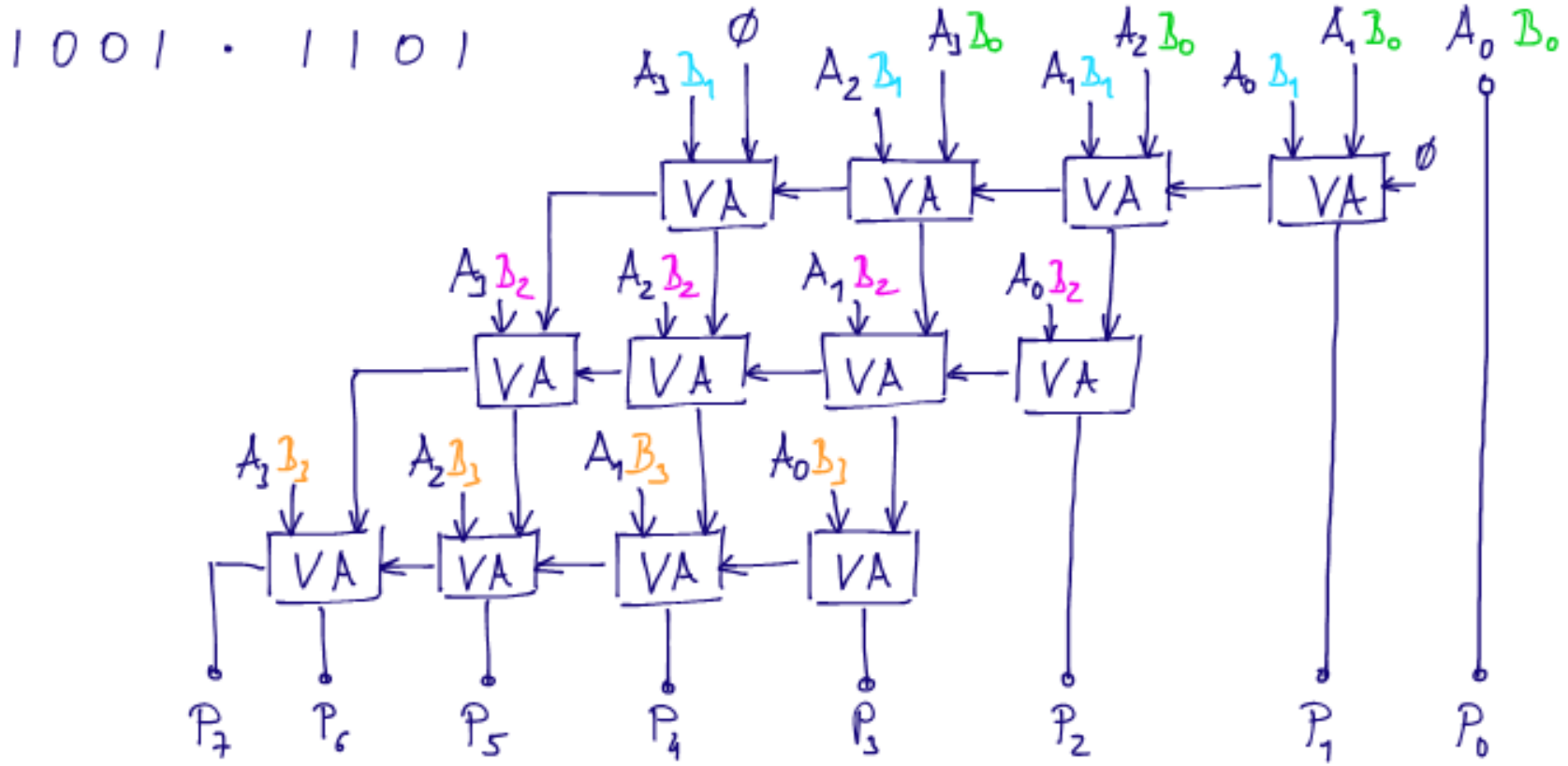


Binary Multiplication

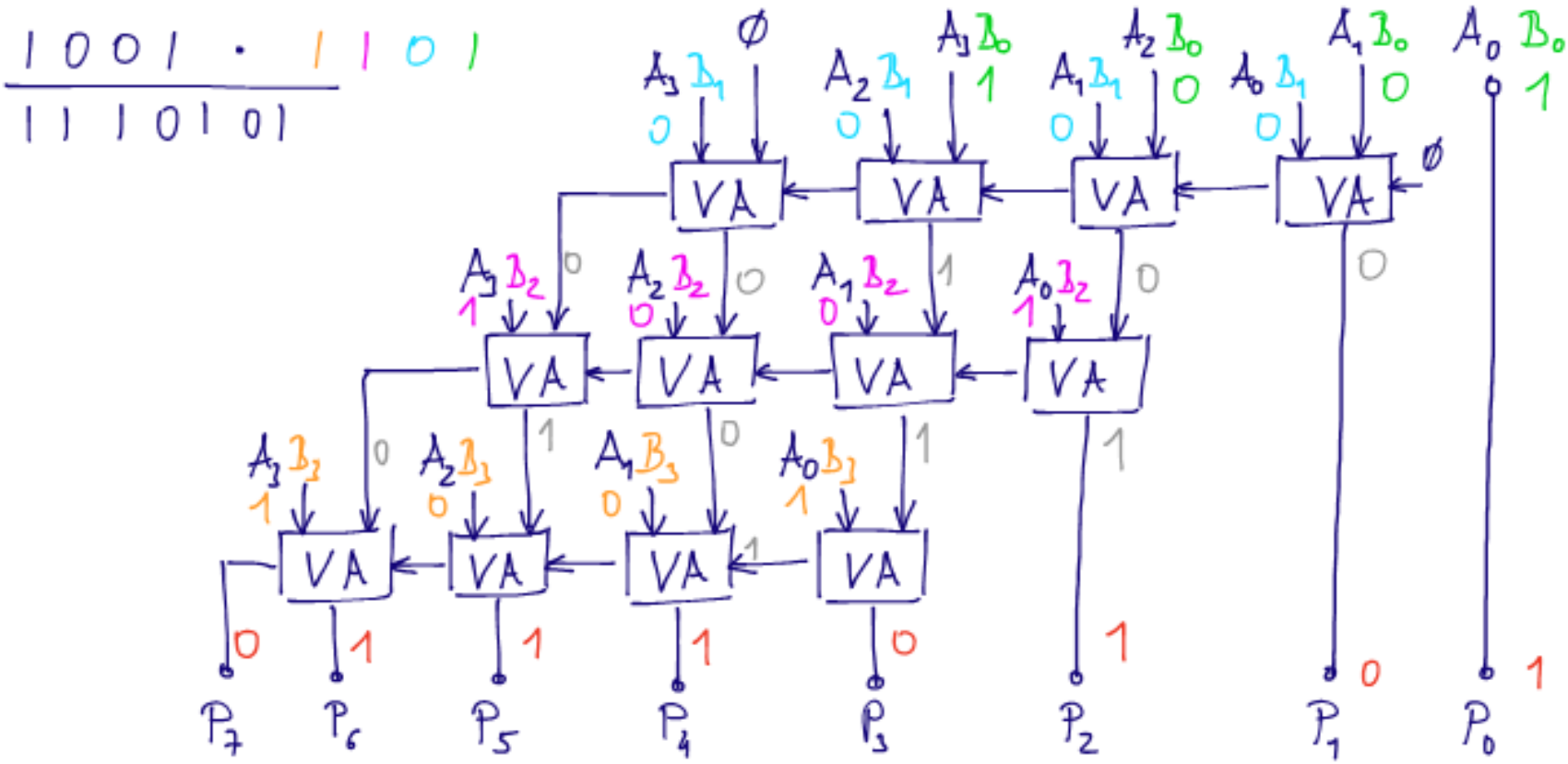
Multiplying Numbers



Parallel Multiplier



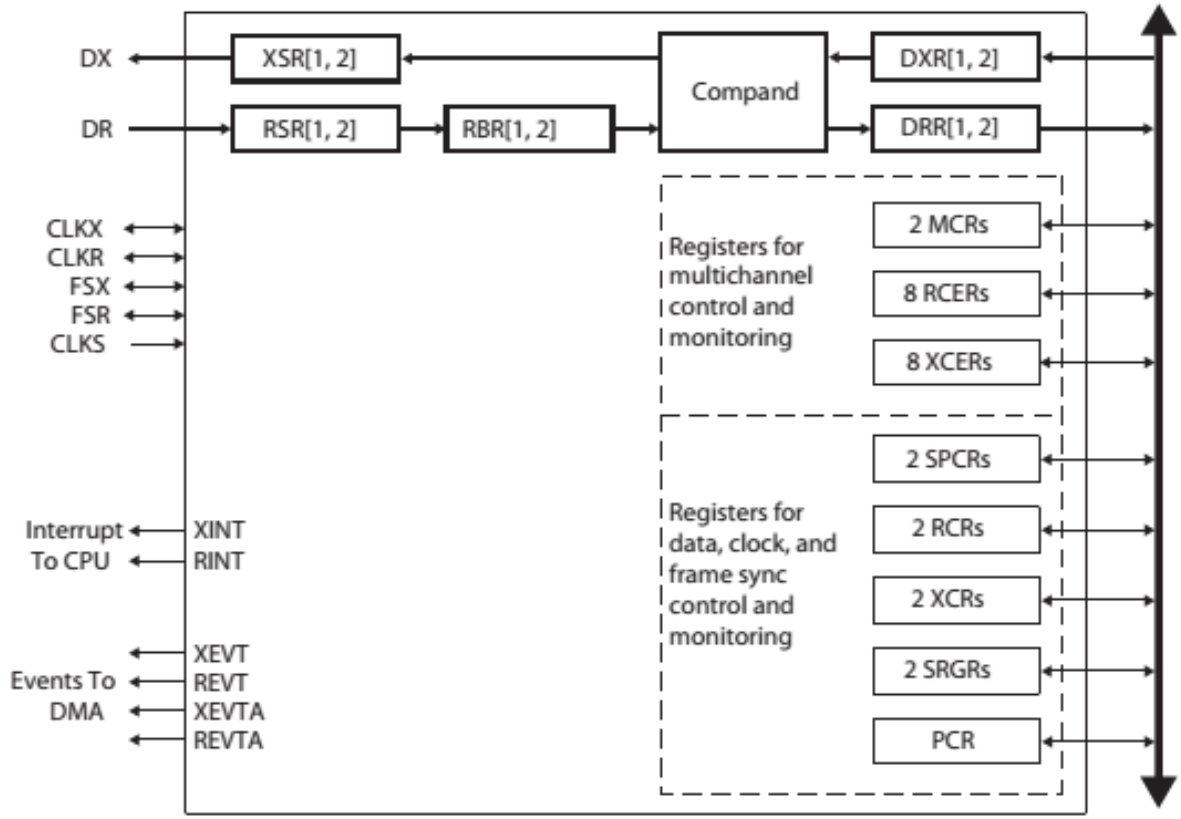
Parallel Multiplier



Peripherals

McBSP

- TMS320 McBSP

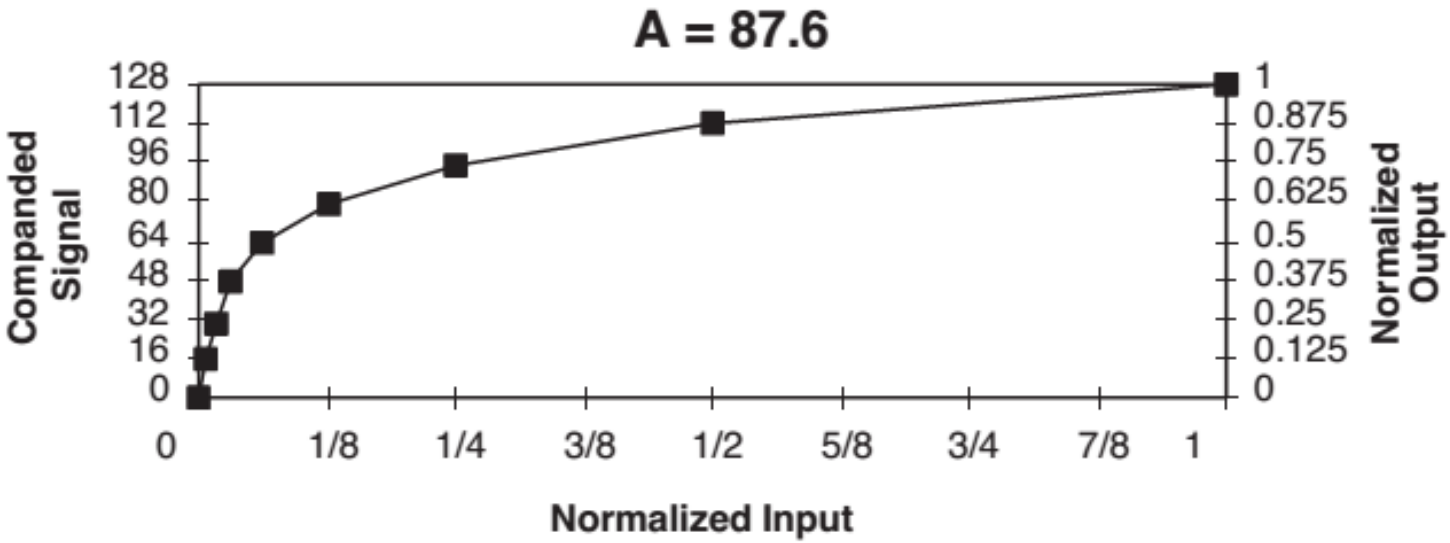


Companding

- Compression & expansion
- Used in many different applications
  - Roots in telephone systems
- Non-linear compression
  - Exploits human sense of hearing
  - μ-law: North America, North Asia
  - A-law: Europe, rest of the world

Companding: A-Law

- Logarithmic mapping



Companding: A-Law

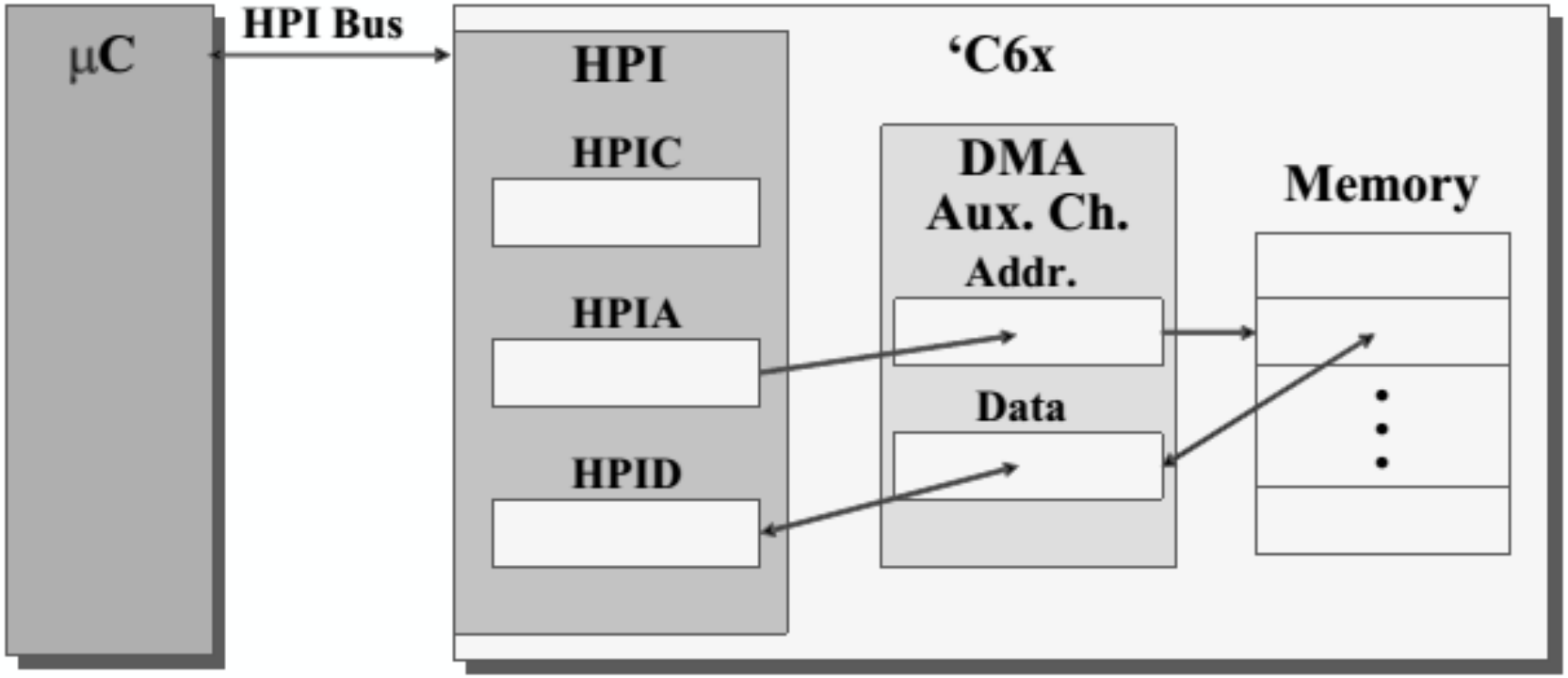
- Binary encoding table

Input Values													Compressed Code Word							
													Chord			Step				
Bit:	11	10	9	8	7	6	5	4	3	2	1	0	Bit:	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	a	b	c	d	x		0	0	0	a	b	c	d
	0	0	0	0	0	0	1	a	b	c	d	x		0	0	1	a	b	c	d
	0	0	0	0	0	1	a	b	c	d	x	x		0	1	0	a	b	c	d
	0	0	0	0	1	a	b	c	d	x	x	x		0	1	1	a	b	c	d
	0	0	0	1	a	b	c	d	x	x	x	x		1	0	0	a	b	c	d
	0	0	1	a	b	c	d	x	x	x	x	x		1	0	1	a	b	c	d
	0	1	a	b	c	d	x	x	x	x	x	x		1	1	0	a	b	c	d
	1	a	b	c	d	x	x	x	x	x	x	x		1	1	1	a	b	c	d

Host Port Interface

- Allows host processor to access internal DARAM, SARAM and portions of external memory
  - 16 bit parallel access
  - Connect peripherals (shared memory), PCI bridge
- Directly connected to the DMA controller

Host Port Interface



Enhanced Direct Memory Access

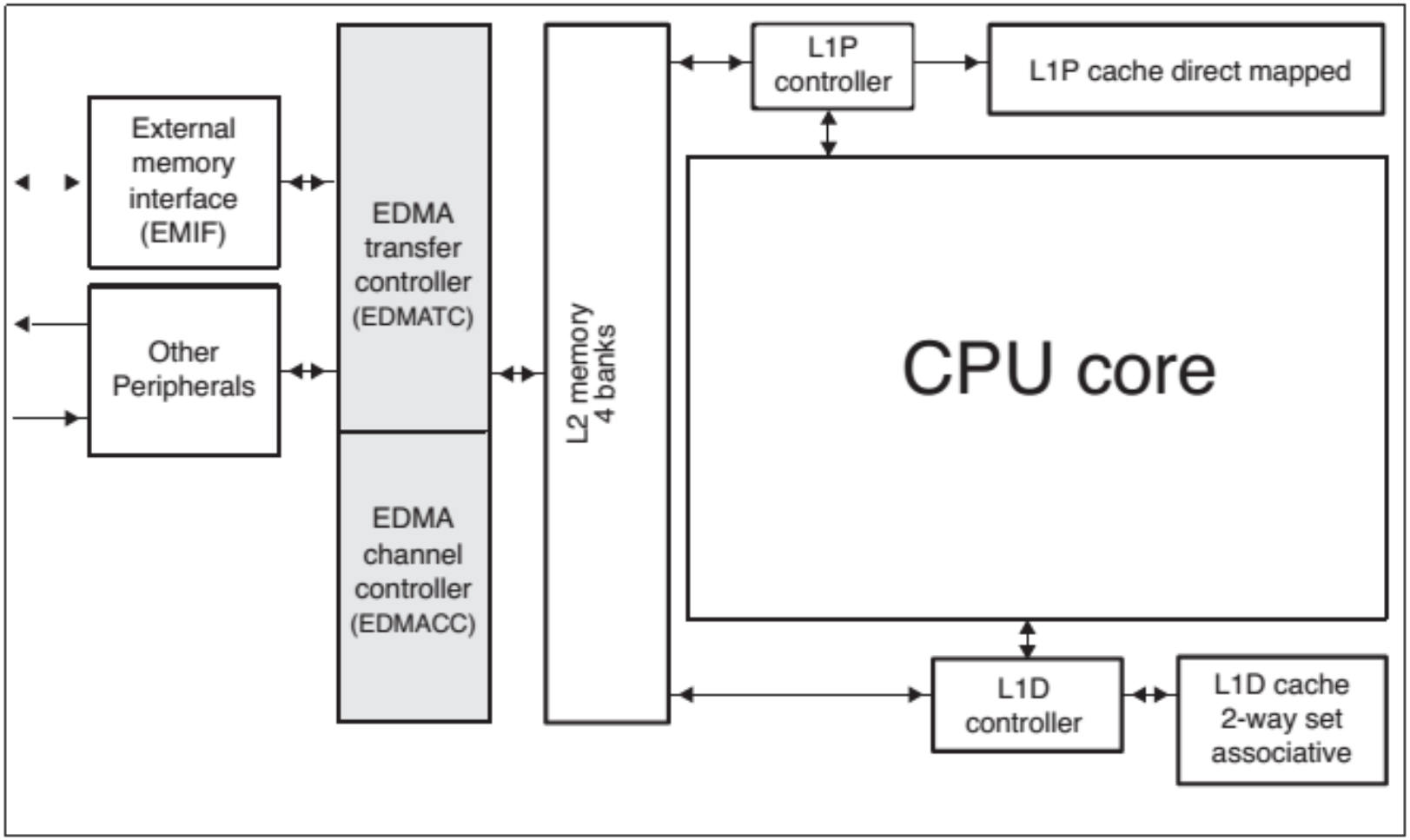
Enhanced Direct Memory Access

- DMA allows data transfer between L2 cache and device peripherals
  - CPU only configures DMA transfer
  - DMA transfer is independent, CPU can do other things
- Transfer rate: *1 element per cycle*
  (unless there is a conflict)
- *Multiple DMA channels* available
  (e.g., 16 on the TMS320C67x)

EDMA Components

- EDMA Transfer Controller (EDMATC)
  - Handles all data transfers
- EDMA Channel Controller (EDMACC)
  - User-programmable part of EDMA
  - Transfer mode (1D/2D)
  - Addressing mode
  - Trigger
  - Priority
  - . . .

EDMA Block Diagram



EDMA Triggers

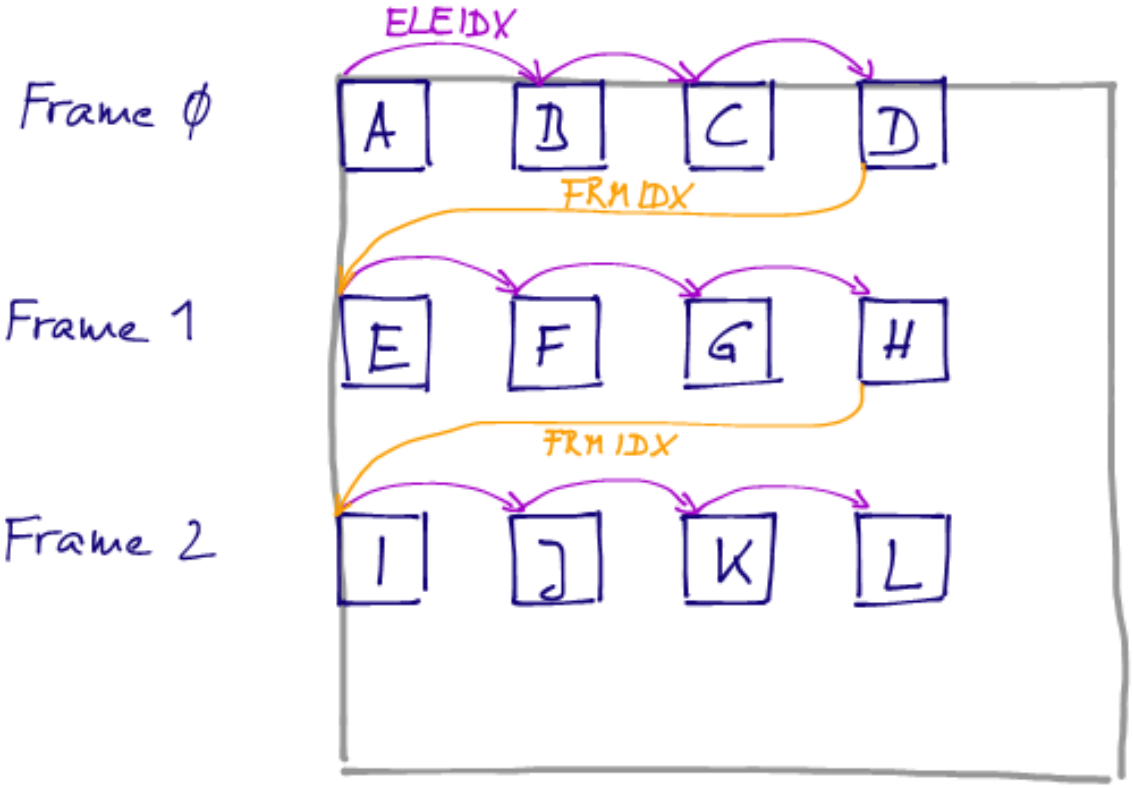
- Event-triggered transfer request
  - Peripheral
  - Externally generated trigger
- CPU-triggered transfer request
  - CPU “manually” triggers a transfer
- Chain-triggered transfer request
  - Completion of a given transfer triggers a new EDMA channel transfer request

EDMA Transfer Modes

- 1-dimensional transfers
  - 1D element synchronized
  - 1D frame synchronized
- 2-dimensional transfers
  - 2D array synchronized
  - 2D block synchronized

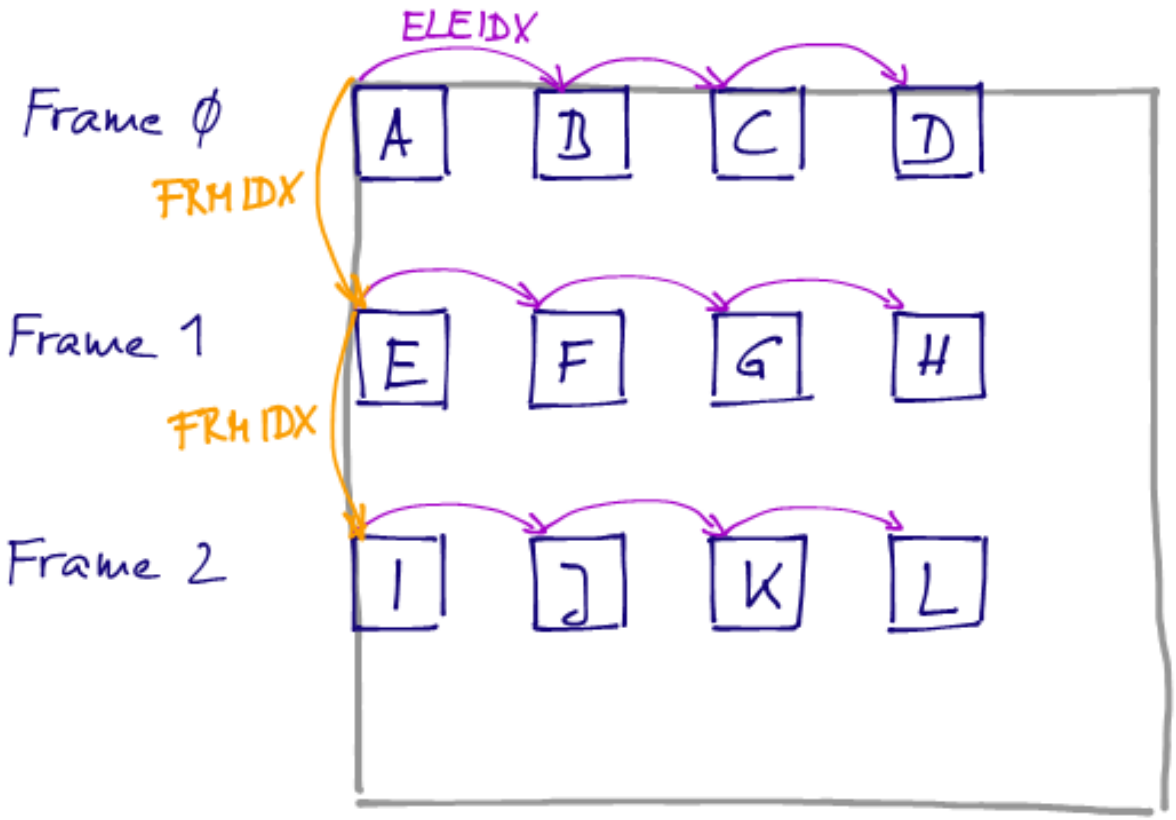
EDMA Transfer Modes

- 1D element synchronized

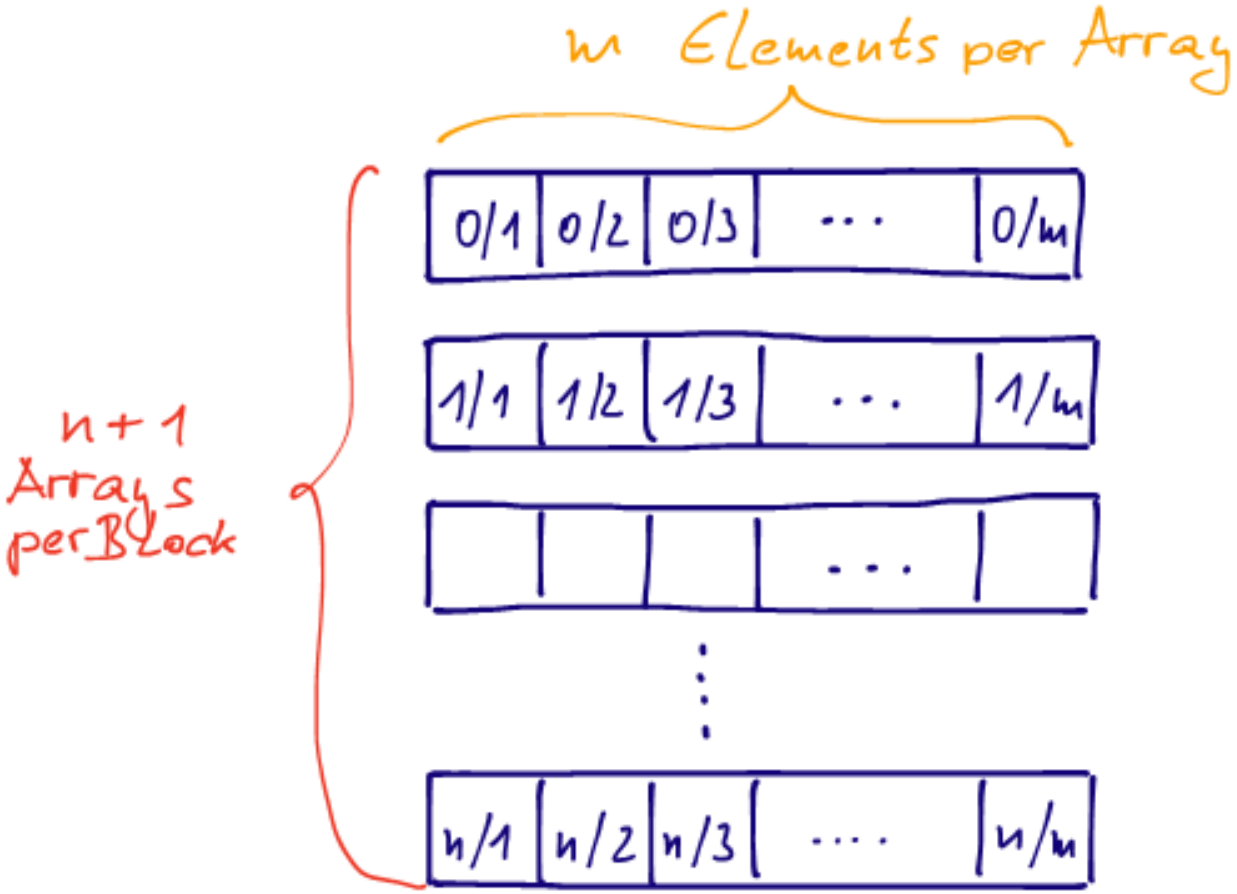


EDMA Transfer Modes

- 1D frame synchronized

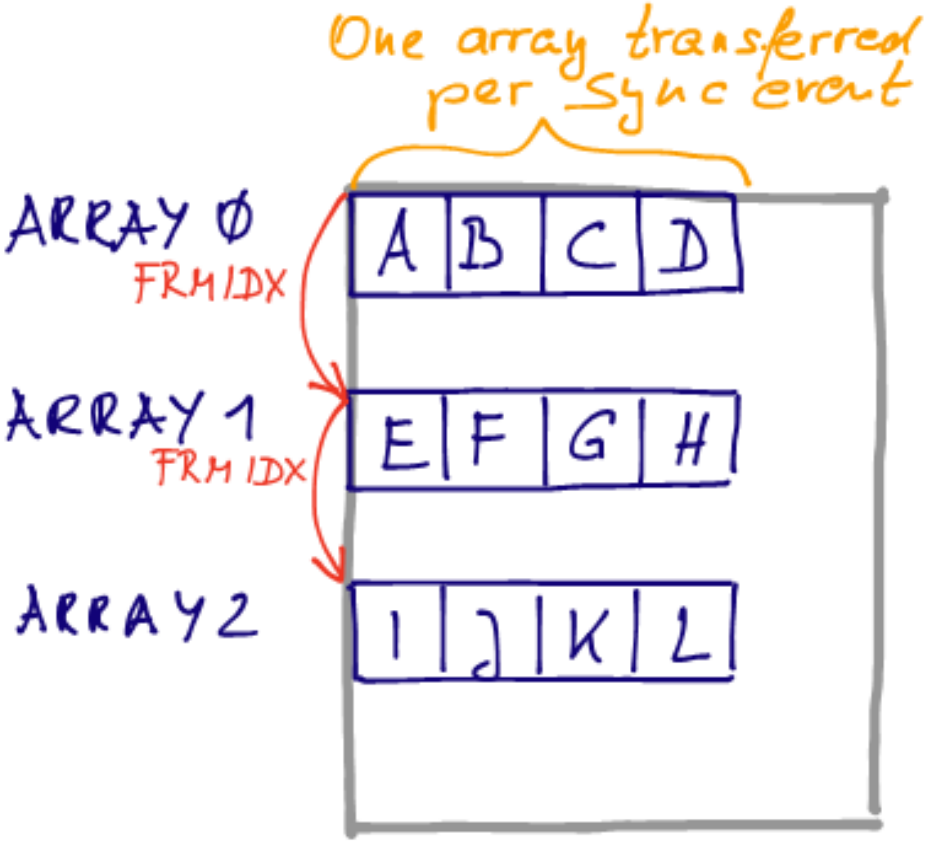


EDMA 2D Data Organization



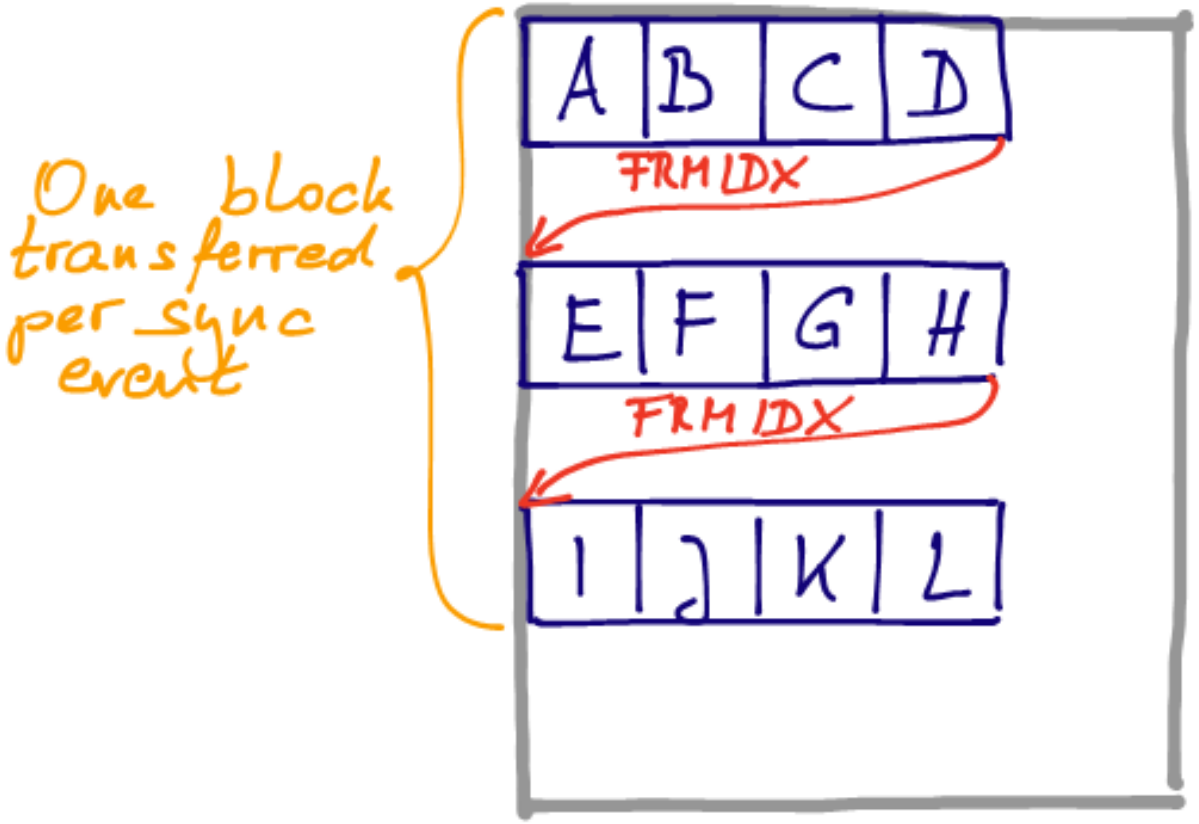
EDMA Transfer Modes

- 2D array synchronized



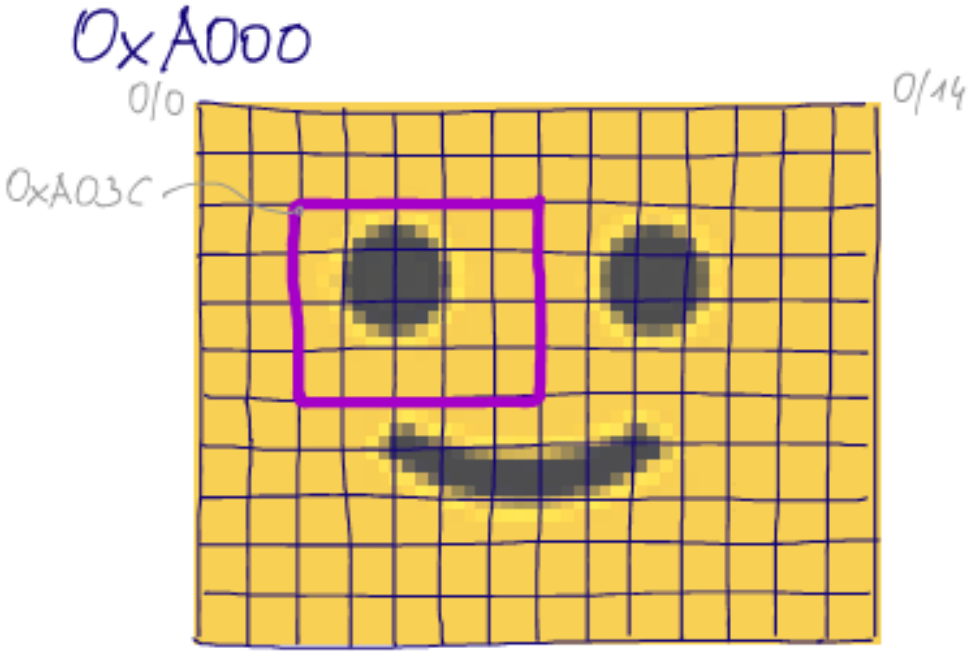
EDMA Transfer Modes

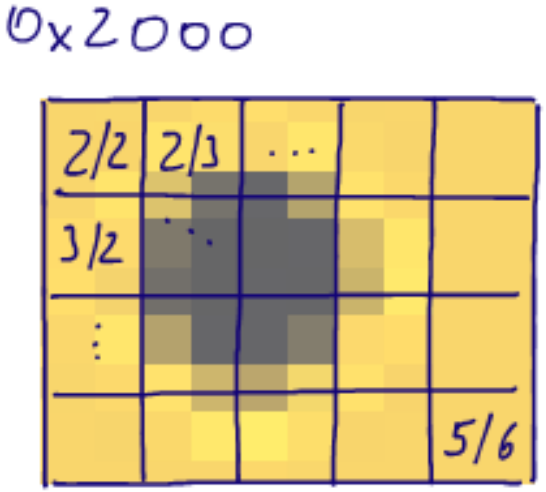
- 2D block synchronized



EDMA Transfer Examples

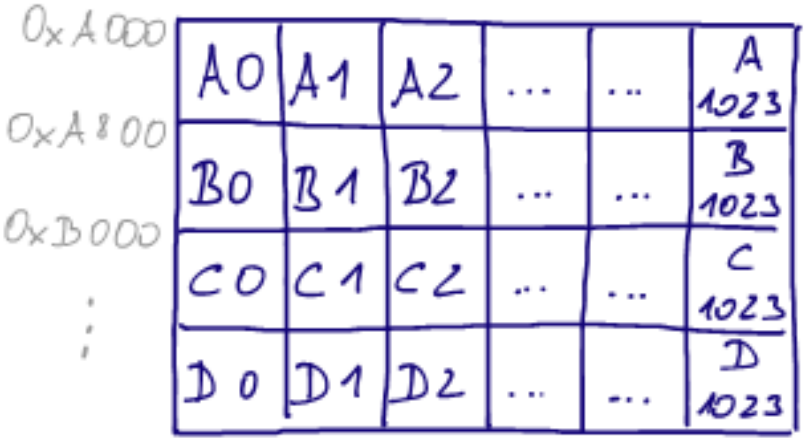
- Subframe extraction

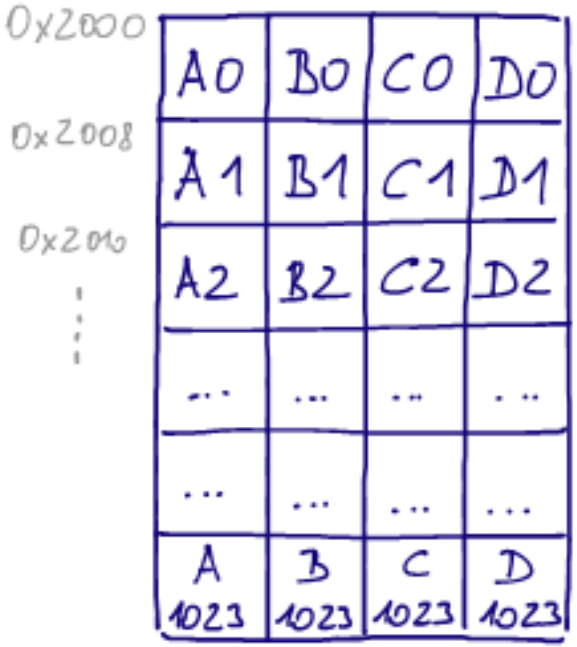




EDMA Transfer Examples

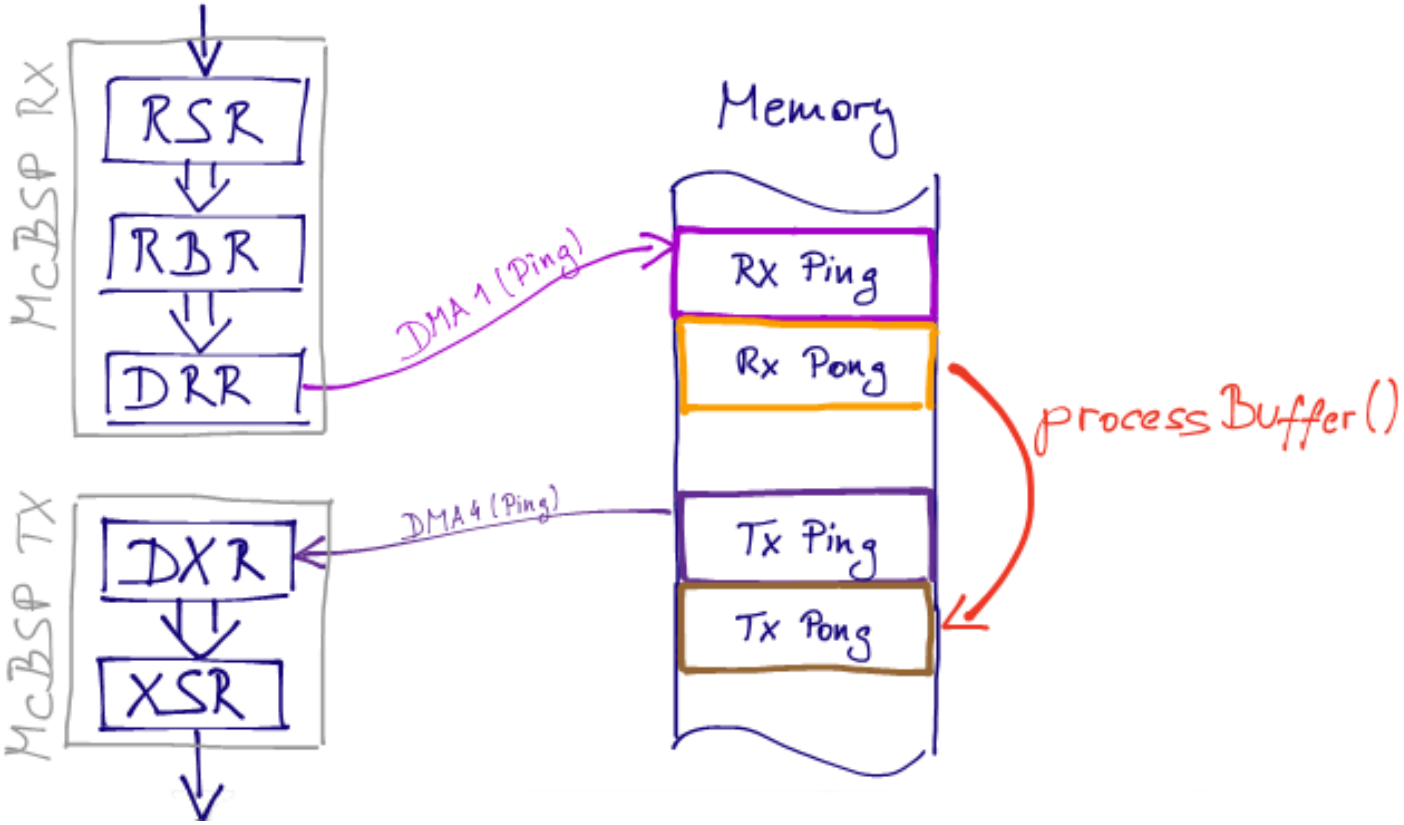
- Data sorting/rearranging





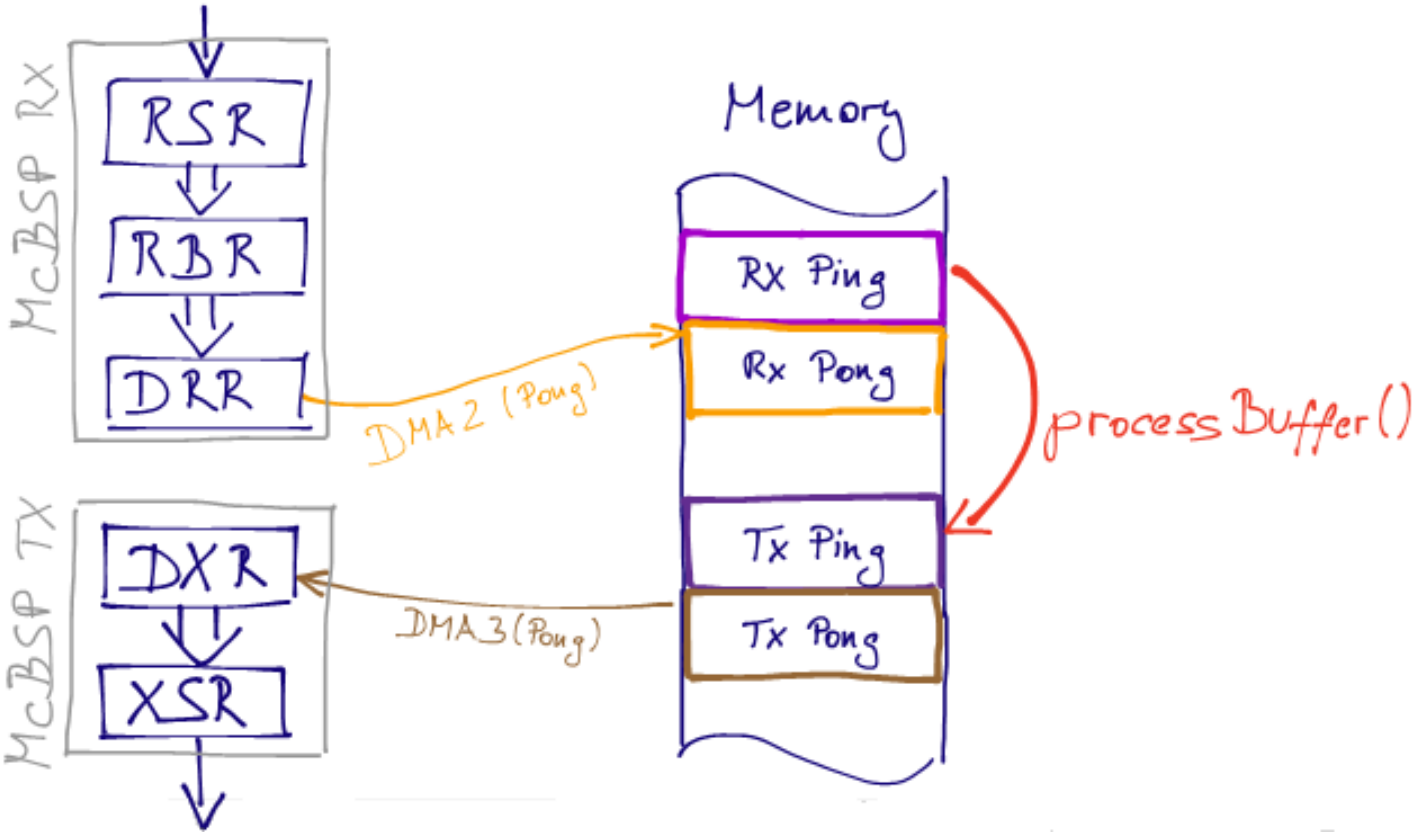
EDMA Transfer Examples

- Ping-pong buffering



EDMA Transfer Examples

- Ping-pong buffering



Quick Direct Memory Access (QDMA)

- CPU directly submits transfer request to transfer controller (EDMATC)
- QDMA registers are local to the CPU
  - Faster setup
- Transfer of only *one frame* (1D) or one block (2D)
- Used for fast, one-time transfers

EDMA Performance

- Peak performance: 1 element per cycle
  - Element size: 32 bit (word)
    => *2.4 GB/s* @ 600 MHz CPU
- Reasons for degraded performance:
  - Multiple concurrent DMA transfers (same priority)
  - Small element size
  - Small element count

  => EDMA is designed to transfer large amounts of data

Characteristics of DSP Architectures

Special Program Control

- Loops in software
  - Increment / Decrement loop counter
  - Compare loop counter
  - Conditional branch
- Zero overhead loops
  - Processor can execute loops without consuming cycles to decrement & test loop counter and branch
  - Zero overhead compared to *unrolled loop*

Special Program Control

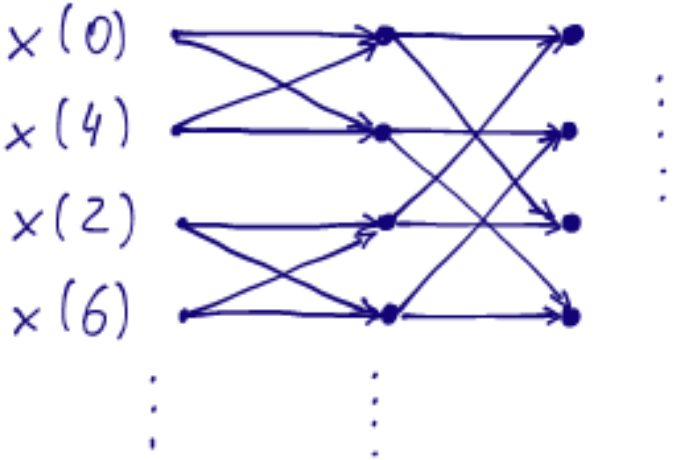
- Short interrupt latency
  - Fast react to (external) events
  - Real-time applications
  - Shadow register file
- Multiple stacks

Special Addressing Modes

- Address generation
  - Conversion of logical to physical address
  - Separate address generation unit and address registers
  - Often in combination with “post-increment” or similar
- Circular- or Modulo-addressing
  - Fast access to ring-buffers
  - No overhead for boundary check (auto-wrap)

Special Addressing Modes

- Bit-reversed addressing
  - Reversed weights of address bits (e.g., “butterfly”-operation for FFT)



Idx	Bin	Bit rev.	Idx (rev.)
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

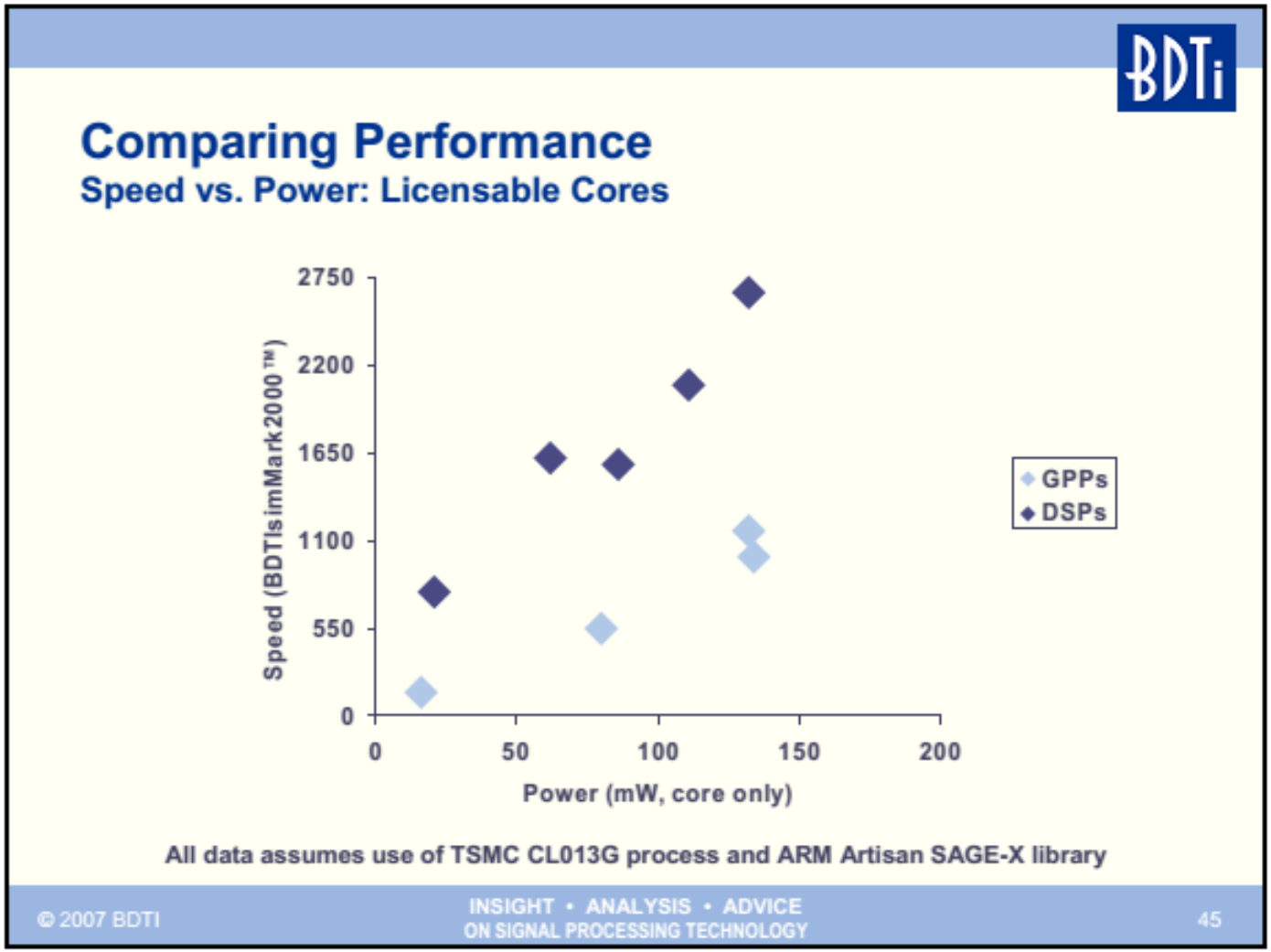
Power Management

- Low energy consumption
  - Top priority for mobile applications
  - Standardized parameter: Watt / MIPS
- Depends on usage profile, power management profile

Power Management Techniques

- Methods for reducing energy consumption
  - Reduce supply voltage
    e.g., 1.2 V for the core
  - Reduce clock frequency
    $P_{CMOS} = \alpha * C_L * f_{clk} * V_{DD}^2$
  - Switch off / deep sleep unused components
- Practical realizations
  - Bus encoding
  - Clock gating
  - Dynamic voltage scaling
  - Implementation at circuit level
  - Optimized software

Power Management



Peripherals & Input/Output Units

- On-Chip
  - Because often applied in embedded systems
- I/O Interface (example)
  - Serial port
  - Host interfaces
  - DMA
- Peripherals (example)
  - Timer
  - PLL
  - ADC, DAC

Selected DSP Implementations

Multi-Chip, Single-Chip

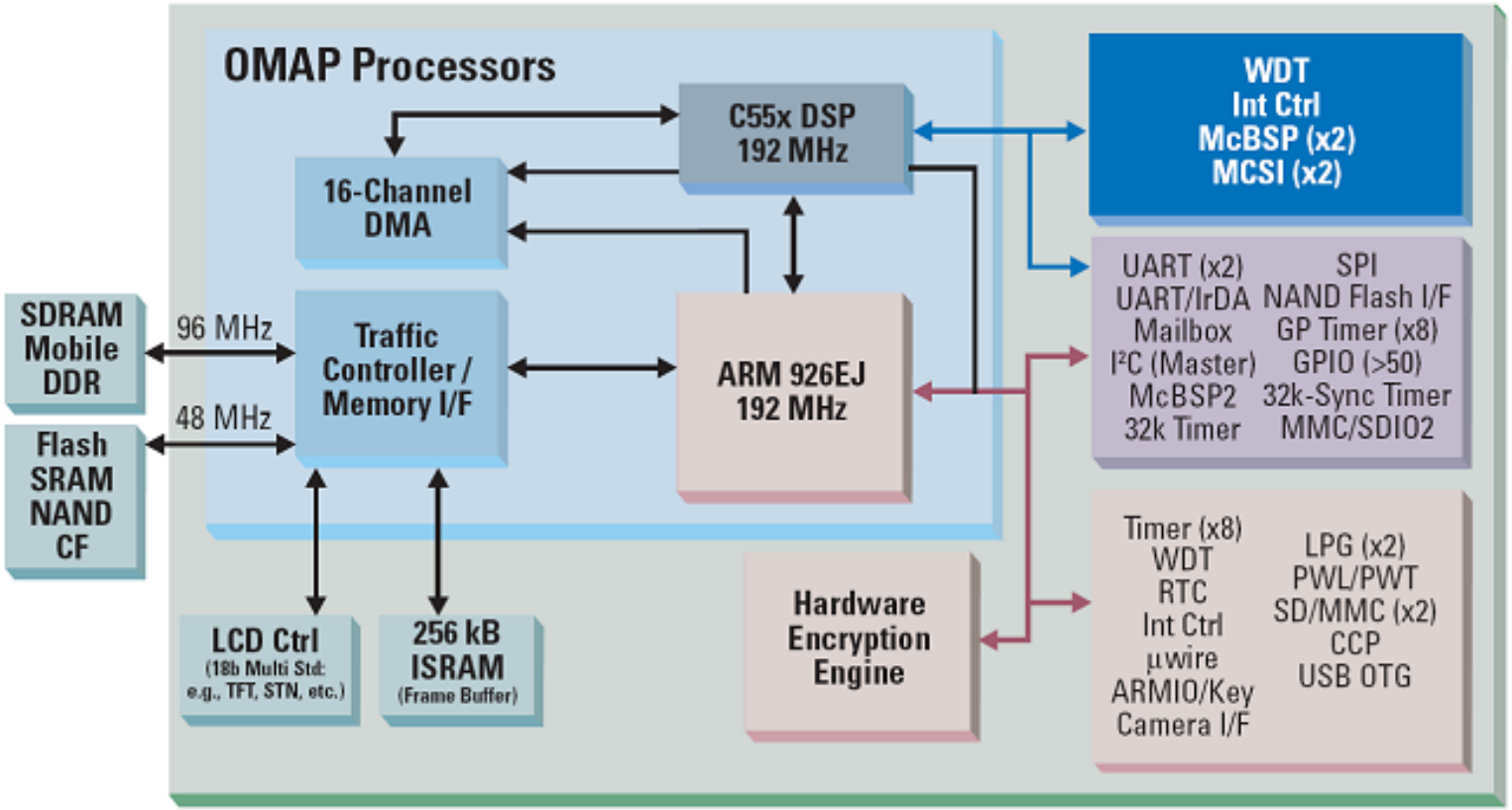
- Multi-chip Modules
  - Functionality of DSP is partitioned into several chips
  - Nowadays obsolete because of high integration level
- Single-chip processor
  - De-facto standard implementation
  - Mostly with internal memories and caches
  - Peripherals and I/O included

Multi-Core, Licensed Core

- Multi-core processors
  - Multiple DSPs
  - Homogeneous as well as heterogeneous architectures
    - TI C8x: 1x RISC + 4 DSP
    - TI OMAP: 1x DSP + 1 ARM-RISC
    - DaVinci: ARM and/or DSPs
- (licensable) DSP Cores
  - For integration in system-on-chip (SoC) solutions
  - DSP core available as VHDL/Verilog module (optimized)

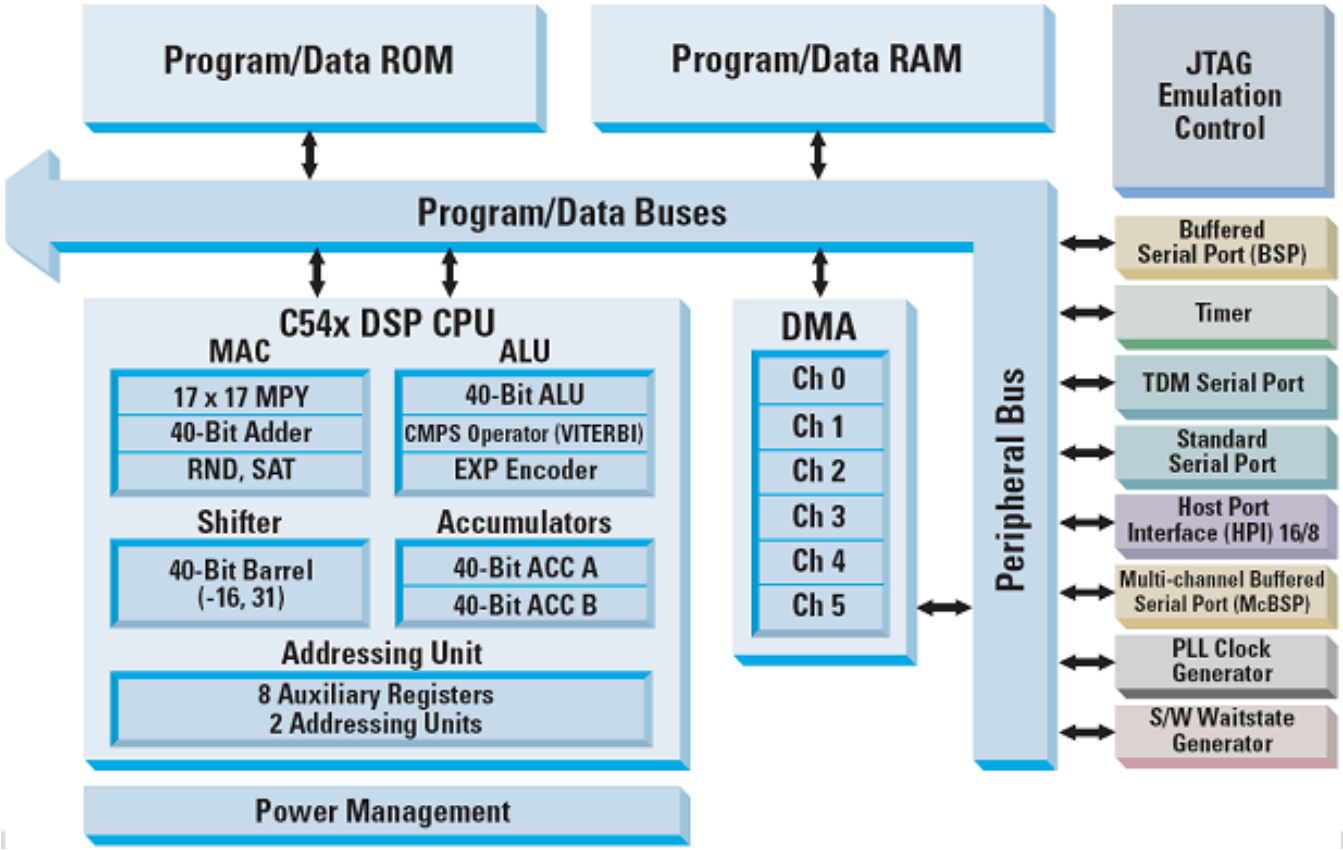
Multi-Core (Multiprocessor on-chip)

- TI OMAP



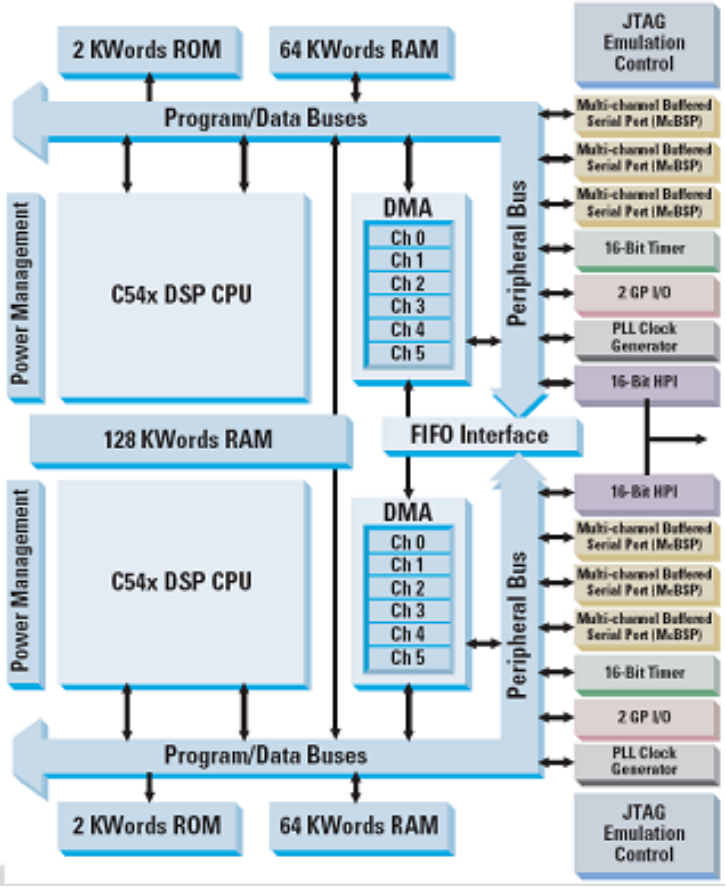
Multi-Core (Multiprocessor on-chip)

- TI TMS320C5421



Multi-Core (Multiprocessor on-chip)

- TI TMS320C5421 Multi-Core



Number Formats

Representing Numbers

- Unsigned integer
- Signed integer
- floating point numbers
- Signed/Unsigned fixed-point number

Signed/Unsigned Integer

- Unsigned integer
  - $0 \ldots 2^n - 1$

32	16	8	4	2	1
1	1	0	1	1	0

$= 2 + 4 + 16 + 32 = 54$

Signed/Unsigned Integer

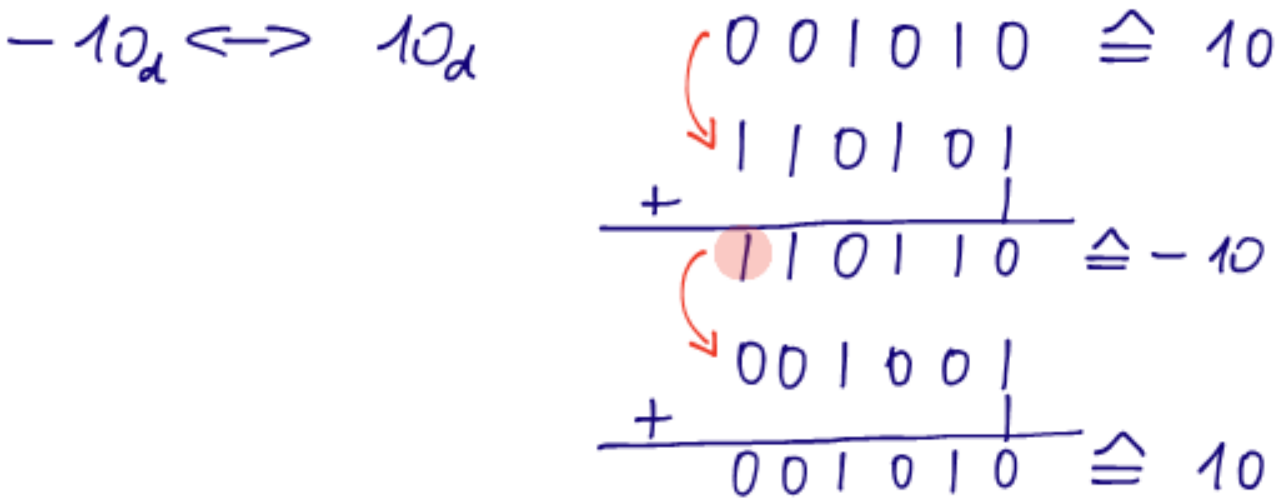
- Signed integer
  - Represented in 2's complement
  - $-2^{n-1} \ldots 2^{n-1} - 1$

−32	16	8	4	2	1
1	1	0	1	1	0

$= 2 + 4 + 16 - 32 = -10$

Signed/Unsigned Conversion

- 2's complement
  - Conversion: 1's complement + 1



Fractional Numbers: Fixed-Point

- Fractional weights of bits

1011,0110

2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
8	4	2	1	0,5	0,25	0,125	0,0625
1	0	1	1	0	1	1	0

$\hat{=}$ 11,375

-2^3
-8
1

Fractional Numbers: Fixed-Point

- Fractional weights of bits

$1_{|}0\,1\,1\,0\,1\,1\,0$

2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
1	0,5	0,25	0,125	0,0625	0,03125	0,015625	0,0078125
1	0	1	1	0	1	1	0

$\hat{=}\ 1,421875$

-2^0
-1
1

Fractional Numbers: Fixed-Point

- Use n bits for fractional part and
  m bits for integral part
  Q m.n notation, e.g. Q 0.15 / Q 0.16, Q 32.0, Q 4.12

$$Q_{m.n}: \quad x = -b_{N-1}2^{m} + \sum_{k=0}^{N-2} b_k 2^{k-n}$$

$$-2^{m} \quad \ldots \quad (2^{N-1}-1)2^{-n}$$

$$2^{N-n-1} - 2^{-n}$$

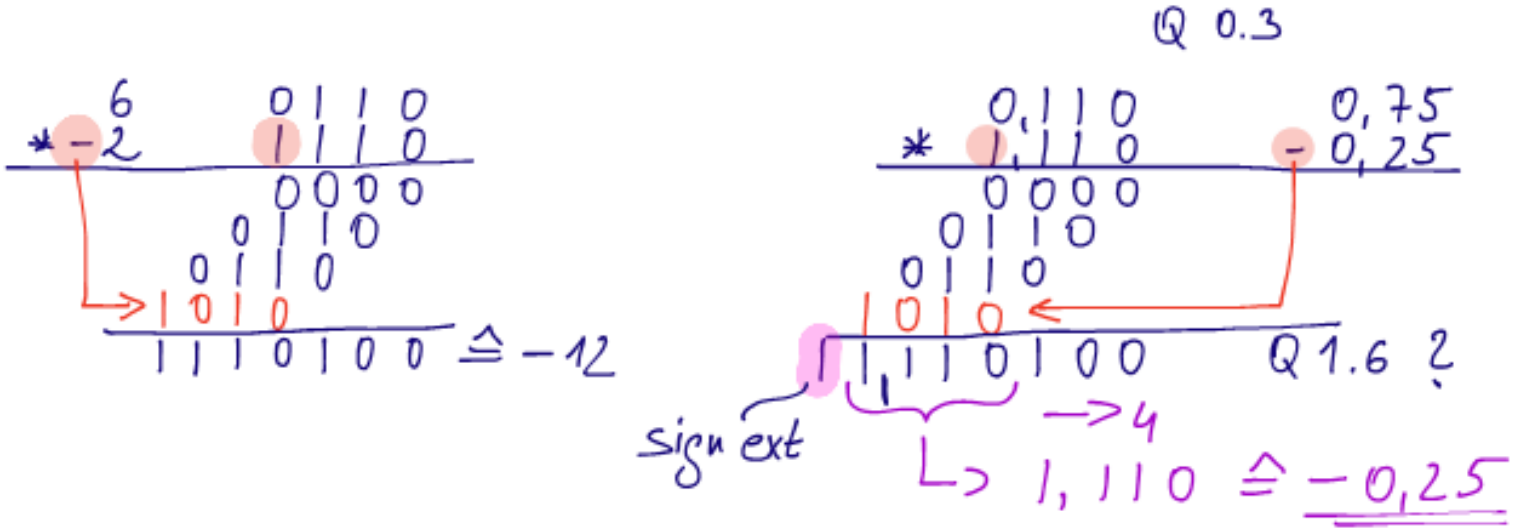
- *Programmer needs to keep track of implied fractional part!*

Fixed-Point Arithmetic

- Adding/subtracting numbers
  - Same as adding integer numbers
  - *Both numbers have to be in the same format!*

Fixed-Point Arithmetic

- Multiplying numbers



- Numbers can be in different format
- *Keep track of decimal point!*

Floating-Point Numbers

- Single precision `(float)`

$$(-1)^S \cdot \left(1 + \sum_{i=1}^{23} b_{23-i} \cdot 2^{-i}\right) \cdot 2^{e-127}$$

$$\pm 1,18 \cdot 10^{-38} \;\ldots\; 3,4 \cdot 10^{38}$$

- Double precision `(double)`

$$(-1)^S \cdot \left(1 + \sum_{i=1}^{52} b_{52-i} \cdot 2^{-i}\right) \cdot 2^{e-1023}$$

$$\pm 2.23 \cdot 10^{-308} \;\ldots\; 1,80 \cdot 10^{308}$$

Floating-Point Numbers

- Floating-point representation

$$12{,}375 = 1100{,}011$$
$$= 1{,}100011 \cdot 2^3$$

$S = 0$

$E = 3 + 127 = 130$

$F = 100011$

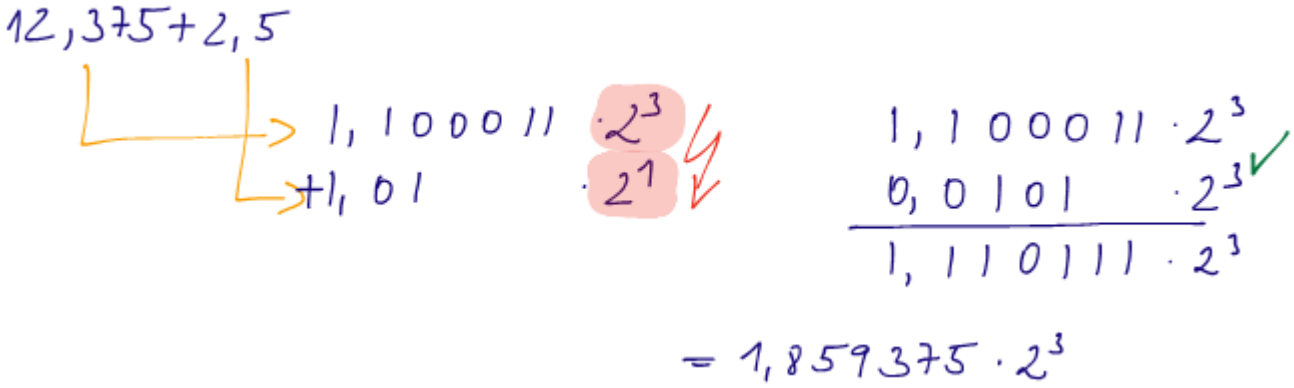
$$0 \mid 1000001 0 \mid 1000110 0 \ldots 0$$

Floating-Point Numbers

- Special numbers
  - *NAN*: E = 255, F ≠ 0
  - ±∞: E = 255, F = 0
  - ±0: E = 0, F = 0

Floating-Point Arithmetic

- Adding/subtracting numbers



- Procedure
  - Make exponents equal
  - Add mantissa
  - Normalize result

Floating-Point Arithmetic

- Multiplying numbers

$$2,44 * -12,16$$

$$-1^{0}\ 1,22 \cdot 2^{1} * -1^{1} \cdot 1,52 \cdot 2^{3}$$

$$= -1^{0+1} \cdot 1,22 \cdot 1,52 \cdot 2^{3+1} = -1^{1} \cdot 1,8544 \cdot 2^{4}$$

- Procedure
  - Multiply mantissa
  - Add exponents
  - Normalize result

Precision & Dynamic Range

	Range	Dynamic (dB)	Precision
Unsigned Integer (16 bit)	0 bis 65,536	$20\ log_{10}(2^{16}) = 96$	1
Signed Integer (16 bit)	-32,768 bis 32,767	$20\ log_{10}(2^{15}) = 90$	1
Unsigned Fractional (Q 0.16)	0 bis 0.99998474	96	2^{-16}
Signed Fractional (Q 0.16)	-1 bis 0.99996948	90	2^{-15}
Single Precision (16 bit)	$1.18 \cdot 10^{-38}$ bis $3.4 \cdot 10^{38}$	1529.19	2^{-23}
Double Precision (32 bit)	$2.23 \cdot 10^{-308}$ bis $1.80 \cdot 10^{308}$	12318.15	2^{-52}

Block Floating-Point format

$$\begin{pmatrix} 0,0011 \\ 0,0101 \\ 0,0010 \end{pmatrix} = \begin{pmatrix} 0,1100 \cdot 2^{-2} \\ 0,1010 \cdot 2^{-1} \\ 0,1000 \cdot 2^{-2} \end{pmatrix} = \begin{pmatrix} 0,0110 \\ 0,1010 \\ 0,0100 \end{pmatrix} \cdot 2^{-1}$$

Canonical Signed Digit (CSD) Coding

- Ternary logic
  - -1, 0, +1

- Useful on hardware without parallel multiplier
  - Groups of 1s are eliminated

$\bar{1} = -1$

$0,\underbrace{0111}_{100\bar{1}}\,00\,\underbrace{011}_{10\bar{1}} \;\hat{=}\; 0,100\bar{1}00\bar{1}0\bar{1}$

CSD-Coding

- Example

$y = c \cdot x \qquad c = 0,875 \mathrel{\widehat{=}} 0,111$

$y = x \cdot 2^{-1} + x \cdot 2^{-2} + x \cdot 2^{-3}$

$c = 0,111 \mathrel{\widehat{=}} 1,00\overline{1} \quad (= 1 - 2^{-3})$

$y = x \cdot 2^{0} - x \cdot 2^{-3}$

Comparison

	Fixed-point	Floating-Point
Bit width	16 or 24	32
Dynamic range	higher	lower
SW-Development effort	higher	
Compiler support	limited	good
Product development time	long	quick
Clock rate	faster	
Die size	smaller	
Price	cheaper	more expensive
Power consumption	lower	

Application Domains

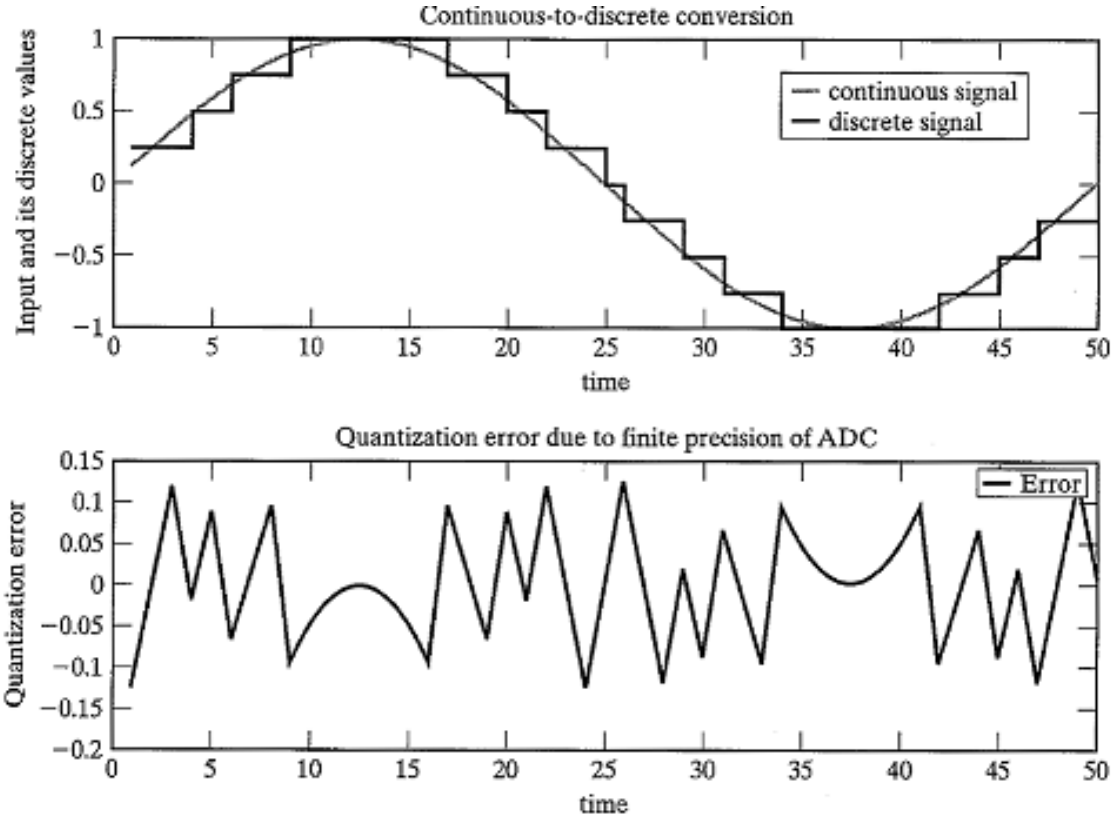
- Fixed-point processors
  - Disk drive and motor control
  - Consumer audio applications (MP3, gaming, cameras)
  - Speech coding/decoding, channel coding
  - Communication devices (modem, cellphones)
- Floating-point processors
  - Image processing (visual, radar, sonar, seismic)
  - High-end audio (synthesis, coding/decoding, mixing)
  - Prototyping

Finite-Wordlength Effects

Finite-Wordlength Effects

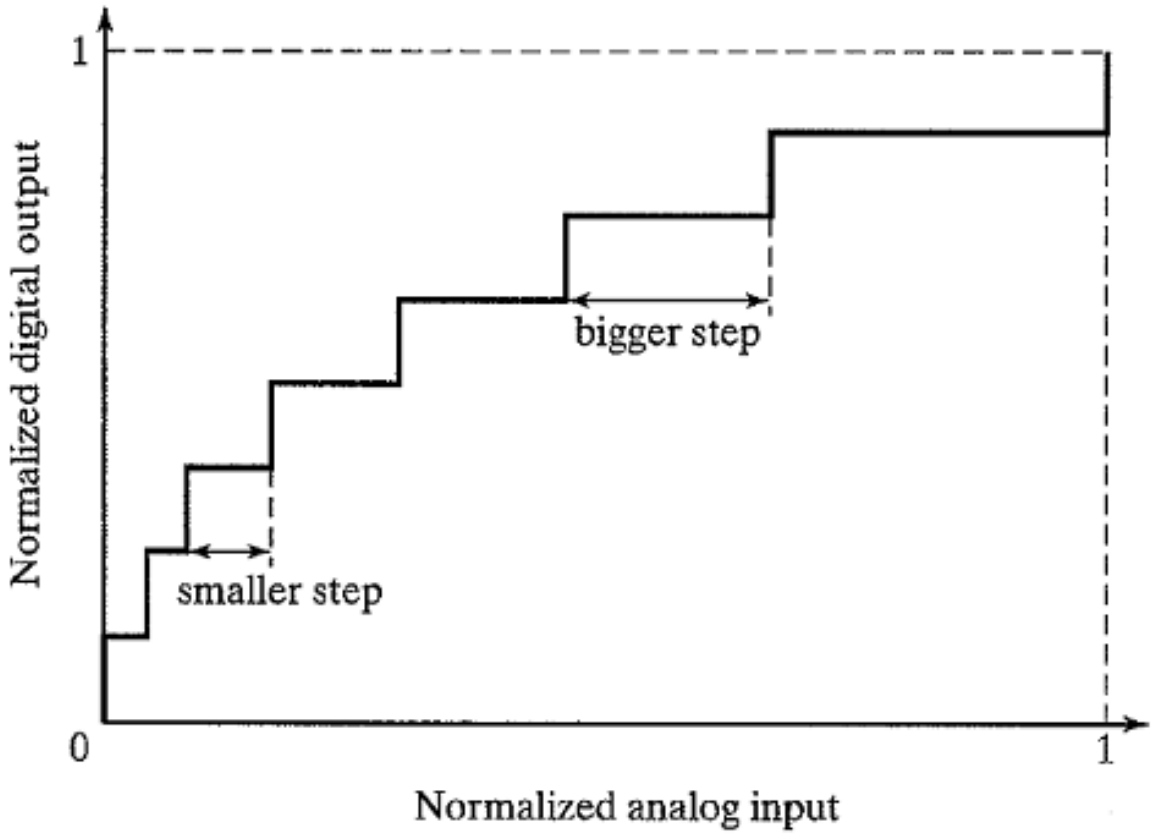
- Causes
  - Limited precision of ADC
  - Limited precision representing coefficients
  - Limited dynamic range in arithmetic operations
  - Rounding / truncating of data for storage / output

Input Quantization



$$SQNR = 6.02 \cdot N + 4.77 + 10 \cdot log_{10}(\sigma_x^2) \quad [dB]$$

Input Quantization



Nonlinear companding characteristics between the analog input and the digitized output

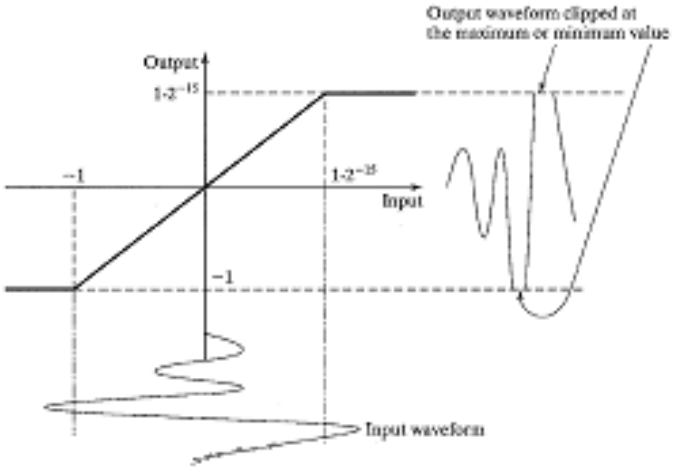
Coefficient Quantization

- Limited precision of coefficients may affect the system
  - e.g., significantly altered response of a filter
- Example: symmetrical FIR filter, L=40

Format	Error (SSE)
[16, 15] / Q.15	3.09^{-9}
[10, 9] / Q.9	1.46729^{-5}
[8, 7] / Q.7	1.864^{-4}

Overflow

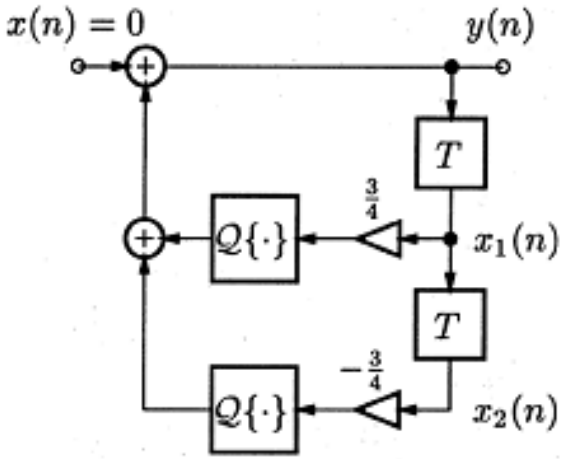
- Adding two N-bit numbers results in a sum of N+1 bits
  - Overflow / underflow
  - Distorted signal
- Accumulator with 8 guard bits allows up to 256 additions without overflow
- Saturated operations

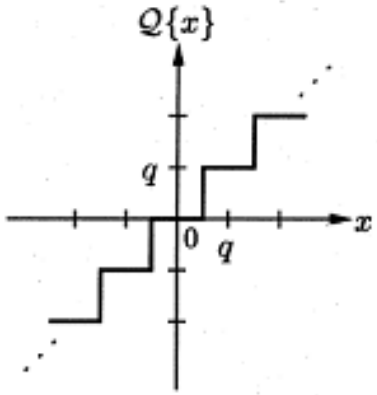


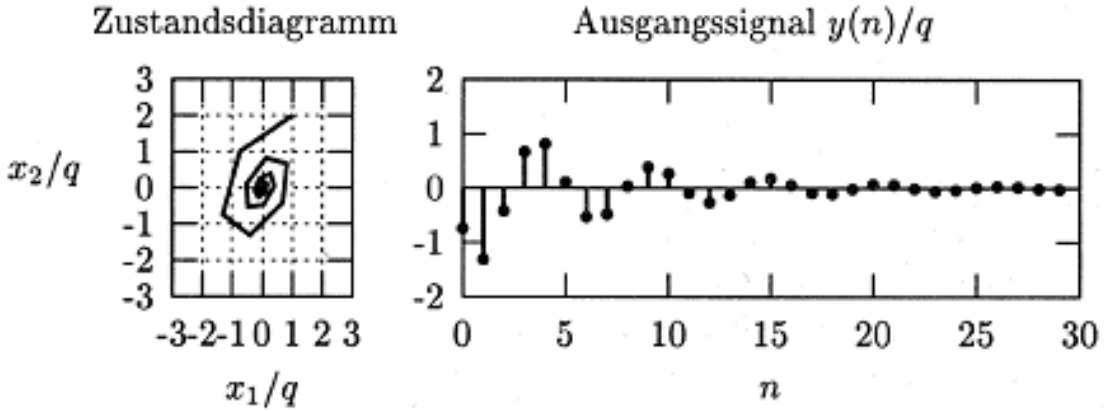
Overflow

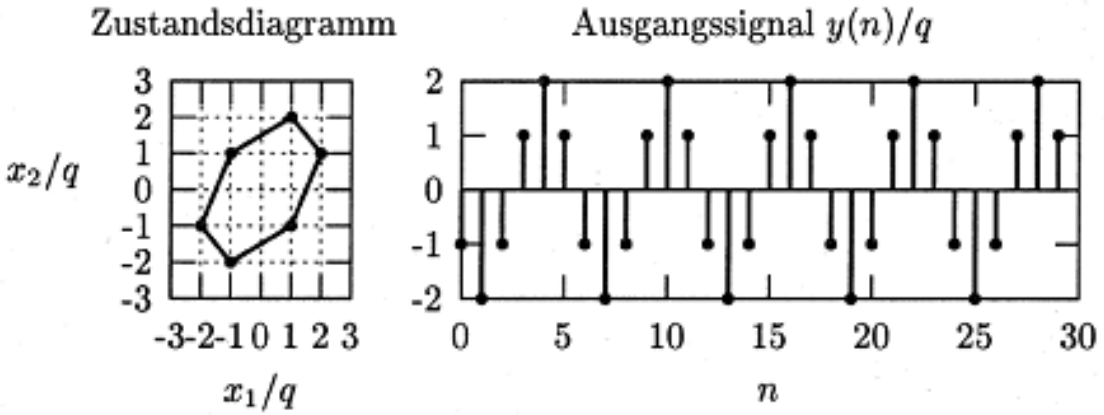
- Multiplication of two N-bit numbers results in a 2N bits number
  - Rounding or truncating numbers to prevent growing word length

Quantization Limit Cycles (Granulargrenzzyklen)

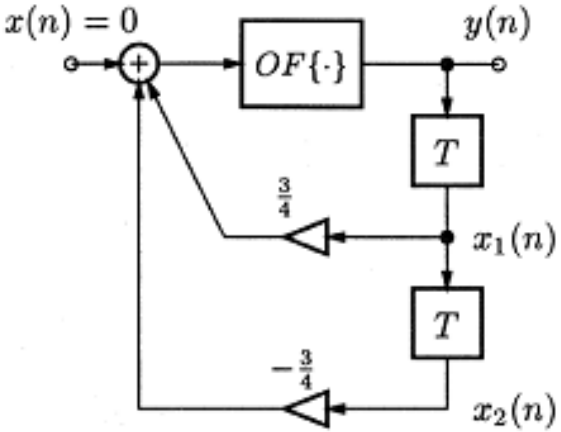


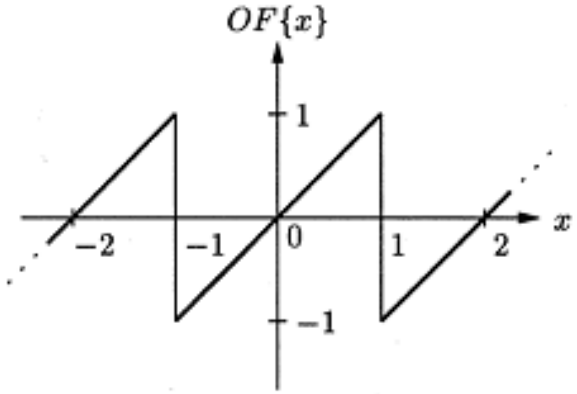


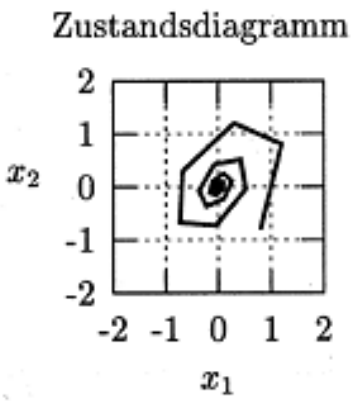


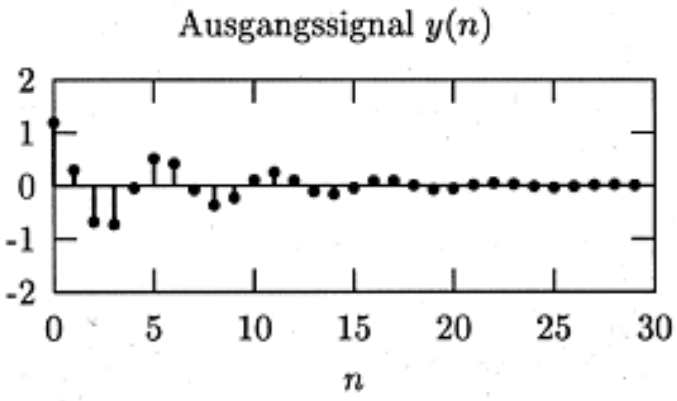


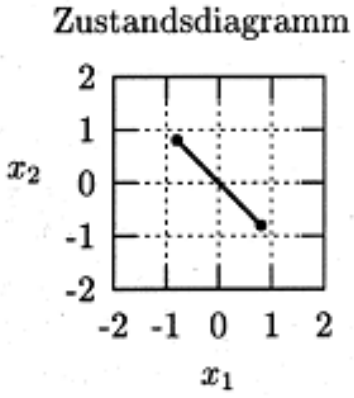
Overflow Oscillations (Überlaufgrenzzyklen)

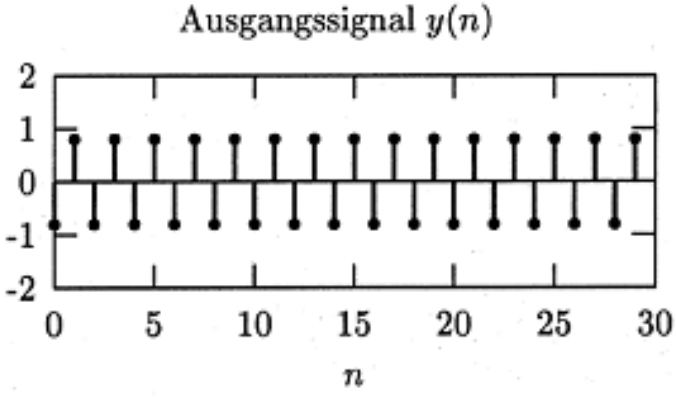












DSPs vs. General Purpose CPUs

DSPs vs. General Purpose CPUs

- What is missing
  - Branch Prediction
  - Out-of-order execution
  - Speculative execution
  - Superscalar
- Different requirements
  - Hard real-time constraints
  - Power consumption
  - . . .

Conclusion