Applications of signal processors

INTRODUCTION TO DIGITAL SIGNAL PROCESSORS

Author: Grzegorz Szwoch

Gdańsk University of Technology, Department of Multimedia Systems

Lecture organization

Course „Zastosowania procesorów sygnałowych”:

- Semester 5 – lecture (1 h weekly), in Polish.
- Semester 6 – project (2 h every two weeks).
  Students perform project tasks on their own in our laboratory room, with DSP development boards.

Course supervisor:

Grzegorz Szwoch

greg@multimed.org

room WETI 732

Criteria of finishing the course

- Final exam – during the last lecture.
- 10 questions × 3 pts, max. 30 pts in total.
- At least 16 points needed to pass.
- Second exam – during the second session.
- If a student fails both exams, they can get additional chances to pass, provided that:
  - they missed the pass threshold by at most 3 pts,
  - they were present at 80% or more of the lectures.
- Students presence at the lectures will be recorded.

Different criteria may be established for non-Polish speaking students.

Literature

All lecture slides are available on the Department website – multimed.org, “Lecture slides” page.

Additional literature (suggested):

- S.M. Kuo, B.H. Lee, W. Tian: “Real-Time Digital Signal Processing”. Wiley 2013.
- S.W. Smith: „The Scientist and Engineer's Guide to Digital Signal Processing”. www.dspguide.com
- Search the Internet ☺

The course plan

- Introduction to digital signal processors (DSP)
- DSP architecture
- Operating systems on DSP
- Numeric systems
- DSP interfaces
- Fourier transform and convolution
- Digital filters – FIR and IIR
- Signal generation
- Advanced digital signal processing algorithms
- Applications of DSP in digital cameras
- Medical applications of DSP
- The final exam

Motivation

How are DSPs related to telecommunication?

- Signal processors are used in many applications (professional and consumer) in the telecommunication area.
- In their future work, students are likely to cope with systems containing digital signal processors.
- Therefore, they should know how DSP works and how are typical signal processing algorithms implemented.
- We want to show that knowledge from earlier courses (Signal processing, Sound and vision processing) is not only a theory – it is used in practice, with DSPs.

Digital signal processor

Digital signal processor (DSP)
is an electronic chipset that is specialized for optimal processing of digital signal samples, performing repeated operations on each signal sample.

By “optimal”, we mean:

- low processing time (“real time processing”),
- low energy consumption.

These features separate DSPs
from general purpose processors (CPU).

Signal processing methods

There are two main approaches to signal processing.

- *Offline* processing (“file processing”)
  - the complete signal is available,
  - processing time is not critical,
  - a general-purpose processor is more suitable for this task.
- *Online* processing (“real time processing”):
  - the signal is infinite – new samples arrive continuously,
  - each sample must be processed before another one arrives,
  - samples may also be processed in blocks,
  - this is a typical application of signal processors.

Real time processing

- *ts* – time between two successive samples
- *tp* – processing time of one sample
- *to* – overhead time, not related to processing
- *ti* – idle time – waiting for a new sample

"Real time processing" is defined in different ways, depending on the context. For the purpose of digital signal processing, we can assume that real time processing requires that:

$$tp + to < ts$$

$$ti > 0$$

Why not a microprocessor?

Can we use a standard microprocessor (CPU) to perform real time processing of digital signals?

- A microprocessor is a general-purpose device, it performs a wide range of operations; signal processing is not treated in a special way.
- An operating system is required. A typical OS does not ensure constant and predictable processing time (overhead is not constant).
- They require external elements to work, they are not suitable for embedded systems (large size).
- High clock frequency, high energy consumption, high usage cost.

Why a signal processor?

- Architecture and operation of a DSP is optimized for processing samples of digital signals.
- Usually, low clock frequency (ca. 100 MHz).
- Low energy consumption, compared with microprocessors.
- Contains interfaces – it's easy to input/output signals.
- Small size, can be used in embedded systems.
- A ratio of processing time to cost is much better than for microprocessors.

When should we use DSP?

We should use a signal processor:

- for complex, real-time signal processing
- as an element of a device, e.g. a digital camera
- when low energy consumption is needed (e.g. from battery)
- when signal processing consists of basic DSP operations

We should use a microprocessor:

- for offline processing
- when operating system is needed (e.g. a smartphone)
- when a microprocessor is used anyway for other tasks
- when a large degree of flexibility is needed, and energy consumption and device size are less important

Signal processing algorithms

Algorithms for DSP use basic signal processing operations:

- multiply, add, logical operations –often vectorized
- Fourier transform (FFT) and cosine transform (DCT)
- convolution / FIR filtering, correlation, autocorrelation
- IIR filtering
- signal interpolation
- matrix multiplication

DSP is optimized for performing these operations.

Complex signal processing algorithms are built from these basic operations.

Machine code

- Machine code – a set of binary instructions for a processor (e.g. multiply, jump, loop).
- A programmer creates source code of a program, as a text.
- A compiler is software that converts the source code into the machine code.
- A linker is software that merges compiled modules into an executable program.
- When a program is built, it can be executed on a processor.

Assembler

- Assembler represents machine code with readable text.
- Processor instructions are written as mnemonics,
  e.g. MOV – copy data, MPY – multiply, etc.
- The Assembler code is written for a specific processor.
- It operates directly on the processor (e.g. on registers).
- A programmer has almost full control over the resulting machine code.
- It's possible to write highly optimized programs.
- Writing programs in Assembler is hard, requires knowledge of how a processor works, and a lot of experience.
- (We don't use Assembler in our course.)

Assembler

An example of a DSP program written in Assembler:

```
 || RPTBLOCAL OuterLoopEnd-1                          ;outer loop: process a new input
                MOV     *AR0+ << #16, AC1                 ; HI(AC1) = x(n)
                ||RPTBLOCAL     InnerLoopEnd-1            ;inner loop: process a bi-quad
                        NOP_16                            ; CPU_116: Remark 5682
                     || MPYM    *AR1+, AC1, AC0           ; AC0 = b0*x(n)
                        MACM    *AR1+, *(AR6+T0), AC0     ; AC0 += b1*x(n-1)
                        MACM    *AR1+, *AR6, AC0          ; AC0 += b2*x(n-2)
                        MOV     HI(AC1), *(AR6+T1)        ; x(n) replaces x(n-2)
                        MASM    *AR1+, *(AR4+T0), AC0     ; AC0 -= a0*y(n-1)
                        MASM    *AR1+, *AR4, AC0, AC1     ; AC1 -= a1*y(n-2)
                        SFTS    AC1, #1                   ; AC1 *= 2 (correction for Q14)
                        MOV     rnd(HI(AC1)), *(AR4+T1)   ; y(n) replaces y(n-2)
InnerLoopEnd:
                AMOV    AR7, AR1                          ;reinitialize coeff pointer
              || MOV    rnd(HI(AC1)), *AR2+               ;store result to output buffer
OuterLoopEnd:

        MOV     AR4, *AR3                                 ; Update delay pointer
```

C language

- The C programming language is often used as a “high level” language for DSP programming.
- We rely on the C compiler in creating optimal machine code.
- It’s often not possible, a compiler is not able to guess the programmer’s intentions and it “plays safe”.
- Special programming directives (*pragmas*) must be used.
- Compared with Assembler, the resulting code is often slower, and it uses more memory.
- Writing programs is easier and quicker.
- (We use C in our course project.)

C language

An example of a DSP program written in C:

```
// Real FFT of length N/2
for (i = 0; i < N / 2; i++) {
          pRFFT_In[2 * i] = pInput[2 * i];           //arrange real input sequence to
          pRFFT_In[2 * i + 1] = pInput[2 * i + 1];   //N/2 complex sequence..
}

memcpy (pRFFT_InOrig, pRFFT_In, N * sizeof (float));
tw_gen (w, N / 2);
split_gen (A, B, N / 2);
twiddle = (float *) w;

// Forward FFT Calculation using N/2 complex FFT..
DSPF_sp_fftSPxSP (N / 2, pRFFT_In, twiddle, pTemp, brev, rad, 0, N / 2);
// FFT Split call to get complex FFT out of length N..
FFT_Split (N / 2, pTemp, A, B, pRFFT_Out);

// Inverse FFT calculation
// IFFT Split call to get complex Inv FFT out of length N..
IFFT_Split (N / 2, pRFFT_Out, A, B, pTemp);
// Inverse FFT Calculation using N/2 complex IFFT..
DSPF_sp_ifftSPxSP (N / 2, pTemp, twiddle, pRFFT_InvOut, brev, rad, 0, N / 2);
```

C and Assembler working together

- Can we use both languages together? Yes, we can!
- A linker can merge modules written in Assembler and C into a single program.
- Critical operations may be written in Assembler, optimized to work very fast.
- A general “logic” of the program may be written in C.
- We can often use optimized signal processing procedures (such as FFT) written by others, linking them with our code.
- DSP vendors usually provide libraries with optimized DSP operations written in Assembler, e.g. DSPLIB for Texas Instruments DSPs.

Evaluation boards

Evaluation board / module / kit:

- a board containing a signal processor, interfaces and additional elements (memory, flash card, codecs, etc.)
- used for development, testing, debugging and optimization of DSP programs
- communicates with a PC via USB port
- allows for step by step program execution and evaluation of processor and memory state (via debug probe)
- in the final product, only the processor is mounted, not the whole development board

Evaluation boards

Texas Instruments C5535 evaluation board, used in this course

TMDX5535EZDSP USB Stick Development Kit

Development environment

In development of DSP programs, the following tools are used:

- programming tools (IDE)
  - source code editor
  - compiler and linker
  - debugger
- programming libraries (SDK)
  - processor functions (*Processor SDK*), including the operating system
  - development board functions (*Board SDK*)
  - signal processing functions (e.g. *DSPLIB*)
  - additional functions, e.g. network connectivity

Stages of program development

- A development board is connected to PC.
- The program is compiled in *Debug* mode
  – code optimizations are turned off.
- A compiled program is sent to the DSP board and executed.
- The program can be debugged – we can stop it, evaluate values of variables, search for errors.
- Program testing – if the results are valid.
- For performance tests, the program must be compiled in *Release* mode, with code optimizations turned on.
  In this mode, the program cannot be debugged.

A program is ready

- A program is ready when it works as expected – without errors and with satisfactory performance.
- The program is compiled in *Release* mode.
- The program is flashed into a persistent memory of DSP or the board, using a special module.
- The CPU is ready to be installed in the device.
- At this point, we can no longer debug the program, we can only obtain and analyze the results.
- Therefore, sufficient effort must be put into the development phase, so that the final program works as expected.

Performance scores of a DSP

Computational performance:

- MIPS – millions of instructions executed per second
- FLOPS – number of floating-point operations executed per second, usually with a prefix (e.g. MFLOPS – millions)
- MMACS – millions of MAC instructions executed per second; MAC is specific for DSP ($x \leftarrow x + a \times b$)

Energy performance:

- power per 1 MHz of DSP clock frequency, in mW/MHz, given a power supply voltage, measured in the active and in the standby modes

DSP producers

Main digital signal processor producers:

- Texas Instruments (www.ti.com)
- Analog Devices (www.analog.com)
- NXP Semiconductors (www.nxp.com)
- Freescale (www.freescale.com)
- XMOS (www.xmos.com)
- CEVA (www.ceva-dsp.com)

Example applications

DSP in a digital camera – functions:

- auto exposition
- auto focus
- auto program selection
- demosaicing of sensor data
- noise reduction
- composing HDR images
- special effects
- face detection
- moving object following, keeping it in focus
- JPG and MPEG compression

Applications – consumer products

- digital cameras and camcorders
- music and video players
- sound synthesizers and samplers
- electronic toys
- electronic babysitter
- home appliances, e.g. washing machine
- remotely controlled vehicles, drones
- smart home devices

Applications - telecommunication

- encoding, decoding, compression of sound and images
- noise and interference reduction
- acoustic and telecommunication echo removal
- improving speech intelligibility in presence of noise
- teleconference systems – enhancing the speaker direction
- speech recognition and synthesis, voice controlling
- video monitoring - video content recognition
- monitoring stations – analysis of sensor data

Applications - medicine

- analysis of test results (USG, tomography, etc.)
- analysis of brain waves (EEG)
- hearing aids and cochlear implants
- electronic throat, prostheses
- rehabilitation devices
- remote health monitoring, telemedicine
- physical activity monitoring

Other applications

Military applications:

- radar, sonar, LiDAR – object detection, speed measurement
- cryptography and watermarking – data protection
- navigation
- missile guidance

Car industry applications:

- car functions management (such as ABS)
- navigation
- hands-free audio system
- obstacle detection, parking assist
- autonomous vehicles

Multiprocessor and multicore systems

- In some cases, a single DSP does not have sufficient power to ensure real time signal processing.
- Possible solutions:
  - multicore signal processors,
  - multiprocessor systems – several processors working together.
- Example: image analysis “pixel by pixel” – the image can be split into segments, each processor/thread analyzes one part, the results are merged.
- Synchronization is a problem, it’s more difficult than in PC.
- Overhead time is high, programming is difficult.

Hybrid processors

- Two different processor types in a single chipset:
  - one or more CPU cores, usually ARM,
  - several DSP cores.
- The main core runs the operating system, e.g. Linux.
- A program running on the main core manages the computations.
- DSP cores perform processing operations.
- Complex programming (a system of queues).
- High overhead.
- High flexibility in program creation.

Examples of hybrid processors

66AK2H12 Keystone (Texas Instruments)

- 4 cores ARM A15
- 8 cores DSP C66x

Snapdragon 835 (Qualcomm, 2019)

- CPU: 8 cores ARM of different types (1+3+4)
- GPU (graphics processor)
- DSP: Hexagon 690, multicore, + *tensor accelerator*

DSP alternatives: FPGA

FPGA (*Field-Programmable Gate Array*)

- A chipset, which hardware architecture can be configured by a programmer.
- A matrix of programmable logic units with memory, connections between units may be modified.
- Architecture can be optimized for a specific algorithm.
- FPGA replaces DSP in applications in which DSP performance is not sufficient (parallel processing, high frequency signals).
- High flexibility.
- Very high cost.
- Main producers: Altera, Xilinx

DSP alternatives: ASIC

ASIC (*Application-Specific Integrated Circuit)*

- A specialized chipset, with architecture optimized for a specific application, in factory.
- It cannot be modified the way FPGA can.
- High (optimal) performance for that specific application.
- Faster than FPGA and DSP, uses less energy.
- No flexibility – only a specific application.
- High cost of design and production.
- If an error is found in the algorithm, it often cannot be fixed (only the firmware can be modified, but not the architecture). On the DSP, we can simply supply a new program.

DSP alternatives: GPU

GPU (*Graphic Processing Unit*)

- A chipset optimized for 3D images creation.
- A parallel processor – many (several hundreds) of processing units, performing the same instructions on different data (SIMD – single instruction, multiple data).
- Because of that, GPUs are used in parallel processing, e.g. image analysis or neural networks.
- High energy consumption, high cost of GPU and its usage.
- High overhead in data processing (synchronization).
- Example application: autonomous vehicles (environment analysis).
- Example GPUs: NVidia Jetson TX2, Jetson Nano.

SoC

SoC – *System on a Chip*

- A chipset that includes many components in a single chip:
  - microprocessor (CPU)
  - memory (different types)
  - input/output interfaces, including network
  - DSP (in some implementations)
  - GHPU (in some implementations)
  - additional coprocessors (complex implementations)
- Practically, a full microcomputer in a single chipset.
- SoCs are the current market trend, especially in mobile devices and embedded systems.