# Digital Signal Processing: An Introduction

Dmitry Teytelman

Dimtel, Inc., San Jose, CA, 95124, USA

June 16, 2009

# Outline

1. Introduction to DSP
   - Defining the Terms
   - Sampling and Quantization
   - Z-transform
   - Digital Filtering
   - Efficient Filter Structures
2. Real-time digital signal processing
   - Definition and applications
   - Available solutions
3. Advantages and disadvantages
   - General-purpose Processors
   - Special-purpose DSP chips
   - Field Programmable Gate Arrays

# Outline

1. Introduction to DSP
   - Defining the Terms
   - Sampling and Quantization
   - Z-transform
   - Digital Filtering
   - Efficient Filter Structures
2. Real-time digital signal processing
   - Definition and applications
   - Available solutions
3. Advantages and disadvantages
   - General-purpose Processors
   - Special-purpose DSP chips
   - Field Programmable Gate Arrays

# Digital Signal Processing: Domains

- Digital signal processing involves three important mathematical processes:
  - Time quantization — going from continuous to discrete time;
  - Amplitude quantization — going from continuous to discrete signal amplitudes;
  - Digital to analog conversion — going back to continuous time and amplitude.

# Discrete Amplitude and Noise

- Conceptually, continuous amplitude signal can take any value.
- In practice, there is some minimal voltage step $\Delta V$ that we can resolve.
- Why is that?
- Signal is useful information $V_c$ plus noise $V_n$.
- At increments comparable to noise RMS we can no longer distinguish signal values.
- Important point - amplitude quantization has certain dynamic range, but input signal must have higher SNR.

# Discrete Amplitude and Noise

- Conceptually, continuous amplitude signal can take any value.
- In practice, there is some minimal voltage step $\Delta V$ that we can resolve.
- Why is that?
- Signal is useful information $V_c$ plus noise $V_n$.
- At increments comparable to noise RMS we can no longer distinguish signal values.
- Important point - amplitude quantization has certain dynamic range, but input signal must have higher SNR.

# Discrete Amplitude and Noise

- Conceptually, continuous amplitude signal can take any value.
- In practice, there is some minimal voltage step $\Delta V$ that we can resolve.
- Why is that?
- Signal is useful information $V_c$ plus noise $V_n$.
- At increments comparable to noise RMS we can no longer distinguish signal values.
- Important point - amplitude quantization has certain dynamic range, but input signal must have higher SNR.

# Discrete Amplitude and Noise

- Conceptually, continuous amplitude signal can take any value.
- In practice, there is some minimal voltage step $\Delta V$ that we can resolve.
- Why is that?
- Signal is useful information $V_c$ plus noise $V_n$.
- At increments comparable to noise RMS we can no longer distinguish signal values.
- Important point - amplitude quantization has certain dynamic range, but input signal must have higher SNR.

# Discrete Amplitude and Noise

- Conceptually, continuous amplitude signal can take any value.
- In practice, there is some minimal voltage step $\Delta V$ that we can resolve.
- Why is that?
- Signal is useful information $V_c$ plus noise $V_n$.
- At increments comparable to noise RMS we can no longer distinguish signal values.
- Important point - amplitude quantization has certain dynamic range, but input signal must have higher SNR.

# Outline

1. Introduction to DSP
   - Defining the Terms
   - Sampling and Quantization
   - Z-transform
   - Digital Filtering
   - Efficient Filter Structures
2. Real-time digital signal processing
   - Definition and applications
   - Available solutions
3. Advantages and disadvantages
   - General-purpose Processors
   - Special-purpose DSP chips
   - Field Programmable Gate Arrays

# Time Sampling

## Continuous to Discrete Time

$$V_n = V_c(nT_s)$$

$$V_s = V_c(t) \sum_{n=-\infty}^{\infty} \delta(t - nT_s) = \sum_{n=-\infty}^{\infty} V_n \delta(t - nT_s)$$

- Multiply the signal by a train of delta functions.
- Multiplication in time domain means convolution in frequency domain.
- Information is lost in this conversion.
- Sampling period $T_s$, sampling frequency $f_s = 1/T_s$.
- Nyquist frequency.

# Time Sampling

## Continuous to Discrete Time

$$V_n = V_c(nT_s)$$

$$V_s = V_c(t) \sum_{n=-\infty}^{\infty} \delta(t - nT_s) = \sum_{n=-\infty}^{\infty} V_n \delta(t - nT_s)$$

- Multiply the signal by a train of delta functions.
- Multiplication in time domain means convolution in frequency domain.
- Information is lost in this conversion.
- Sampling period $T_s$, sampling frequency $f_s = 1/T_s$.
- Nyquist frequency.

# Time Sampling

## Continuous to Discrete Time

$$V_n = V_c(nT_\mathrm{s})$$

$$V_s = V_c(t) \sum_{n=-\infty}^{\infty} \delta(t - nT_\mathrm{s}) = \sum_{n=-\infty}^{\infty} V_n \delta(t - nT_\mathrm{s})$$

- Multiply the signal by a train of delta functions.
- Multiplication in time domain means convolution in frequency domain.
- Information is lost in this conversion.
- Sampling period $T_\mathrm{s}$, sampling frequency $f_\mathrm{s} = 1/T_\mathrm{s}$.
- Nyquist frequency.

# Time Sampling

## Continuous to Discrete Time

$$V_n = V_c(nT_s)$$

$$V_s = V_c(t) \sum_{n=-\infty}^{\infty} \delta(t - nT_s) = \sum_{n=-\infty}^{\infty} V_n \delta(t - nT_s)$$

- Multiply the signal by a train of delta functions.
- Multiplication in time domain means convolution in frequency domain.
- Information is lost in this conversion.
- Sampling period $T_s$, sampling frequency $f_s = 1/T_s$.
- Nyquist frequency.

# Amplitude Quantization

## Quantizer Definition

The quantizer is a nonlinear system whose purpose is to transform the input sample $V_n$ into one of a finite set of prescribed values ($\hat{V}_n$).

- Uniform quantization with step size $\Delta$.
- Quantizing to a given number of bits $N_b$ in the digital representation.
- $\Delta = 2X_m/2^{N_b} = X_m/2^{N_b-1}$ where $X_m$ is the full-scale range of the quantizer.
- Example: in an 8-bit system there are 256 discrete levels. Signal quantization step is $X_m/128$.

# Amplitude Quantization

## Quantizer Definition

The quantizer is a nonlinear system whose purpose is to transform the input sample $V_n$ into one of a finite set of prescribed values ($\hat{V}_n$).

- Uniform quantization with step size $\Delta$.
- Quantizing to a given number of bits $N_b$ in the digital representation.
- $\Delta = 2X_m/2^{N_b} = X_m/2^{N_b-1}$ where $X_m$ is the full-scale range of the quantizer.
- Example: in an 8-bit system there are 256 discrete levels. Signal quantization step is $X_m/128$.

# Amplitude Quantization

## Quantizer Definition

The quantizer is a nonlinear system whose purpose is to transform the input sample $V_n$ into one of a finite set of prescribed values ($\hat{V}_n$).

- Uniform quantization with step size $\Delta$.
- Quantizing to a given number of bits $N_b$ in the digital representation.
- $\Delta = 2X_m/2^{N_b} = X_m/2^{N_b - 1}$ where $X_m$ is the full-scale range of the quantizer.
- Example: in an 8-bit system there are 256 discrete levels. Signal quantization step is $X_m/128$.

# Quantization Errors

- Consider error $e_n = \hat{V}_n - V_n$
- $-\Delta/2 < e_n \leq \Delta/2$
- Assumptions:
  - The error sequence $e_n$ is a sample sequence of a stationary random process.
  - The error sequence is uncorrelated with the sequence $V_n$.
  - The error is a white-noise process.
  - The probability distribution of the error process is uniform over the range of quantization error.
- Then we get for variance of $e_n$: $\sigma_e^2 = \Delta^2/12$
- SNR of a quantizer in dB:
  $\mathrm{SNR} = 6.02N_b + 4.78 - 20\log_{10}(X_m/\sigma_V)$

# Quantization Errors

- Consider error $e_n = \hat{V}_n - V_n$
- $-\Delta/2 < e_n \leq \Delta/2$
- Assumptions:
  - The error sequence $e_n$ is a sample sequence of a stationary random process.
  - The error sequence is uncorrelated with the sequence $V_n$.
  - The error is a white-noise process.
  - The probability distribution of the error process is uniform over the range of quantization error.
- Then we get for variance of $e_n$: $\sigma_e^2 = \Delta^2/12$
- SNR of a quantizer in dB:
  $\mathrm{SNR} = 6.02 N_b + 4.78 - 20 \log_{10}(X_m/\sigma_V)$

# Quantization Errors

- Consider error $e_n = \hat{V}_n - V_n$
- $-\Delta/2 < e_n \leq \Delta/2$
- Assumptions:
  - The error sequence $e_n$ is a sample sequence of a stationary random process.
  - The error sequence is uncorrelated with the sequence $V_n$.
  - The error is a white-noise process.
  - The probability distribution of the error process is uniform over the range of quantization error.
- Then we get for variance of $e_n$: $\sigma_e^2 = \Delta^2/12$
- SNR of a quantizer in dB:
  $\mathrm{SNR} = 6.02N_b + 4.78 - 20\log_{10}(X_m/\sigma_V)$

# Quantization Errors

- Consider error $e_n = \hat{V}_n - V_n$
- $-\Delta/2 < e_n \leq \Delta/2$
- Assumptions:
  - The error sequence $e_n$ is a sample sequence of a stationary random process.
  - The error sequence is uncorrelated with the sequence $V_n$.
  - The error is a white-noise process.
  - The probability distribution of the error process is uniform over the range of quantization error.
- Then we get for variance of $e_n$: $\sigma_e^2 = \Delta^2/12$
- SNR of a quantizer in dB:
  $\mathrm{SNR} = 6.02N_b + 4.78 - 20\log_{10}(X_m/\sigma_V)$

# Digital to Analog Conversion

- Use samples to reconstruct continuous-time signal.
- Different ways to perform reconstruction:
  - Hold sample value for each period (zero-order hold);
  - Linearly interpolate between samples (first-order hold);
  - Many other methods.
- Typically D/A converters use zero-order hold.
- Frequency response of a zero-order hold is $H_0(j\omega) = \frac{2sin(\omega T_s/2)}{\omega T_s} e^{-i\omega T_s/2}$

# Digital to Analog Conversion

- Use samples to reconstruct continuous-time signal.
- Different ways to perform reconstruction:
  - Hold sample value for each period (zero-order hold);
  - Linearly interpolate between samples (first-order hold);
  - Many other methods.
- Typically D/A converters use zero-order hold.
- Frequency response of a zero-order hold is $H_0(j\omega) = \frac{2sin(\omega T_s/2)}{\omega T_s} e^{-i\omega T_s/2}$

# Digital to Analog Conversion

- Use samples to reconstruct continuous-time signal.
- Different ways to perform reconstruction:
  - Hold sample value for each period (zero-order hold);
  - Linearly interpolate between samples (first-order hold);
  - Many other methods.
- Typically D/A converters use zero-order hold.
- Frequency response of a zero-order hold is $H_0(j\omega) = \frac{2 sin(\omega T_s/2)}{\omega T_s} e^{-i\omega T_s/2}$

# Outline

1. Introduction to DSP
   - Defining the Terms
   - Sampling and Quantization
   - Z-transform
   - Digital Filtering
   - Efficient Filter Structures
2. Real-time digital signal processing
   - Definition and applications
   - Available solutions
3. Advantages and disadvantages
   - General-purpose Processors
   - Special-purpose DSP chips
   - Field Programmable Gate Arrays

# Z-transform

## Z-transform Definition

Z-transform of sequence $x_n$ is defined as
$X(z) = \sum_{n=-\infty}^{\infty} x_n z^{-n}$

- Discrete-time Fourier transform of sequence $x_n$ is $X(e^{i\omega}) = \sum_{n=-\infty}^{\infty} x_n e^{-i\omega n}$
- Similar to Laplace and Fourier transforms in continuous time we have z-transform and discrete-time Fourier transform.
- Delay operator $z^{-1}$: $x_{n-1} = x_n z^{-1}$

# Z-transform

## Z-transform Definition

Z-transform of sequence $x_n$ is defined as
$X(z) = \sum_{n=-\infty}^{\infty} x_n z^{-n}$

- Discrete-time Fourier transform of sequence $x_n$ is $X(e^{i\omega}) = \sum_{n=-\infty}^{\infty} x_n e^{-i\omega n}$
- Similar to Laplace and Fourier transforms in continuous time we have z-transform and discrete-time Fourier transform.
- Delay operator $z^{-1}$: $x_{n-1} = x_n z^{-1}$

# Z-transform

## Z-transform Definition

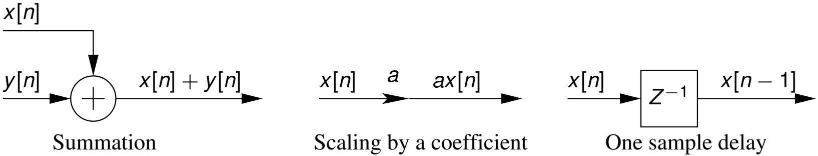
Z-transform of sequence $x_n$ is defined as
$X(z) = \sum_{n=-\infty}^{\infty} x_n z^{-n}$

- Discrete-time Fourier transform of sequence $x_n$ is $X(e^{i\omega}) = \sum_{n=-\infty}^{\infty} x_n e^{-i\omega n}$
- Similar to Laplace and Fourier transforms in continuous time we have z-transform and discrete-time Fourier transform.
- Delay operator $z^{-1}$: $x_{n-1} = x_n z^{-1}$

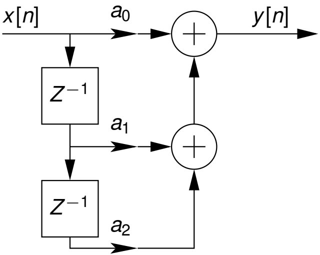
# Outline

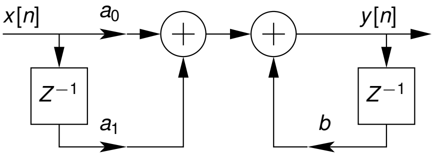
1. Introduction to DSP
   - Defining the Terms
   - Sampling and Quantization
   - Z-transform
   - Digital Filtering
   - Efficient Filter Structures
2. Real-time digital signal processing
   - Definition and applications
   - Available solutions
3. Advantages and disadvantages
   - General-purpose Processors
   - Special-purpose DSP chips
   - Field Programmable Gate Arrays

# Digital Filtering Basics




Legend



# Two Classes of Filters

- All linear time-invariant digital filters can be split into two classes:
  - Finite Impulse Response (FIR): filter output depends only on a finite number of past input samples;
  - Infinite Impulse Response (IIR): filter has internal memory, output theoretically persists to infinity.
- Internal memory — feedback.
- Feedback can be unstable — IIR filter designer has to worry about stability.
- FIR filters are unconditionally stable.

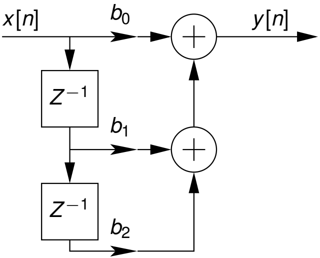
# Two Classes of Filters

- All linear time-invariant digital filters can be split into two classes:
  - Finite Impulse Response (FIR): filter output depends only on a finite number of past input samples;
  - Infinite Impulse Response (IIR): filter has internal memory, output theoretically persists to infinity.
- Internal memory — feedback.
- Feedback can be unstable — IIR filter designer has to worry about stability.
- FIR filters are unconditionally stable.

# Two Classes of Filters

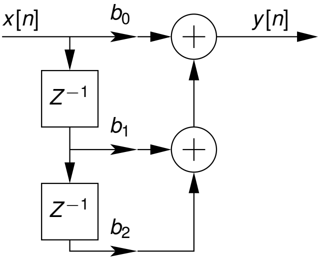
- All linear time-invariant digital filters can be split into two classes:
  - Finite Impulse Response (FIR): filter output depends only on a finite number of past input samples;
  - Infinite Impulse Response (IIR): filter has internal memory, output theoretically persists to infinity.
- Internal memory — feedback.
- Feedback can be unstable — IIR filter designer has to worry about stability.
- FIR filters are unconditionally stable.

# FIR Filter



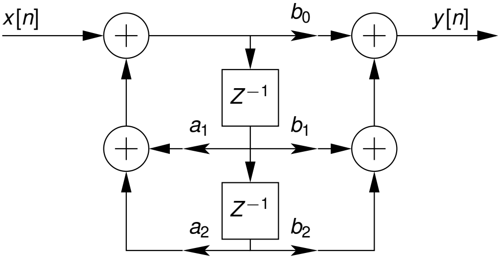

- Response of an FIR:
  $y[n] = \sum_{i=0}^{N-1} b_i x[N-1-i]$
- Each term in the sum is called "tap".
- $N$-tap filter requires $N$ multiplies and $N$ adds.
- Z-transform of FIR response:
  $H(z) = \sum_{i=0}^{N-1} b_i z^{-i}$

# FIR Filter



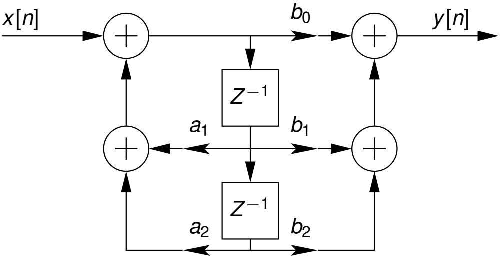

- Response of an FIR: $y[n] = \sum_{i=0}^{N-1} b_i x[N-1-i]$
- Each term in the sum is called "tap".
- $N$-tap filter requires $N$ multiplies and $N$ adds.
- Z-transform of FIR response: $H(z) = \sum_{i=0}^{N-1} b_i z^{-i}$

# IIR Filter: Biquad Structure



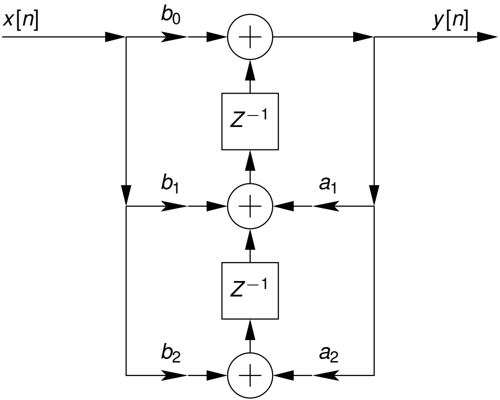

- Direct Form II realization
- Second-order transfer function
- $H(z) = \frac{b_0+b_1 z^{-1}+b_2 z^{-2}}{1+a_1 z^{-1}+a_2 z^{-2}} = \frac{b_0 z^2+b_1 z+b_2}{z^2+a_1 z+a_2}$

# IIR Filter: Biquad Structure



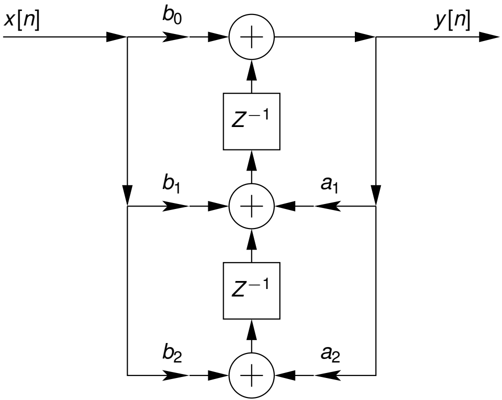

- Direct Form II realization
- Second-order transfer function
- $H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} = \frac{b_0 z^2 + b_1 z + b_2}{z^2 + a_1 z + a_2}$

# IIR Filter: Transposed




- Transposed Direct Form II realization
- $H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} = \frac{b_0 z^2 + b_1 z + b_2}{z^2 + a_1 z + a_2}$

# IIR Filter: Transposed




- Transposed Direct Form II realization
- $H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} = \frac{b_0 z^2 + b_1 z + b_2}{z^2 + a_1 z + a_2}$

# IIR Filter Stability

- Z-domain transfer function is stable if the poles (roots of the denominator polynomial) are within a unit circle.
- $|p| < 1$
- Critically stable for $|p| = 1$.
- Integrator is critically stable: $y_n = y_{n-1} + x_n$.

# IIR Filter Stability

- Z-domain transfer function is stable if the poles (roots of the denominator polynomial) are within a unit circle.
- $|p| < 1$
- Critically stable for $|p| = 1$.
- Integrator is critically stable: $y_n = y_{n-1} + x_n$.

# IIR Filter Stability

- Z-domain transfer function is stable if the poles (roots of the denominator polynomial) are within a unit circle.
- $|p| < 1$
- Critically stable for $|p| = 1$.
- Integrator is critically stable: $y_n = y_{n-1} + x_n$.

# Outline

1. Introduction to DSP
   - Defining the Terms
   - Sampling and Quantization
   - Z-transform
   - Digital Filtering
   - Efficient Filter Structures

2. Real-time digital signal processing
   - Definition and applications
   - Available solutions

3. Advantages and disadvantages
   - General-purpose Processors
   - Special-purpose DSP chips
   - Field Programmable Gate Arrays

# Good Filters

- Structures for efficient filter implementation:
  - Resource usage — no multiplies;
  - Resource usage — many zero coefficients;
  - Resource usage — symmetric structures;
  - Improving quantization effects.
- A few examples
- Cascaded Integrator Comb (CIC)
- Half-band filters
- Lattice structures

# Good Filters

- Structures for efficient filter implementation:
  - Resource usage — no multiplies;
  - Resource usage — many zero coefficients;
  - Resource usage — symmetric structures;
  - Improving quantization effects.
- A few examples
- Cascaded Integrator Comb (CIC)
- Half-band filters
- Lattice structures

# Good Filters

- Structures for efficient filter implementation:
  - Resource usage — no multiplies;
  - Resource usage — many zero coefficients;
  - Resource usage — symmetric structures;
  - Improving quantization effects.
- A few examples
- Cascaded Integrator Comb (CIC)
- Half-band filters
- Lattice structures

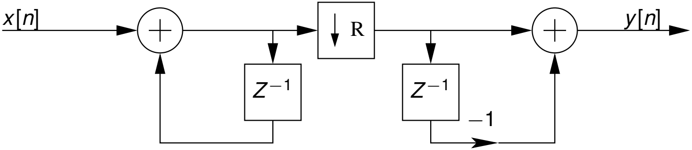
# Good Filters

- Structures for efficient filter implementation:
  - Resource usage — no multiplies;
  - Resource usage — many zero coefficients;
  - Resource usage — symmetric structures;
  - Improving quantization effects.
- A few examples
- Cascaded Integrator Comb (CIC)
- Half-band filters
- Lattice structures

# Good Filters

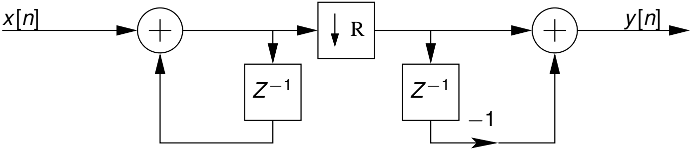
- Structures for efficient filter implementation:
  - Resource usage — no multiplies;
  - Resource usage — many zero coefficients;
  - Resource usage — symmetric structures;
  - Improving quantization effects.
- A few examples
- Cascaded Integrator Comb (CIC)
- Half-band filters
- Lattice structures

# Cascaded Integrator Comb




- Sampling rate reduced by $R$.
- $y_n = \sum_{i=0}^{R-1} x[n-i]$
- What about integrator overflow?

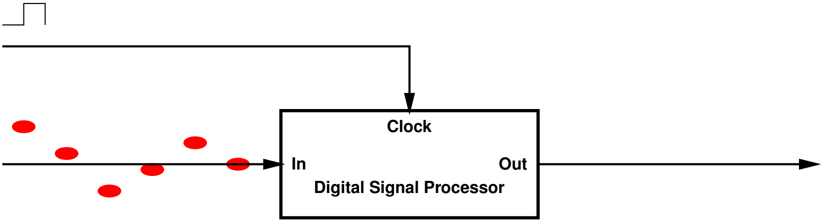
# Cascaded Integrator Comb




- Sampling rate reduced by $R$.
- $y_n = \sum_{i=0}^{R-1} x[n-i]$
- What about integrator overflow?

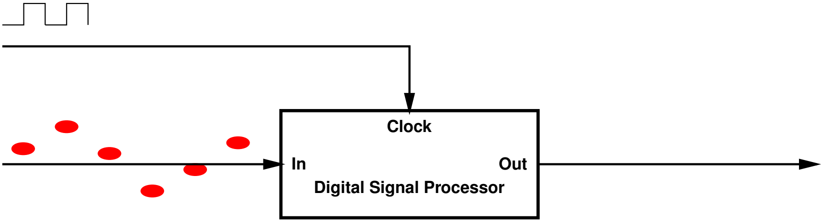
# Outline

# Real-time Signal Processing Definition



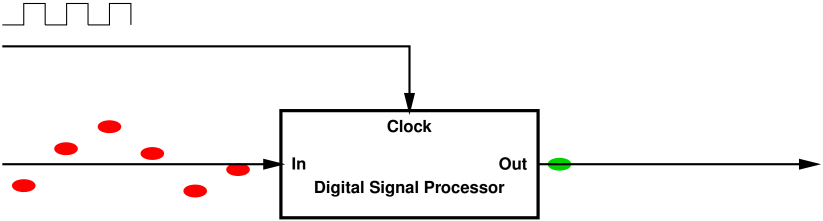

- Continuous input stream of samples.
- Output (processed) samples generated every clock cycle.
- Fixed delay (latency) between input and output.
- Sampling clock defines available per-sample processing time.
- System defining elements: sampling rate, latency, algorithm complexity.

# Real-time Signal Processing Definition



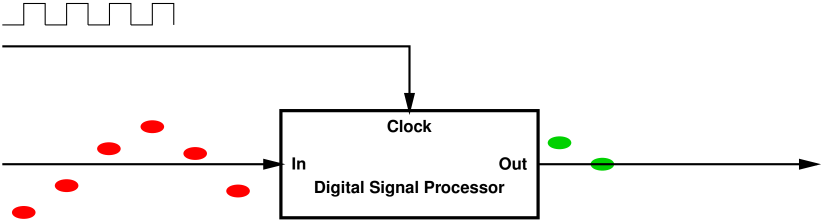

- Continuous input stream of samples.
- Output (processed) samples generated every clock cycle.
- Fixed delay (latency) between input and output.
- Sampling clock defines available per-sample processing time.
- System defining elements: sampling rate, latency, algorithm complexity.

# Real-time Signal Processing Definition



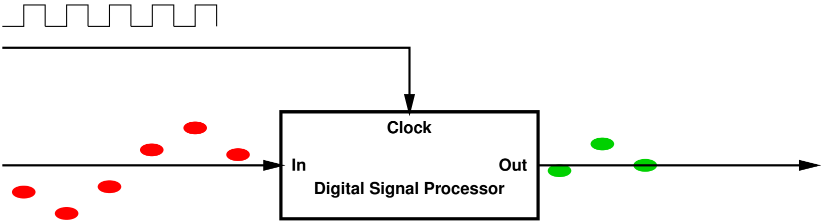

- Continuous input stream of samples.
- Output (processed) samples generated every clock cycle.
- Fixed delay (latency) between input and output.
- Sampling clock defines available per-sample processing time.
- System defining elements: sampling rate, latency, algorithm complexity.

# Real-time Signal Processing Definition



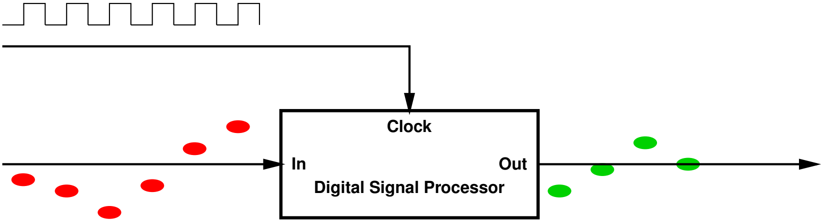

- Continuous input stream of samples.
- Output (processed) samples generated every clock cycle.
- Fixed delay (latency) between input and output.
- Sampling clock defines available per-sample processing time.
- System defining elements: sampling rate, latency, algorithm complexity.

# Real-time Signal Processing Definition



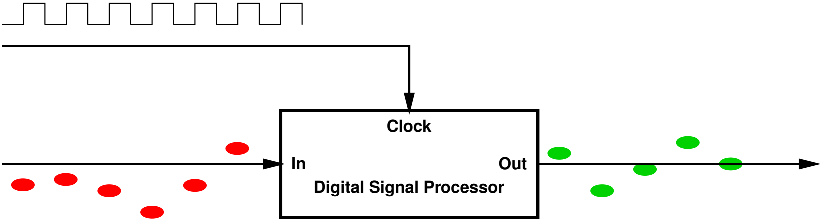

- Continuous input stream of samples.
- Output (processed) samples generated every clock cycle.
- Fixed delay (latency) between input and output.
- Sampling clock defines available per-sample processing time.
- System defining elements: sampling rate, latency, algorithm complexity.

# Real-time Signal Processing Definition



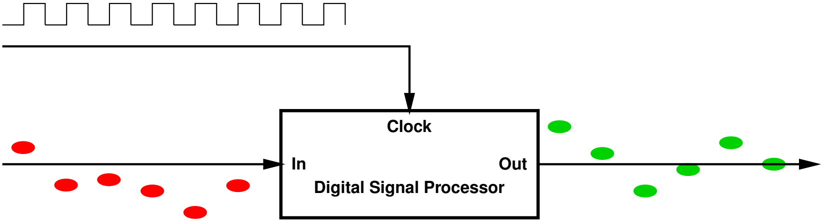

- Continuous input stream of samples.
- Output (processed) samples generated every clock cycle.
- Fixed delay (latency) between input and output.
- Sampling clock defines available per-sample processing time.
- System defining elements: sampling rate, latency, algorithm complexity.

# Real-time Signal Processing Definition




- Continuous input stream of samples.
- Output (processed) samples generated every clock cycle.
- Fixed delay (latency) between input and output.
- Sampling clock defines available per-sample processing time.
- System defining elements: sampling rate, latency, algorithm complexity.

# Real-time Signal Processing Definition




- Continuous input stream of samples.
- Output (processed) samples generated every clock cycle.
- Fixed delay (latency) between input and output.
- Sampling clock defines available per-sample processing time.
- System defining elements: sampling rate, latency, algorithm complexity.

# Accelerator Applications

- Real-time signal processing.
  - Low-level RF;
  - Orbit feedback;
  - Collision point feedback;
  - Coupled-bunch instabilities control;
  - BPMs
- There are also non real-time needs:
  - Off-line diagnostics …
  - …and configuration
  - These are often easier to satisfy with the off-the-shelf hardware.

# Accelerator Applications

- Real-time signal processing.
  - Low-level RF;
  - Orbit feedback;
  - Collision point feedback;
  - Coupled-bunch instabilities control;
  - BPMs
- There are also non real-time needs:
  - Off-line diagnostics . . .
  - . . . and configuration
  - These are often easier to satisfy with the off-the-shelf hardware.

# Accelerator Applications

- Real-time signal processing.
  - Low-level RF;
  - Orbit feedback;
  - Collision point feedback;
  - Coupled-bunch instabilities control;
  - BPMs
- There are also non real-time needs:
  - Off-line diagnostics . . .
  - . . . and configuration
  - These are often easier to satisfy with the off-the-shelf hardware.

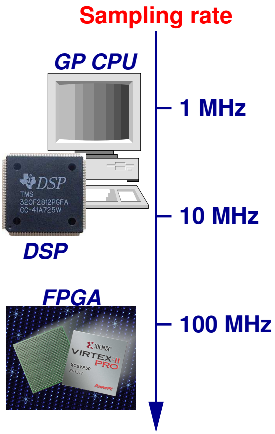
# Accelerator Applications

- Real-time signal processing.
  - Low-level RF;
  - Orbit feedback;
  - Collision point feedback;
  - Coupled-bunch instabilities control;
  - BPMs
- There are also non real-time needs:
  - Off-line diagnostics . . .
  - . . . and configuration
  - These are often easier to satisfy with the off-the-shelf hardware.

# Outline

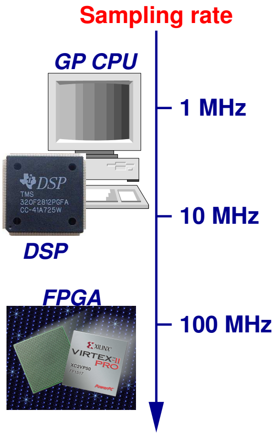
1. Introduction to DSP
   - Defining the Terms
   - Sampling and Quantization
   - Z-transform
   - Digital Filtering
   - Efficient Filter Structures
2. Real-time digital signal processing
   - Definition and applications
   - Available solutions
3. Advantages and disadvantages
   - General-purpose Processors
   - Special-purpose DSP chips
   - Field Programmable Gate Arrays

# Real-time Signal Processing Solutions: Sampling Rate



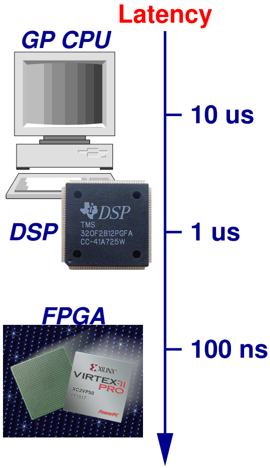

- Sampling rates of interest from 100 kHz to 1000+ MHz.
- Options range from general-purpose CPUs to dedicated hardware.
- Special-purpose DSP chips fall somewhere in the middle.
- Are DSPs really faster than GP CPUs?
- Dedicated hardware solutions have mostly converged on FPGA devices.

# Real-time Signal Processing Solutions: Sampling Rate




- Sampling rates of interest from 100 kHz to 1000+ MHz.
- Options range from general-purpose CPUs to dedicated hardware.
- Special-purpose DSP chips fall somewhere in the middle.
- Are DSPs really faster than GP CPUs?
- Dedicated hardware solutions have completely converged on FPGA devices.

# Real-time Signal Processing Solutions: Latency




- Finishing order quite similar to the previous slide.
- For latency DSPs do have an edge on the general-purpose CPUs.

# Three Choices

- General-purpose processors:
  - Basically a plain-vanilla Intel-architecture PC.
  - Instruction rates in the multi-GHz range.
  - Hierarchical memory structure complicates algorithm timing.
- Special-purpose DSPs.
  - Off-the-shelf or custom design.
  - Slower clocks than GP CPUs.
  - Multiple execution units.
  - Architectural features for real-time processing.
- FPGAs
  - Most likely custom design, some off-the-shelf availability.
  - Highly parallel.
  - Sample processing rates into hundreds of MHz.

# Three Choices

- General-purpose processors:
  - Basically a plain-vanilla Intel-architecture PC.
  - Instruction rates in the multi-GHz range.
  - Hierarchical memory structure complicates algorithm timing.
- Special-purpose DSPs.
  - Off-the-shelf or custom design.
  - Slower clocks than GP CPUs.
  - Multiple execution units.
  - Architectural features for real-time processing.
- FPGAs
  - Most likely custom design, some off-the-shelf availability.
  - Highly parallel.
  - Sample processing rates into hundreds of MHz.

# Three Choices

- General-purpose processors:
  - Basically a plain-vanilla Intel-architecture PC.
  - Instruction rates in the multi-GHz range.
  - Hierarchical memory structure complicates algorithm timing.
- Special-purpose DSPs.
  - Off-the-shelf or custom design.
  - Slower clocks than GP CPUs.
  - Multiple execution units.
  - Architectural features for real-time processing.
- FPGAs
  - Most likely custom design, some off-the-shelf availability.
  - Highly parallel.
  - Sample processing rates into hundreds of MHz.

# Outline

# General-purpose CPUs: advantages

- Low cost per MIPS.
- Wide variety of development tools/environments.
- Easy to prototype and test algorithms.
- Intel/AMD CPUs have DSP extensions:
    - MMX, MMX2, SSE, SSE2, . . .

# General-purpose CPUs: disadvantages

- Real-time support issues.
- Input and output.
  - Real-time streaming I/O needs thought.
- Integration:
  - Startup and booting.
  - Power interruption handling.
  - Software maintenance.

# Outline

# DSPs: advantages and disadvantages.

Advantages.

- Geared for real-time processing.
- Special instructions for filtering, Fourier transforms.

Disadvantages.

- General-purpose CPUs include DSP engines
- It is doubtful that DSPs have any speed edge at this time

# DSPs: advantages and disadvantages.

Advantages.

- Geared for real-time processing.
- Special instructions for filtering, Fourier transforms.

Disadvantages.

- General-purpose CPUs include DSP engines
- It is doubtful that DSPs have any speed edge at this time

# Outline

# FPGAs: Pros and Cons

Pros:

- Natural for synchronous real-time processing
- Parallel structures provide significant speed gain
    - Each clock cycle multiple processing units execute simultaneously
    - Example: 64-tap FIR at 100 MHz
    - Equivalent to 6.4 GHz instruction rate on a single execution unit.
- Can use soft CPUs or on-chip cores for housekeeping, startup sequences, adaptation.

Cons:

- Custom design likely required.
- FPGAs are better suited to relatively simple processing structures.

# FPGAs: Pros and Cons

Pros:

- Natural for synchronous real-time processing
- Parallel structures provide significant speed gain
  - Each clock cycle multiple processing units execute simultaneously
  - Example: 64-tap FIR at 100 MHz
  - Equivalent to 6.4 GHz instruction rate on a single execution unit.
- Can use soft CPUs or on-chip cores for housekeeping, startup sequences, adaptation.

Cons:

- Custom design likely required.
- FPGAs are better suited to relatively simple processing structures.

# FPGAs: Pros and Cons

Pros:

- Natural for synchronous real-time processing
- Parallel structures provide significant speed gain
  - Each clock cycle multiple processing units execute simultaneously
  - Example: 64-tap FIR at 100 MHz
  - Equivalent to 6.4 GHz instruction rate on a single execution unit.
- Can use soft CPUs or on-chip cores for housekeeping, startup sequences, adaptation.

Cons:

- Custom design likely required.
- FPGAs are better suited to relatively simple processing structures.

# FPGAs: Pros and Cons

Pros:

- Natural for synchronous real-time processing
- Parallel structures provide significant speed gain
  - Each clock cycle multiple processing units execute simultaneously
  - Example: 64-tap FIR at 100 MHz
  - Equivalent to 6.4 GHz instruction rate on a single execution unit.
- Can use soft CPUs or on-chip cores for housekeeping, startup sequences, adaptation.

Cons:

- Custom design likely required.
- FPGAs are better suited to relatively simple processing structures.

# A Possible Design Philosophy

- Design resources are pretty much always limited.
- Pragmatically look for minimalistic solutions . . .
- . . . without sacrificing functionality.
- Optimal solution strongly depends on the skills available.
  - Custom hardware with minimal software for UI and diagnostics.
  - Off-the-shelf DSP system with soft processing.
- Technology choice is not necessarily driven by the technical merits.

# A Possible Design Philosophy

- Design resources are pretty much always limited.
- Pragmatically look for minimalistic solutions . . .
- . . . without sacrificing functionality.
- Optimal solution strongly depends on the skills available.
  - Custom hardware with minimal software for UI and diagnostics.
  - Off-the-shelf DSP system with soft processing.
- Technology choice is not necessarily driven by the technical merits.

# A Possible Design Philosophy

- Design resources are pretty much always limited.
- Pragmatically look for minimalistic solutions . . .
- . . . without sacrificing functionality.
- Optimal solution strongly depends on the skills available.
  - Custom hardware with minimal software for UI and diagnostics.
  - Off-the-shelf DSP system with soft processing.
- Technology choice is not necessarily driven by the technical merits.

# Summary

- Introduction to DSP: sampling, noise, filtering.
- Some hardware/software implementation ideas.
- You really learn by implementing the structures!!!

# Summary

- Introduction to DSP: sampling, noise, filtering.
- Some hardware/software implementation ideas.
- You really learn by implementing the structures!!!

# Summary

- Introduction to DSP: sampling, noise, filtering.
- Some hardware/software implementation ideas.
- You really learn by implementing the structures!!!