Nyquist's Theorem and Sampling

Before we look at SDR and its various implementations in embedded systems, we'll review a theorem fundamental to sampled data systems such as those encountered in Software-Defined Radios.

Nyquist's Theorem:

"Any signal can be represented by discrete samples if the sampling frequency is at least twice the bandwidth of the signal."

Notice that we highlighted the word bandwidth rather than frequency. In what follows, we'll attempt to show the implications of this theorem and the correct interpretation of sampling frequency, also known as sampling rate.

A Simple Technique to Visualize Sampling

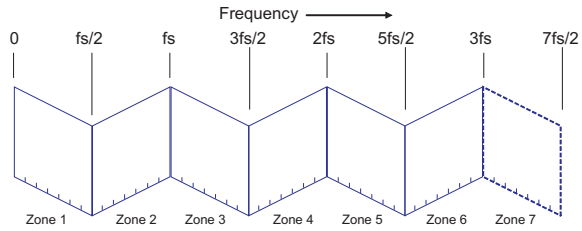


Figure 1

To visualize what happens in sampling, imagine that you are using transparent "fan-fold" computer paper. Use the horizontal edge of the paper as the frequency axis and scale it so that the paper folds line up with integer multiples of one-half of the sampling frequency f_s. Each sheet of paper now represent what we will call a "Nyquist Zone", as shown in Figure 1.

Pentek, Inc. • One Park Way, Upper Saddle River, NJ 07458 • Tel: (201) 818-5900 • Fax: (201) 818-5904 • Email: info@pentek.com • http://www.pentek.com

Sampling

Sampling Basics

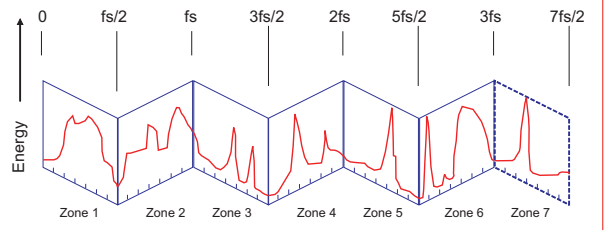


Figure 2

Use the vertical axis of the fan-fold paper for signal energy and plot the frequency spectrum of the signal to be sampled, as shown in Figure 2. To see the effects of sampling, collapse the transparent fan-fold paper into a stack.

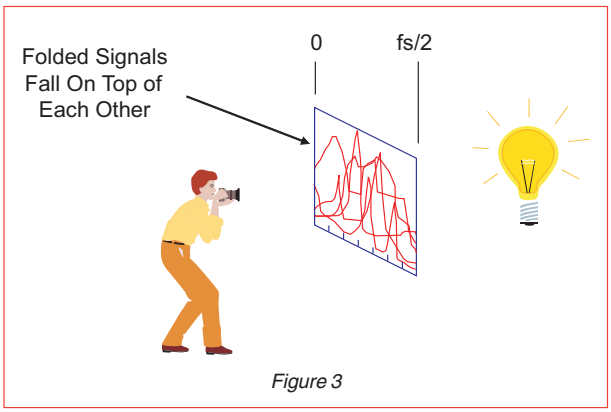


Figure 3

The resulting spectrum can be seen by holding the transparent stack up to a light and looking through it. You can see that signals on all of the sheets or zones are "folded" or "aliased" on top of each other — and they can no longer be separated.

Once this folding or aliasing occurs during sampling, the resulting sampled data is corrupted and can never be recovered. The term "aliasing" is appropriate because after sampling, a signal from one of the higher zones now appears to be at a different frequency.

Baseband Sampling

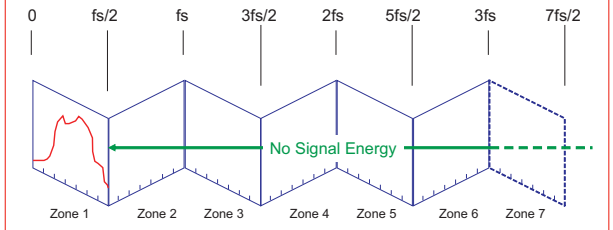


Figure 4

A baseband signal has frequency components that start at $f = 0$ and extend up to some maximum frequency.

To prevent data destruction when sampling a baseband signal, make sure that all the signal energy falls ONLY in the 1st Nyquist band, as shown in Figure 4.

There are two ways to do this:

1. Insert a lowpass filter to eliminate all signals above $f_s/2$, or
2. Increase the sampling frequency so all signals present fall below $f_s/2$.

Note that $f_s/2$ is also known as the "folding frequency".

Sampling Bandpass Signals

Let's consider bandpass signals like the IF frequency of a communications receiver that might have a 70 MHz center frequency and 10 MHz bandwidth. In this case, the IF signal contains signal energery from 65 to 75 MHz.

If we follow the baseband sampling rules above, we must sample this signal at twice the highest signal frequency, meaning a sample rate of at least 150 MHz.

However, by taking advantage of a technique called "undersampling", we can use a much lower sampling rate.

Undersampling

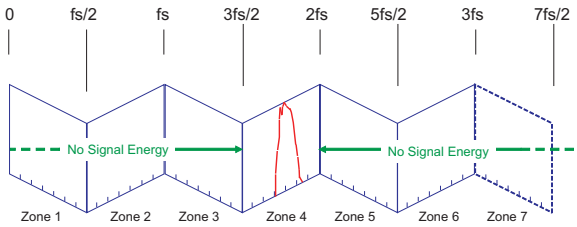


Figure 5

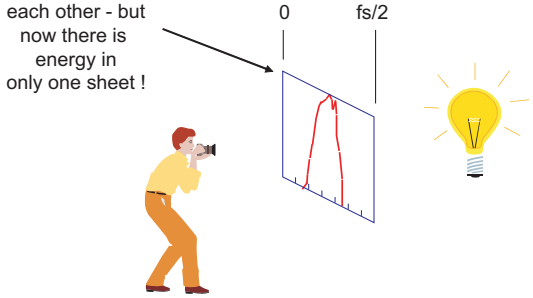


Figure 6

Undersampling allows us to use aliasing to our advantage, providing we follow the strict rules of the Nyquist Theorem.

In our previous IF signal example, suppose we try a sampling rate of 40 MHz.

Figure 5 shows a fan-fold paper plot with Fs = 40 MHz. You can see that zone 4 extends from 60 MHz to 80 MHz, nicely containing the entire IF signal band of 65 to 75 MHz.

Now when you collapse the fan fold sheets as shown in Figure 6, you can see that the IF signal is preserved after sampling because we have no signal energy in any other zone.

Also note that the odd zones fold with the lower frequency at the left (normal spectrum) and the even zones fold with the lower frequency at the right (reversed spectrum).

In this case, the signals from zone 4 are frequency-reversed. This is usually very easy to accommodate in the following stages of SDR systems.

The major rule to follow for successful undersampling is to make sure all of the energy falls entirely in one Nyquist zone.

There two ways to do this:

1. Insert a bandpass filter to eliminate all signals outside the one Nyquist zone.
2. Increase the sampling frequency so all signals fall entirely within one Nyquist zone.

Summary

Baseband sampling requires the sample frequency to be at least twice the signal bandwidth. This is the same as saying that all of the signals fall within the first Nyquist zone.

In real life, a good rule of thumb is to use the 80% relationship:

$$\text{Bandwidth} = 0.8 * f_s / 2 = 0.4 * f_s$$

Undersampling allows a lower sample rate even though signal frequencies are high, PROVIDED all of the signal energy falls within one Nyquist zone.

To repeat the Nyquist theorem: The sampling frequency must be at least twice the signal bandwidth — not the signal frequency.

Analog Radio Receiver Block Diagram

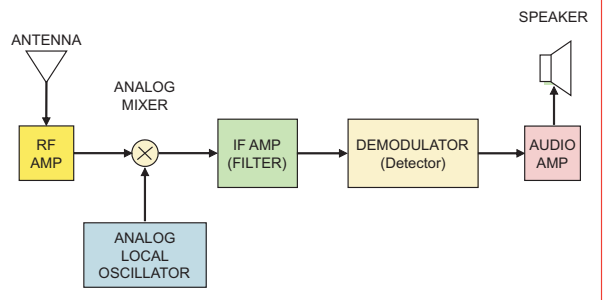


Figure 7

Analog Radio Receiver Mixer

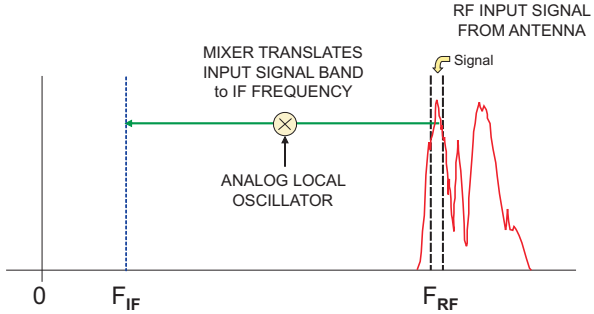


Figure 8

The conventional heterodyne radio receiver shown in Figure 7, has been in use for nearly a century. Let's review the structure of the analog receiver so comparison to a digital receiver becomes apparent.

First the RF signal from the antenna is amplified, typically with a tuned RF stage that amplifies a region of the frequency band of interest.

This amplified RF signal is then fed into a mixer stage. The other input to the mixer comes from the local oscillator whose frequency is determined by the tuning control of the radio.

The mixer translates the desired input signal to the IF (Intermediate Frequency) as shown in Figure 8.

The IF stage is a bandpass amplifier that only lets one signal or radio station through. Common center frequencies for IF stages are 455 kHz and 10.7 MHz for commercial AM and FM broadcasts.

The demodulator recovers the original modulating signal from the IF output using one of several different schemes.

For example, AM uses an envelope detector and FM uses a frequency discriminator. In a typical home radio, the demodulated output is fed to an audio power amplifier which drives a speaker.

The mixer performs an analog multiplication of the two inputs and generates a difference frequency signal.

The frequency of the local oscillator is set so that the difference between the local oscillator frequency and the desired input signal (the radio station you want to receive) equals the IF.

For example, if you wanted to receive an FM station at 100.7 MHz and the IF is 10.7 MHz, you would tune the local oscillator to:

$$100.7 - 10.7 = 90 \text{ MHz}$$

This is called "downconversion" or "translation" because a signal at a high frequency is shifted down to a lower frequency by the mixer.

The IF stage acts as a narrowband filter which only passes a "slice" of the translated RF input. The bandwidth of the IF stage is equal to the bandwidth of the signal (or the "radio station") that you are trying to receive.

For commercial FM, the bandwidth is about 100 kHz and for AM it is about 5 kHz. This is consistent with channel spacings of 200 kHz and 10 kHz, respectively.

SDR Receiver Block Diagram

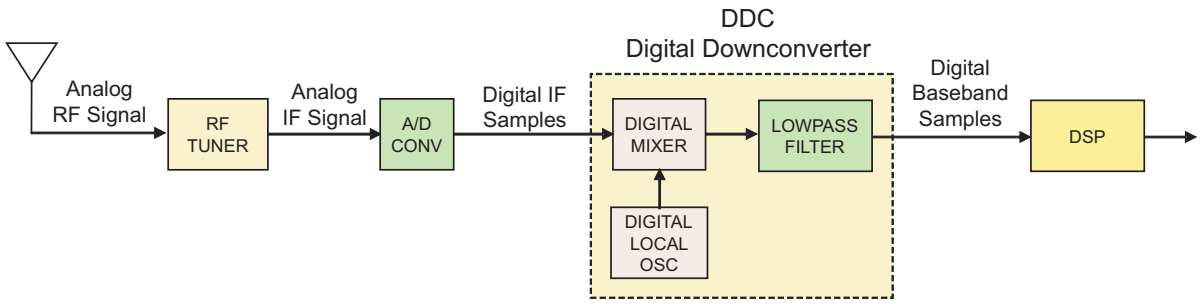


Figure 9

Figure 9 shows a block diagram of a software defined radio receiver. The RF tuner converts analog RF signals to analog IF frequencies, the same as the first three stages of the analog receiver.

The A/D converter that follows digitizes the IF signal thereby converting it into digital samples. These samples are fed to the next stage which is the digital downconverter (DDC) shown within the dotted lines.

The digital downconverter is typically a single monolithic chip or FPGA IP, and it is a key part of the SDR system.

A conventional DDC has three major sections:

- **A digital mixer**
- **A digital local oscillator**
- **An FIR lowpass filter**

The digital mixer and local oscillator translate the digital IF samples down to baseband. The FIR lowpass filter limits the signal bandwidth and acts as a decimating lowpass filter. The digital downconverter includes a lot of hardware multipliers, adders and shift register memories to get the job done.

The digital baseband samples are then fed to a block labeled DSP which performs tasks such as demodulation, decoding and other processing tasks.

Traditionally, these needs have been handled with dedicated application-specific ICs (ASICs), and programmable DSPs.

SDR Receiver Mixer

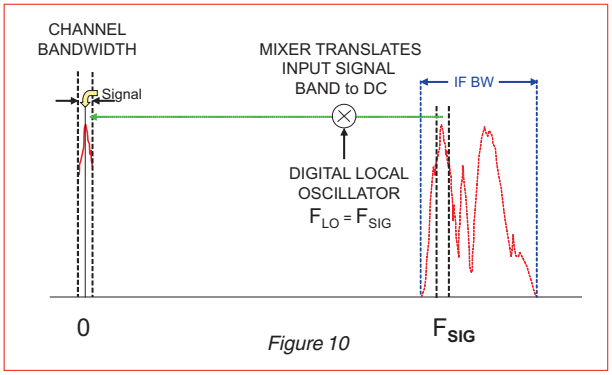


Figure 10

At the output of the mixer, the high frequency wideband signals from the A/D input (shown in Figure 10 above) have been translated down to DC as complex I and Q components with a frequency shift equal to the local oscillator frequency.

This is similar to the analog receiver mixer except *there*, the mixing was done down to an IF frequency. *Here*, the complex representation of the signal allows us to go right down to DC.

By tuning the local oscillator over its range, any portion of the RF input signal can be mixed down to DC.

In effect, the wideband RF signal spectrum can be "slid" around 0 Hz, left and right, simply by tuning the local oscillator. Note that upper and lower sidebands are preserved.

DDC Local Oscillator and Decimation

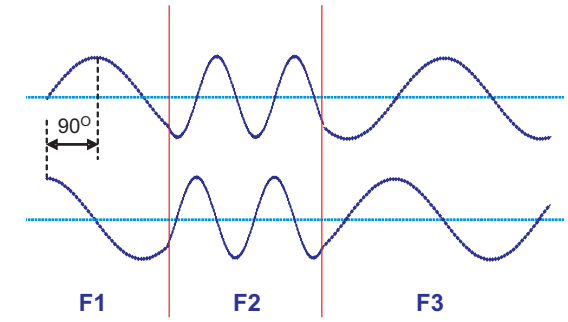


Figure 11A Local Oscillator Frequency Switching

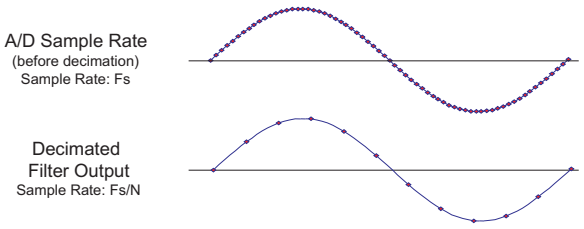


Figure 11B FIR Filter Decimation

Because the local oscillator uses a digital phase accumulator, it has some very nice features. It switches between frequencies with phase continuity, so you can generate FSK signals or sweeps very precisely with no transients as shown in Figure 11A.

The frequency accuracy and stability are determined entirely by the A/D clock so it's inherently synchronous to the sampling frequency. There is no aging, drift or calibration since it's implemented entirely with digital logic.

Since the output of the FIR filter is band-limited, the Nyquist theorem allows us to lower the sample rate. If we are keeping only one out of every N samples, as shown in Figure 11B above, we have dropped the sampling rate by a factor of N.

DDC Signal Processing

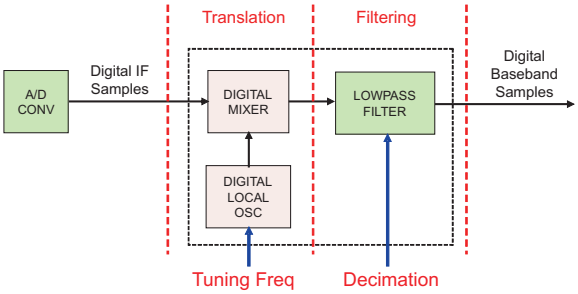


Figure 12

This process is called *decimation* and it means keeping one out of every N signal samples. If the decimated output sample rate is kept higher than twice the output bandwidth, no information is lost.

The clear benefit is that decimated signals can be processed easier, can be transmitted at a lower rate, or stored in less memory. As a result, decimation can dramatically reduce system costs!

As shown in Figure 12, the DDC performs two signal processing operations:

1. Frequency translation with the tuning controlled by the local oscillator.
2. Lowpass filtering with the bandwidth controlled by the decimation setting.

We will next turn our attention to the Software-Defined Radio Transmitter.

SDR Transmitter Block Diagram

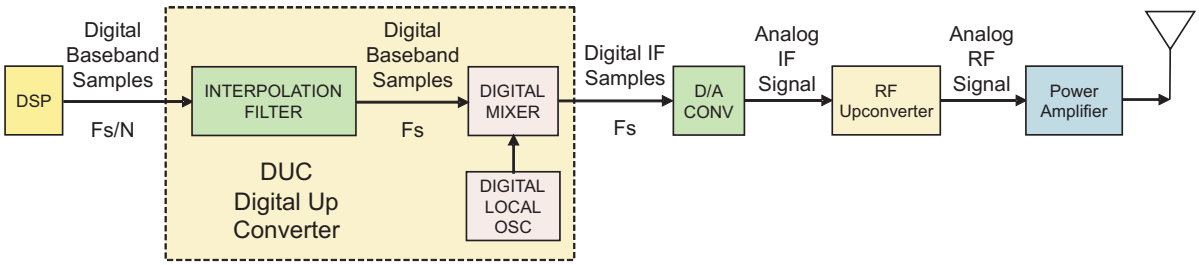


Figure 13

The input to the transmit side of an SDR system is a digital baseband signal, typically generated by a DSP stage as shown in Figure 13 above.

The digital hardware block in the dotted lines is a DUC (digital upconverter) that translates the baseband signal to the IF frequency.

The D/A converter that follows converts the digital IF samples into the analog IF signal.

Next, the RF upconverter converts the analog IF signal to RF frequencies.

Finally, the power amplifier boosts signal energy to the antenna.

DUC Signal Processing

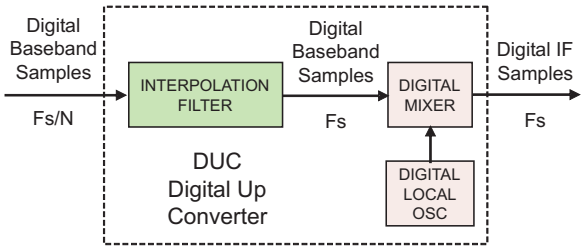


Figure 14

Inside the DUC shown in Figure 14, the digital mixer and local oscillator at the right translate baseband samples up to the IF frequency. The IF translation frequency is determined by the local oscillator.

The mixer generates one output sample for each of its two input samples. And, the sample frequency at the mixer output must be equal to the D/A sample frequency f_s.

Therefore, the local oscillator sample rate and the baseband sample rate must be equal to the D/A sample frequency f_s.

The local oscillator already operates at a sample rate of f_s, but the input baseband sample frequency at the left is usually much lower. This problem is solved with the *Interpolation Filter.*

Interpolation Filter: Time domain

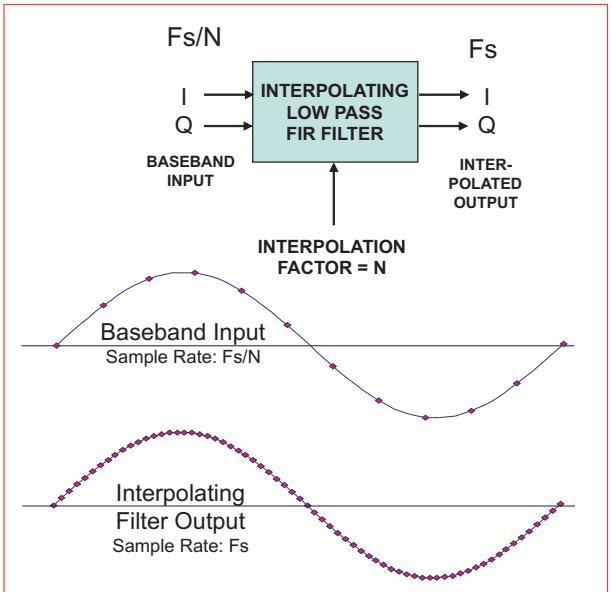


Figure 15

Interpolation Filter: Frequency Domain

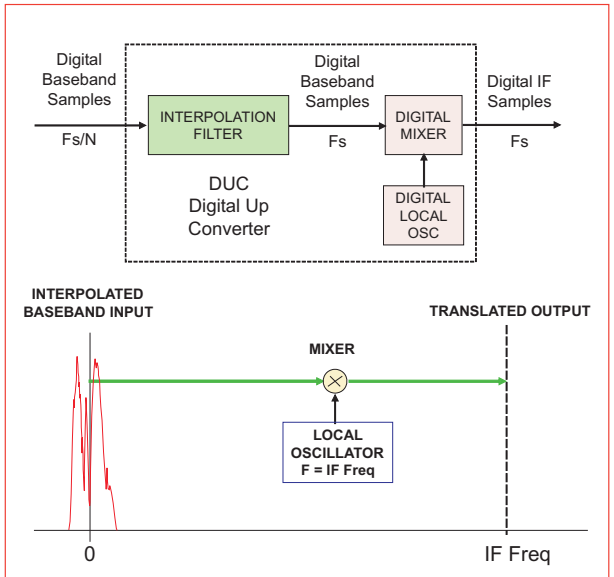


Figure 16

The interpolation filter must boost the baseband input sample frequency of f_s / N up to the required mixer input and D/A output sample frequency of f_s.

The interpolation filter increases the sample frequency of the baseband input signal by a factor N, known as the interpolation factor.

At the bottom of Figure 15, the effect of the interpolation filter is shown in the time domain.

Notice the baseband signal frequency content is completely preserved by filling in additional samples in the spaces between the original input samples.

The signal processing operation performed by the interpolation filter is the inverse of the decimation filter we discussed previously in the DDC section.

Figure 16 is a frequency domain view of the digital upconversion process.

This is exactly the opposite of the frequency domain view of the DDC in Figure 10.

The local oscillator setting is set equal to the required IF signal frequency, just as with the DDC.

DDC Processing

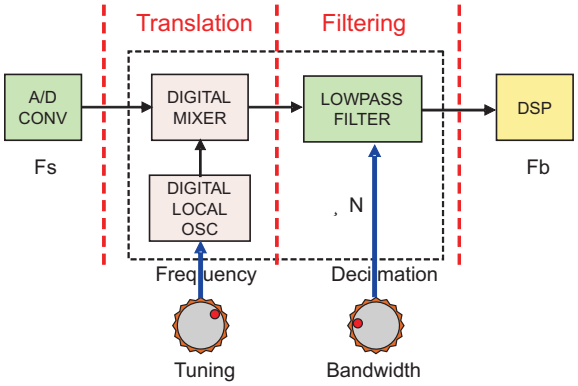


Figure 17

Figure 17 shows the two-step processing performed by the digital downconverter.

Frequency translation from IF down to baseband is performed by the local oscillator and mixer.

The "tuning knob" represents the programmability of the local oscillator frequency to select the desired signal for downconversion to baseband.

The baseband signal bandwidth is set by setting decimation factor N and the lowpass FIR filter:

- Baseband sample frequency $f_b = f_s / N$
- Baseband bandwidth = $0.8 * f_b$

The baseband bandwidth equation reflects a typical 80% passband characteristic, and complex (I+Q) samples.

The "bandwidth knob" represents the programmability of the decimation factor to select the desired baseband signal bandwidth.

DUC Processing

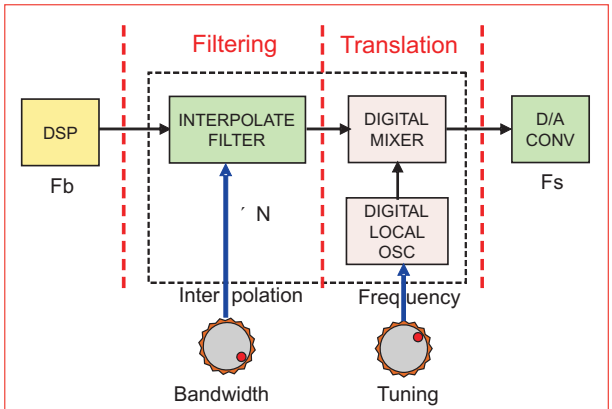


Figure 18

Figure 18 shows the two-step processing performed by the digital upconverter:

The ratio between the required output sample rate and the sample rate input baseband sample rate determines the interpolation factor N.

- Baseband bandwidth = $0.8 * f_b$
- Output sample frequency $f_s = f_b * N$

Again, the bandwidth equation assumes a complex (I+Q) baseband input and an 80% filter.

The "bandwidth knob" represents the programmability of the interpolation factor to select the desired input baseband signal bandwidth.

Frequency translation from baseband up to IF is performed by the local oscillator and mixer.

The "tuning knob" represents the programmability of the local oscillator frequency to select the desired IF frequency for translation up from baseband.

Key DDC and DUC Benefits

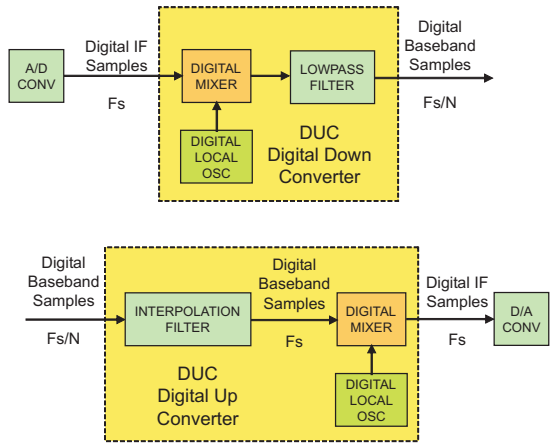


Figure 19

SDR Tasks

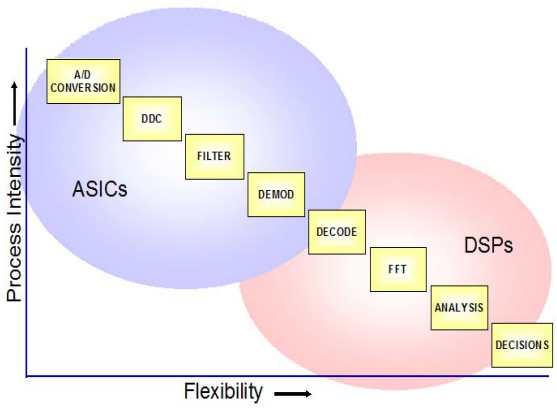


Figure 20

Think of the DDC as a hardware preprocessor for programmable DSP or GPP processor. It preselects only the signals you are interested in and removes all others. This provides an optimum bandwidth and minimum sampling rate into the processor.

The same applies to the DUC. The processor only needs to generate and deliver the baseband signals sampled at the baseband sample rate. The DUC then boosts the sampling rate in the interpolation filter, performs digital frequency translation, and delivers samples to the D/A at a very high sample rate.

The number of processors required in a system is directly proportional to the sampling frequency of input and output data. As a result, by reducing the sampling frequency, you can dramatically reduce the cost and complexity of the programmable DSPs or GPPs in your system.

Not only do DDCs and DUCs reduce the processor workload, the reduction of bandwidth and sampling rate helps save time in data transfers to another subsystem. This helps minimize recording time and disk space, and reduces traffic and bandwidth across communication channels.

Here we've ranked some of the popular signal processing tasks associated with SDR systems on a two axis graph, with computational Processing Intensity on the vertical axis and Flexibility on the horizontal axis.

What we mean by process intensity is the degree of highly-repetitive and rather primitive operations. At the upper left, are dedicated functions like A/D converters and DDCs that require specialized hardware structures to complete the operations in real time. ASICs are usually chosen for these functions.

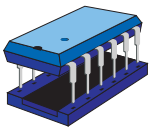
Flexibility pertains to the uniqueness or variability of the processing and how likely the function may have to be changed or customized for any specific application. At the lower right are tasks like analysis and decision making which are highly variable and often subjective.

Programmable general-purpose processors or DSPs are usually chosen for these tasks since these tasks can be easily changed by software.

Now let's temporarily step away from the software radio tasks and take a deeper look at programmable logic devices.

Early Roles for FPGAs

- Used primarily to replace discrete digital hardware circuitry for:
  - Control logic
  - Glue logic
  - Registers and gates
  - State machines
  - Counters and dividers



- Devices were selected by hardware engineers
- Programmed functions were seldom changed after the design went into production

Figure 21

As true programmable gate functions became available in the 1970's, they were used extensively by hardware engineers to replace control logic, registers, gates, and state machines which otherwise would have required many discrete, dedicated ICs.

Often these programmable logic devices were one-time factory-programmed parts that were soldered down and never changed after the design went into production.

Legacy FPGA Design Methodologies

- Tools were oriented to hardware engineers
  - Schematic processors
  - Boolean processors
  - Gates, registers, counters, multipliers
- Successful designs required high-level hardware engineering skills for:
  - Critical paths and propagation delays
  - Pin assignment and pin locking
  - Signal loading and drive capabilities
  - Clock distribution
  - Input signal synchronization and skew analysis

Figure 22

These programmable logic devices were mostly the domain of hardware engineers and the software tools were tailored to meet their needs. You had tools for accepting boolean equations or even schematics to help generate the interconnect pattern for the growing number of gates.

Then, programmable logic vendors started offering predefined logic blocks for flip-flops, registers and counters that gave the engineer a leg up on popular hardware functions.

Nevertheless, the hardware engineer was still intimately involved with testing and evaluating the design using the same skills he needed for testing discrete logic designs. He had to worry about propagation delays, loading, clocking and synchronizing—all tricky problems that usually had to be solved the hard way—with oscilloscopes or logic analyzers.

FPGAs: New Device Technology

- 500+ MHz DSP slices and memory structures
- Over 3500 dedicated on-chip hardware multipliers
- On-board GHz serial transceivers
- Partial reconfigurability maintains operation during changes
- Switched fabric interface engines
- Over 690,000 logic cells
- Gigabit Ethernet media access controllers
- On-chip 405 PowerPC RISC microcontroller cores
- Memory densities approaching 85 million bits
- Reduced power with core voltages at 1 volt
- Silicon geometries to 28 nanometers
- High-density BGA and flip-chip packaging
- Over 1200 user I/O pins
- Configurable logic and I/O interface standards

Figure 23

FPGAs: New Development Tools

- High Level Design Tools
  - Block Diagram System Generators
  - Schematic Processors
  - High-level language compilers for VHDL & Verilog
  - Advanced simulation tools for modeling speed, propagation delays, skew and board layout
  - Faster compilers and simulators save time
  - Graphically-oriented debugging tools
- IP (Intellectual Property) Cores
  - FPGA vendors offer both free and licensed cores
  - FPGA vendors promote third party core vendors
  - Wide range of IP cores available

Figure 24

It's virtually impossible to keep up to date on FPGA technology, since new advancements are being made every day.

The hottest features are processor cores inside the chip, computation clocks to 500 MHz and above, and lower core voltages to keep power and heat down.

Several years ago, dedicated hardware multipliers started appearing and now you'll find literally thousands of them on-chip as part of the DSP initiative launched by virtually all FPGA vendors.

High memory densities coupled with very flexible memory structures meet a wide range of data flow strategies. Logic slices with the equivalent of over ten million gates result from steadily shrinking silicon geometries.

BGA and flip-chip packages provide plenty of I/O pins to support on-board gigabit serial transceivers and other user-configurable system interfaces.

New announcements seem to be coming out every day from chip vendors like Xilinx and Altera in a never-ending game of outperforming the competition.

To support such powerful devices, new design tools are appearing that now open up FPGAs to both hardware and software engineers. Instead of just accepting logic equations and schematics, these new tools accept entire block diagrams as well as VHDL and Verilog definitions.

Choosing the best FPGA vendor often hinges heavily on the quality of the design tools available to support the parts.

Excellent simulation and modeling tools help to quickly analyze worst case propagation delays and suggest alternate routing strategies to minimize them within the part. This minimizes some of the tricky timing work for hardware engineers and can save one hours of tedious troubleshooting during design verification and production testing.

In the last few years, a new industry of third party IP (Intellectual Property) core vendors now offer thousands of application-specific algorithms. These are ready to drop into the FPGA design process to help beat the time-to-market crunch and to minimize risk.

FPGAs for SDR

- Parallel Processing
- Hardware Multipliers for DSP
  - FPGAs can now have over 500 hardware multipliers
- Flexible Memory Structures
  - Dual port RAM, FIFOs, shift registers, look up tables, etc.
- Parallel and Pipelined Data Flow
  - Systolic simultaneous data movement
- Flexible I/O
  - Supports a variety of devices, buses and interface standards
- High Speed
- Available IP cores optimized for special functions

Figure 25

FPGAs Bridge the SDR Application Space

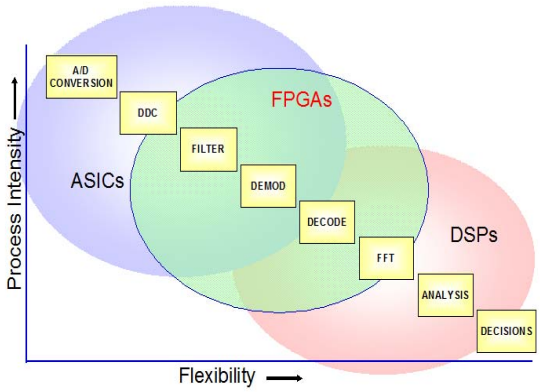


Figure 26

Like ASICs, all the logic elements in FPGAs can execute in parallel. This includes the hardware multipliers, and you can now get over 3500 of them on a single FPGA.

This is in sharp contrast to programmable DSPs, which normally have just a handful of multipliers that must be operated sequentially.

FPGA memory can now be configured with the design tool to implement just the right structure for tasks that include dual port RAM, FIFOs, shift registers and other popular memory types.

These memories can be distributed along the signal path or interspersed with the multipliers and math blocks, so that the whole signal processing task operates in parallel in a systolic pipelined fashion.

Again, this is dramatically different from sequential execution and data fetches from external memory as in a programmable DSP.

As we said, FPGAs now have specialized serial and parallel interfaces to match requirements for high-speed peripherals and buses.

As a result, FPGAs have significantly invaded the application task space as shown by the center bubble in the task diagram above.

They offer the advantages of parallel hardware to handle some of the high process-intensity functions like DDCs and the benefit of programmability to accommodate some of the decoding and analysis functions of DSPs.

These advantages may come at the expense of increased power dissipation and increased product costs. However, these considerations are often secondary to the performance and capabilities of these remarkable devices.