Waveforms

Time-varying scalar value

Commonly called a “signal” in the control-theory literature

Sound: air pressure over time

\[ f(t) \]

Raster video: brightness over time

Speed over time, position over time, etc.
The Fourier Series

Any periodic function can be expressed as a sum of harmonics. For a smooth function $f(t)$ with period $T$, i.e.,

$$f(t) = f(t + T),$$

there exists coefficients $a_n$, $b_n$ such that

$$f(t) = a_0 + \sum_{m=1}^{\infty} a_m \cos \frac{2\pi mt}{T} + b_m \sin \frac{2\pi mt}{T}$$
The Fourier Series for a Square Wave

\[ f(t) = \frac{4}{\pi} \left( \sin t - \frac{1}{3} \sin 3t + \frac{1}{5} \sin 5t - \frac{1}{7} \sin 7t + \frac{1}{9} \sin 9t + \cdots \right) \]
The Fourier Series for a Square Wave

\[ f(t) = \frac{4}{\pi} \left( \sin t + \frac{\sin 3t}{3} \right) \]
The Fourier Series for a Square Wave

\[ f(t) = \frac{4}{\pi} \left( \sin t + \frac{\sin 3t}{3} + \frac{\sin 5t}{5} + \cdots \right) \]
The Fourier Series for a Square Wave

\[ f(t) = \frac{4}{\pi} \left( \sin t + \frac{\sin 3t}{3} + \frac{\sin 5t}{5} + \frac{\sin 7t}{7} + \cdots \right) \]
The Fourier Series for a Square Wave

\[ f(t) = \frac{4}{\pi} \left( \sin t + \frac{\sin 3t}{3} + \frac{\sin 5t}{5} + \frac{\sin 7t}{7} + \frac{\sin 9t}{9} \right) \]
The Fourier Series for a Square Wave

\[ f(t) = \frac{4}{\pi} \left( \sin t + \frac{\sin 3t}{3} + \frac{\sin 5t}{5} + \frac{\sin 7t}{7} + \frac{\sin 9t}{9} + \cdots \right) \]
Bandwidth-Limited Signals

Basic observation: nothing changes infinitely fast

Bounding the rate of change sets the bandwidth of a signal

Hertz or Hz: “per second”

Source: Stereophile magazine: Marantz SM-11S1, A $4000 audiophile amplifier rated 5 Hz–120 kHz. Small-signal 10kHz squarewave into 8 ohms.
The Bandwidth of Sound

Human ears are almost a Fourier transform
Empirically, humans hear 20 Hz–20 kHz. Highest frequency limit tends to decrease with age.
Nyquist Theorem

To reconstruct a bandwidth-limited signal from samples, you need to sample at least twice the maximum frequency.

Sampling at $8 \times f$
Nyquist Theorem

To reconstruct a bandwidth-limited signal from samples, you need to sample at least twice the maximum frequency.

Sampling at $4 \times f$
Nyquist Theorem

To reconstruct a bandwidth-limited signal from samples, you need to sample at least twice the maximum frequency.

Sampling at $2 \times f$
Nyquist Theorem

To reconstruct a bandwidth-limited signal from samples, you need to sample at least twice the maximum frequency.

Sampling at $4/3 \times f$
Nyquist Theorem

To reconstruct a bandwidth-limited signal from samples, you need to sample at least twice the maximum frequency.

Sampling at $1 \times f$
Audio Sampling Rates

CD-quality audio: 44.1 kHz

Telephone-quality audio: 8 kHz
Signal-to-Noise Ratio

You can’t always get what you want
But if you try sometimes you might find
You get what you need —The Rolling Stones

Signals are never pure: there’s always something that makes them deviate from the ideal.

Signal-to-Noise ratio:

\[ SNR = \frac{\text{Signal Power}}{\text{Noise Power}} \]

Usually measured using a log scale, i.e.,

\[ dB = 10 \log_{10} \frac{P_{\text{signal}}}{P_{\text{noise}}} \]
Human Hearing dB, SNR, and bits

\[ n \times 6.02 + 1.76 = \text{SNR in dB} \]

CD samples:

16 bits = 98 dB

Near the limit of human hearing
The CODEC on the DE1-SoC: Wolfson WM8731

enCOder/DECoder: analog-to-digital converter (ADC) + digital-to-analog converter (DAC)

Two 24-bit ADCs; two 24-bit DACs + headphone amp

Sampling rates: 8 kHz – 96 kHz, 16–24 bit words
## DE1-SoC Interface to the Audio Codec

### Pin Assignment of Audio CODEC

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>FPGA Pin No.</th>
<th>Description</th>
<th>I/O Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUD_ADCLRCK</td>
<td>PIN_K8</td>
<td>Audio CODEC ADC LR Clock</td>
<td>3.3V</td>
</tr>
<tr>
<td>AUD_ADCDAT</td>
<td>PIN_K7</td>
<td>Audio CODEC ADC Data</td>
<td>3.3V</td>
</tr>
<tr>
<td>AUD_DACLRCK</td>
<td>PIN_H8</td>
<td>Audio CODEC DAC LR Clock</td>
<td>3.3V</td>
</tr>
<tr>
<td>AUD_DACDAT</td>
<td>PIN_J7</td>
<td>Audio CODEC DAC Data</td>
<td>3.3V</td>
</tr>
<tr>
<td>AUD_XCK</td>
<td>PIN_G7</td>
<td>Audio CODEC Chip Clock</td>
<td>3.3V</td>
</tr>
<tr>
<td>AUD_BCLK</td>
<td>PIN_H7</td>
<td>Audio CODEC Bit-stream Clock</td>
<td>3.3V</td>
</tr>
<tr>
<td>I2C_SCLK</td>
<td>PIN_J12 or PIN_E23</td>
<td>I2C Clock</td>
<td>3.3V</td>
</tr>
<tr>
<td>I2C_SDAT</td>
<td>PIN_K12 or PIN_C24</td>
<td>I2C Data</td>
<td>3.3V</td>
</tr>
</tbody>
</table>

**I²C bus for configuration:** data format, volume levels, etc.

**Synchronous serial protocol (data + L/R + bit clock) for data**
WM8731 Serial Protocol

One of four communication modes, set by I²C registers:

Figure 26 Left Justified Mode
## DE1-SoC I²C Multiplexer

Control mechanism for the I²C multiplexer

**Pin Assignment of I²C Bus**

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>FPGA Pin No.</th>
<th>Description</th>
<th>I/O Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGA_I2C_SCLK</td>
<td>PIN_J12</td>
<td>FPGA I²C Clock</td>
<td>3.3V</td>
</tr>
<tr>
<td>FPGA_I2C_SDAT</td>
<td>PIN_K12</td>
<td>FPGA I²C Data</td>
<td>3.3V</td>
</tr>
<tr>
<td>HPS_I2C1_SCLK</td>
<td>PIN_E23</td>
<td>I²C Clock of the first HPS I²C conroller</td>
<td>3.3V</td>
</tr>
<tr>
<td>HPS_I2C1_SDAT</td>
<td>PIN_C24</td>
<td>I²C Data of the first HPS I²C conroller</td>
<td>3.3V</td>
</tr>
<tr>
<td>HPS_I2C2_SCLK</td>
<td>PIN_H23</td>
<td>I²C Clock of the second HPS I²C conroller</td>
<td>3.3V</td>
</tr>
<tr>
<td>HPS_I2C2_SDAT</td>
<td>PIN_A25</td>
<td>I²C Data of the second HPS I²C conroller</td>
<td>3.3V</td>
</tr>
</tbody>
</table>
Storing Waveforms

If you store each sample,

\[
\frac{\text{samples}}{\text{second}} \times \frac{\text{bits}}{\text{sample}} \times \text{channels} = \frac{\text{bits}}{\text{second}}
\]

Total memory consumption:

\[
\frac{\text{bits}}{\text{seconds}} \times \text{seconds} = \text{bits}
\]

E.g., CD-quality audio: 44.1 kHz, 16 bits/sample, 2 channels

\[
44.1 \text{ kHz} \times 16 \times 2 = 1.4 \text{ Mbps} = 175 \text{ KB/s}
\]

A 74-minute CD:

\[
1.4 \text{ Mbps} \times 60 \frac{\text{seconds}}{\text{minute}} \times 74 \text{ minutes} \times \frac{\text{byte}}{8 \text{ bits}} = 783 \text{ MB}
\]
Reducing Memory: Sample Less; Use Fewer Bits

74 minutes of CD-quality audio

(16 bits/sample, stereo, 44.1 kHz)

\[
44.1 \text{ kHz} \times 32 \text{ bits} \times 60 \text{ sec/min} \times 74 \text{ min} \div 8 \text{ bits/byte} = 783 \text{ MB}
\]

74 minutes of telephone-quality audio:

(8 bits/sample, mono, 8 kHz)

\[
8 \text{ kHz} \times 8 \text{ bits} \times 60 \text{ sec/min} \times 74 \text{ min} \div 8 \text{ bits/byte} = 35 \text{ MB}
\]
Reducing Memory: Lossy Compression (Companding)

μ-law and A-law compression
Logarithmic encoding of 12 bit samples in 8 bits
Trades dynamic range for quantization noise

Source: Ozhiker, Wikimedia commons
ADPCM: Adaptive Predictive Pulse Code Modulation

Uses 4 bits/sample to reconstruct 8-bit samples

Encodes the difference between the next sample and its predicted value
MPEG Layer 3 Compression: Perceptual Coding

Carefully reproduce what we hear well and worry less about what we can’t (soft sounds masked by loud ones)
Sound Synthesis: Analog

Modular analog sound synthesis c. 1968
Oscillators + noise sources + envelope generators + filters

Moog synthesizer
Subtractive Synthesis

Start with a saw, square, or triangle wave, then filter
The AY-3-8912 Programmable Sound Generator
FM Synthesis

What does it sound like? Any pop music from the 1980s
Summary of Audio Waveform Generation

- Direct sampling (Pulse Code Modulation)
  Consider sampling frequency, bits/sample

- Lossy Compression
  Companding ($\mu$-law, A-law)
  ADPCM
  Perceptual Coding (MP3 et al.)

- Synthesis
  Subtractive (oscillators, filters, envelopes)
  FM (Carrier $\times$ modulator, envelopes)
  Wavetable/sampling (sound snippets + note events)
Representing Images

Same story; two dimensional waveforms

E.g., a single frame of VGA/standard definition television:

\[
640 \times 480 \times 24 \frac{\text{bits}}{\text{pixel}} = 900 \text{ KB}
\]

HD is terrifying:

\[
1920 \times 1080 \times 24 \frac{\text{bits}}{\text{pixel}} = 5.9 \text{ MB}
\]
JPEG: Still Image Compression

1010011

8 × 8 blocks

Huffman⁻¹

Zig-zag⁻¹

Quantize⁻¹

DCT⁻¹

YCbCr-to-RGB

Colorsace conversion

Space-to-frequency domain conversion

Quantization

Zig-zag encoding

Huffman encoding