SAMPLIFY: HIGH-SPEED COMPRESSION AND DECOMPRESSION OF BANDLIMITED SIGNALS FOR SOFTWARE-DEFINED RADIO

Al Wegener
(Samplify Systems LLC, Portola Valley, CA USA, awegener@samplify.com)

ABSTRACT

Software-defined radio (SDR) has already adopted many commercial off-the-shelf (COTS) technologies from computer systems that lower the cost of flexible data acquisition and signal processing hardware and software. One useful technology that has not heretofore been used in SDR systems is signal compression. Compression has not been used in SDR systems for two primary reasons. First, SDR sampling rates are prohibitively high, limiting SDR compression algorithms to those that can cost-effectively be implemented in hardware. Second, SDR signals have varying bandwidths, center frequencies, and dynamic ranges, presenting a challenging set of conditions for any compression algorithm. This paper describes the SignalZIP™ lossless and Samplify™ lossy compression algorithms that compress bandlimited samples acquired by A/D converters, or provided to D/A converters, for SDR use. Samplify compression products, available both as evaluation software and as netlists for Altera and Xilinx FPGAs, allow SDR users to send 2x to 6x more data across existing busses (PCI, cPCI, VME, etc.) and networks (Ethernet, USB, FireWire, etc.)

1. COMPRESSION FOR SDR: MOTIVATION

Compression provides comparable benefits for SDR systems as do speech, audio, image, and video compression solutions. If SDR signals can be transferred and/or stored in compressed formats that provide comparable results (bit error rates, correlation results, measurements), SDR systems can realize both improved performance and reduced costs. SDR systems are usually built using industry-standard busses (PCI, VME, PXI, etc.), networks (Ethernet, USB, RACE++, FPDP, etc.), and FPGAs (Altera, Xilinx). Performing high-speed signal processing using FPGAs or DSPs interconnected by high-speed switching fabrics is arguably at the heart of SDR. While many intellectual property (IP) blocks are available for FPGAs, none has been offered that compresses signals at 100+ Msamp/sec while providing a variety of operating modes, such as lossless, lossy, and noise-reducing compression.

Imagine SDR systems whose bandwidth allocation could be optimized for a particular mix of signal processing operations. Such SDR systems could optimize their most precious resource: bandwidth. Through the use of expansion mezzanine cards such as PMC and ATCA AMC, FPGA gates and DSP MIPS can be expanded after an SDR platform has been fielded. However, the inherent signal transfer capacity of SDR platforms is usually not expandable because this capacity is an inherent part of the bus or network that connects the processing elements. Compression allows system bandwidth to be modified and optimized even after SDR system deployment.

Compression's ability to extend the life of legacy signal processing systems that use VME, PCI, and compact PCI (cPCI) backplanes may be its most compelling benefit. Adding compression to such legacy systems using FPGAs flexibly expands bus bandwidth and storage capacity without requiring the replacement of existing backplanes and networks. Compression is added by replacing legacy boards with newer boards that compress backplane traffic. When existing SDR FPGA boards have available capacity (CLBs or LEs), SignalZIP and Samplify compression can even be integrated into these legacy boards as FPGA IP blocks.

2. COMPRESSION EXAMPLES

2.1. Introduction

The following three examples demonstrate how high-speed sampled data compression can improve SDR systems.

2.2. Example 1: Improving FFT Resolution

Many SDR systems detect the presence of signals. The first step in a wideband signal detection process is peak detection and identification using an FFT. Because SDR analog front ends often operate at A/D sampling rates that exceed the bandwidth of the bus that connects the cards, snapshot RAMs are used to capture samples directly from high-speed A/D converters. If high-speed compression is inserted between the A/D converter and the snapshot RAM, 2x to 4x more data can be stored in the same amount of physical RAM. When an FFT is performed on the decompressed data, the resolution of each FFT bin is also improved by 2x to 4x. In Figure 1, we assume an A/D

Proceeding of the SDR 05 Technical Conference and Product Exposition. Copyright © 2005 SDR Forum. All Rights Reserved

sample rate of 200 Msamp/sec (12-bit samples) and a snapshot RAM depth of 4096 samples. When 2x or 4x compression is added, FFT resolution is 2x or 4x better, with a slight degradation in noise floor from the Samplify fixed-rate lossy compression algorithm used in this example to compress the signal.

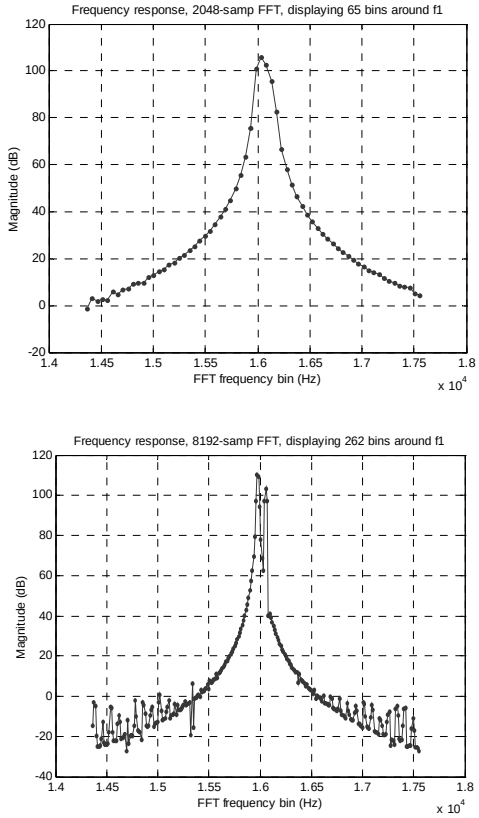


Figure 1: Compressing snapshot RAM captures improves FFT resolution from 25 kHz/bin (4k FFT; top) to 6.25 kHz/bin (16k FFT, bottom), allowing closely spaced signals to be resolved.

2.3 Example 2: Increasing Signal Duration and Bandwidth of Arbitrary Waveform Generators

SDR systems must often generate arbitrary waveforms that emulate real-world signals. In some cases, such waveforms are of limited duration and can be stored in RAM. But since RAM is expensive, the per-sample cost of such arbitrary waveform storage is high. In other cases, waveforms of longer duration are streamed from disk. But in that case, the disk drive interface limits the maximum sampling rate of the D/A converter. Compression can improve both RAM-based and disk-based waveform generators.

Figure 2 shows the first 1000 samples of a 3G wireless test signal (2-carrier Test Model 3, 10 MHz bandwidth) sampled at 61.44 Msamp/sec. The quality of 3G test signals can be measured in several ways, including the adjacent channel leakage ratio (ACLR). The ACLR of Figure 2's waveform is 85 dB, well above an acceptable ACLR level of 60 dB. Using the Samplify compression algorithm to trade bit rate for signal quality, the 3G test signal can be compressed 2:1 with a decrease in ACLR from 85 dB to 77 dB. At 3:1 compression, the ACLR of the signal is 61 dB, just above the target 60 dB level. If an SDR system storing one 3G frame (10 msec) in snapshot RAM were augmented with Samplify decompression between the RAM and the D/A converter, the same RAM could store 2 frames (20 msec) at 2x compression, or 3 frames (30 msec) at 3x compression. In 3G applications where signal duration is more important than ACLR levels, Samplify compression increases snapshot RAM duration by up to 3x.

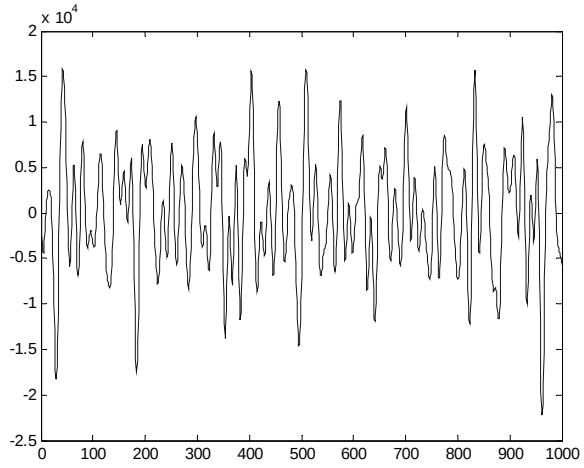


Figure 2: A two-carrier 3G wireless waveform with ACLR of 85 dB can be compressed by 2:1 (ACLR = 77 dB) or 3:1 (ACLR = 61 dB), thus improving both snapshot RAM-based and disk drive-based arbitrary waveform generators.

Similarly, if an uncompressed disk drive interface fed an arbitrary waveform generator at (for example) 20 Msamp/sec, adding compression between the disk drive interface and the D/A converter increases the maximum D/A sample rate to 40 Msamp/sec (2:1 compression) or 60 Msamp/sec (3:1 compression). Compression thus allows older, slower disk drives to support higher D/A sampling rates in an arbitrary waveform generation system.

2.4 Example 3: Compressing Radar Returns

The thruput of SDR systems that process radar signals can be improved through compression. Figure 3 shows the first 3500 samples of a radar return that was sampled at 500 Msamp/sec (8 bits/sample) using a 32/33 PCI data capture card (132 MB/sec). In this example, 16 kB buffers of radar samples are first captured in the capture card's snapshot RAM. Each buffer is then transferred across the PCI bus to a DSP card for correlation measurements. The processing bottleneck in this application is the PCI bus. While a 16k

Proceeding of the SDR 05 Technical Conference and Product Exposition. Copyright © 2005 SDR Forum. All Rights Reserved

buffer is captured in 32 µsec and is processed by the DSP card in 15 µsec, each 16 kB PCI transfer takes 121 µsec. By adding Samplify real-time compression to the capture card and Samplify real-time decompression to the DSP card, the effective PCI bus transfer rate is increased by the compression ratio. For the signal shown in Figure 3, SignalZIP lossless compression achieves 2.87:1 compression, decreasing the buffer transfer time from 121 µsec to 42 µsec. When Samplify lossy compression is used, correlation measurements were within acceptable tolerances until the lossy compression ratio reached 6:1. So in this radar application, Samplify real-time compression decreased the buffer transfer time by 6x, from 121 µsec to less than 20 µsec, without affecting correlation measurements.

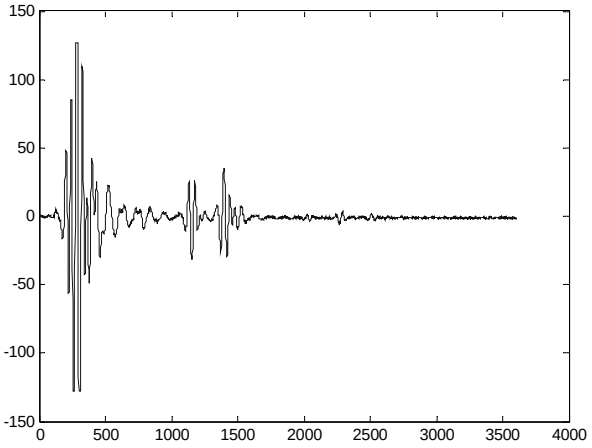


Figure 3: *Radar reflections can be transferred across the PCI bus 2.87 times faster using SignalZIP lossless compression, while Samplify lossy compression achieves a 6x PCI bus speed-up.*

3. COMPRESSION PERFORMANCE AND RATE-DISTORTION CURVES

3.1. Signal Characteristics that Improve Compression

High-speed compression of A/D and D/A converter samples is admittedly a new application area [1], [2]. However, several general statements can be made about the effectiveness of such compression and the application areas for which lossy compression is appropriate. The best advice is simply to try a software version of the compression algorithm to determine how well *user* signal(s) will compress. Using a software version of the compression algorithm also allows users to experiment with lossy compression settings to determine the effects of higher compression ratios on overall system performance.

SignalZIP and Samplify high-speed compression algorithms identify and remove three kinds of redundancies from A/D and D/A sample streams:

a) unused dynamic range
b) noise floor effects
c) sample-to-sample correlations

Regarding unused dynamic range, if the input gain of a signal that drives a 12-bit A/D converter is not properly calibrated, only 11 bits might ever be used. Regarding noise floor effects, every A/D and D/A converter has a figure of merit called "effective number of bits" (ENOB) that is related to the converter's noise floor and is related to SNR. If a 16-bit D/A converter's ENOB is only 15 bits, the LSB of the converter contains random ones and zeros that will not improve measurements or signal quality. Finally, many signals are oversampled, i.e. the data converter sampling rate is several times higher than the highest frequency component in the signal. In such cases, sample-to-sample correlations can be removed during compression and re-inserted during decompression. Oversampled signals are also known as bandlimited signals, since their signal bandwidth (relative to the sampling rate) is limited.

To summarize, SignalZIP and Samplify provide the highest compression ratios when the input signal:

a) does not always occupy the full dynamic range of the data converter, i.e. when the signal is bursty or impulsive or has low duty cycle
b) has a less-than-perfect SNR (i.e. when noise can be removed prior to compression, improving the compression ratio without affecting the results)
c) is bandlimited or oversampled.

3.2. Expectations for Lossless Compression

Many computer users have frequent, regular experience with the computer file compression programs WinZIP and PKzip. These lossless compression applications are used wherever computer data files (.doc, .xls, .txt, etc.) having character string redundancy are transmitted via e-mail or through a network. WinZIP and PKzip regularly reduce computer file size by 2x to 3x. However, since the Lempel-Ziv (LZ) algorithm underlying WinZIP and PKzip applications is lossless, computer users are aware that while the LZ algorithm does its best to make the file smaller, LZ cannot guarantee a specific compressed file size. LZ doesn't compress A/D and D/A converter samples very well because LZ is character-oriented, not sample-oriented.

SignalZIP lossless compression (optimized for compressing A/D and D/A converter samples) removes the redundancies found in sampled data streams. But just like LZ compression, SignalZIP's lossless compression ratio varies, depending on the type and degree of redundancy present in the input signal. As a general rule, SignalZIP often achieves 2:1 lossless compression when the signal bandwidth is 20% or less of the sampling rate, regardless of the signal's center frequency.

Lossless compression performance is strongly correlated with SNR. This makes intuitive sense, since

Proceeding of the SDR 05 Technical Conference and Product Exposition. Copyright © 2005 SDR Forum. All Rights Reserved

signals with lower SNR have a higher random content than signals with higher SNR. And streams of random numbers cannot be compressed because they are uncorrelated. Thus SignalZIP users should expect their lossless compression ratio to degrade with signal quality as SNR decreases.

Similarly, as the input signal becomes less bandlimited and its bandwidth occupies the Nyquist bandwidth from DC to $f_s/2$, its spectrum more and more resembles the spectrum of white noise (equal signal power at all frequencies). This explains why SignalZIP compression degrades as signals become less bandlimited: as the signal's high-power spectral content widens, the signal's characteristics increasingly resemble white noise, which cannot effectively be compressed by *any* lossless algorithm.

3.3. Expectations for Lossy Compression

The Samplify lossy compression algorithm offers two operating modes. In Samplify fixed-rate mode, users select the desired compression ratio (or bit rate), from 1.05:1 to 8:1, in increments of 0.05. Users can fine-tune their chosen bit rate until the resulting SNR is acceptable for the user's application. In Samplify fixed-quality mode, users can fine-tune the desired signal quality (SNR) in increments of 0.5 dB until the resulting bit rate is acceptable for the user's application.

Samplify lossy compression differs in significant ways from popular lossy compression algorithms developed for speech (ADPCM), audio (MP3 or WMA), images (JPEG), and video (MPEG). Speech, audio, image, and video compression algorithms exploit known weaknesses in human hearing and vision. For instance, MP3 and WMA exploit the fact that human hearing is less sensitive to frequencies below 100 Hz and above 3 kHz. MP3 and WMA use fewer bits to encode these audio frequencies. The impressive 10:1 audio and 20:1 video compression ratios create significant, time-varying frequency-domain and time-domain distortions, but humans just can't hear or see these distortions.

Unfortunately for SDR signals, such time-varying, unpredictable distortions are very undesirable. A different lossy compression approach is needed to compress SDR-type signals. Samplify's distortion metric for lossy compression is both simple and familiar to communications engineers: SNR. During lossy compression, changes in SNR quantify Samplify's bit rate vs. quality tradeoff.

3.4. Generating Rate-Distortion Curves Automatically

While lossless compression is desirable in some applications, lossy compression is generally more useful in real-time, sampled data systems because lossy compression can deliver a fixed data rate. Lossless compression has a fixed quality (no distortion), so the compression ratio and bit rate vary with the redundancy of the signal. But varying data rates are awkward to handle in systems with fixed A/D and D/A clock rates. Even though Samplify lossy compression can deliver a fixed bit rate, most engineers may have no idea of what bit rate can be achieved, or how lossy compression affects signal processing performance. A compression evaluation tool is needed that provides them with these answers, with access to all compression settings.

Information theory provides a useful signal analysis tool for lossy compression: the rate-distortion curve. Figure 4 below illustrates a rate-distortion curve. A rate-distortion curve displays the bit rate (measured in bits/sec or bits/sample) or compression ratio on the y axis and the distortion (usually measured as SNR) on the x axis. As the compression ratio increases (bit rate decreases), the distortion rises (SNR decreases).

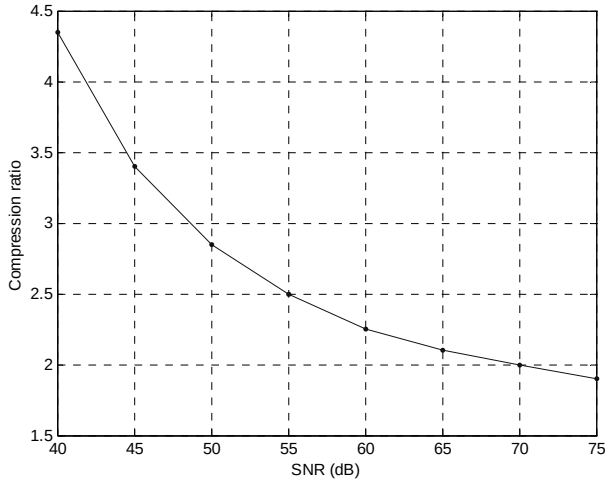


Figure 4: *A signal's rate-distortion curve provides SignalZIP and Samplify users with a one-click summary of the range of tradeoffs between bit rate and signal quality available from compression.*

Once a signal's rate-distortion curve has been generated, Samplify users can select a desired operating point using either Samplify's fixed-rate or fixed-quality mode. Samplify Systems' evaluation software generates rate-distortion curves in Windows, Matlab, and LabVIEW environments that compress users' binary or ASCII (text) input files.

3.5. The Limits of Compression

Compression has certain fundamental limits that will never be exceeded. In 1948, Claude Shannon's foundational information theory paper [3] defined the entropy (information content) of a data stream based on the probabilities of characters in that data stream. Shannon showed mathematically that compression algorithms can never exceed the entropy of an information stream. Shannon's information theory sets an upper bound on the

Proceeding of the SDR 05 Technical Conference and Product Exposition. Copyright © 2005 SDR Forum. All Rights Reserved

compression ratio achieved by any compression algorithm, regardless of its complexity. Shannon also showed that if lossy compression is permitted, i.e. if some errors are allowed in the transmission of the desired characters, the compression ratio can be increased. Unfortunately, Shannon's information-theoretic proofs do not describe how to construct well-performing compression algorithms – they simply indicate how well the best possible compression algorithm can perform.

In the cost-conscious world of software and electronics, compression has an associated cost parameter that strongly influences the desirability and ultimate market acceptance of high-speed compression products. Samplify Systems developed the SignalZIP and Samplify algorithms with the practical goal of minimizing product cost by keeping algorithm complexity low, while operating at high sampling rates. While the compression ratios achieved by SignalZIP and Samplify may someday be eclipsed by higher-complexity (and higher-cost) compression algorithms, high-speed compression users may well decide that a 10% improvement in compression ratio is not worth (for example) a 10x increase in complexity and product cost.

To summarize, SignalZIP and Samplify are the first general-purpose high-speed compression algorithms available for FPGAs and ASICs, and they will continue to rank among the lowest-cost solutions for compressing A/D and D/A converter samples because of their low complexity.

4. SIGNALZIP, SAMPLIFY, AND NOISETRAK OPERATING MODES

SDR users who decide to try compression for their signals should first determine the SignalZIP lossless compression ratio for these signals. After all, if lossless compression provides significant bandwidth and/or storage savings at no reduction in signal quality, SignalZIP is the logical choice.

However, lossless compression *always* creates a varying bit rate. While variations in the compressed bit rate may be small (a few percent), lossless compression cannot guarantee a fixed bit rate. The size of SignalZIP compressed packets varies, depending on the redundancy that SignalZIP discovers in the signal. However, varying packet sizes can be a problem for real-time processing, since such systems are simpler to implement using fixed-size packets and fixed-rate clocks (and without rate-matching FIFOs).

If fixed bit rates or fixed packet sizes are needed, SDR users should try Samplify fixed-rate mode, since it offers the same packet size for every input block of samples, regardless of the redundancy of input signal samples. But as result, the signal quality may vary from packet to packet using fixed-rate mode. With a fixed bit rate, as the redundancy of the input signal decreases (i.e. noise level rises or bandwidth increases), users should also expect the decompressed signal quality to degrade.

If fixed signal quality is desired during compression, such as when a given bit error rate (BER) or specific SNR must be maintained, Samplify's fixed-quality compression mode can be used. In this mode, users select the minimum acceptable SNR of the decompressed signal, with granularity of 0.5 dB. As with SignalZIP mode, the bit rate of Samplify fixed-quality mode will vary. The difference between SignalZIP lossless mode and Samplify fixed-quality mode is simply the allowable distortion level: SignalZIP introduces no distortion, while Samplify fixed-quality mode maintains the distortion level or SNR specified by the user.

NoiseTrak™ is the last operating mode that may be of interest to SDR users. In NoiseTrak mode, the noise content of the input signal is continually monitored, and that noise content is removed prior to compression. As was discussed earlier, a data converter's ENOB determines its useful dynamic range (SNR). Several system and environmental factors also influence the SNR of a signal: the quality of the analog front end or anti-alias filter, the length of cable or PC board trace between the analog input (output) and the A/D (D/A) converter, crosstalk levels, sampling clock jitter, etc.

Ideally, the digital interface of future A/D and D/A converters will deliver simply the bits that matter, instead of identical number of bits for each sample. By using compression after A/D (or prior to D/A) conversion, SDR users can provide simply the bits that matter to data converters, decreasing both bandwidth and storage costs.

5. SIGNALZIP AND SAMPLIFY SOFTWARE AND HARDWARE IMPLEMENTATIONS

5.1. Software for Windows, Matlab, and LabVIEW,

In order to allow SDR systems engineers to evaluate the achievable compression ratios and lossy compression performance of the SignalZIP lossless and Samplify lossy compression algorithms, Samplify Systems developed three software applications: for Windows, for National Instruments' LabVIEW environment, and for The Mathworks' Matlab environment. Figure 5 shows a Samplify for Windows screen shot. All three software applications accept sampled data files in binary or ASCII (text) format. Users can then experiment with both lossless and various lossy compression settings to determine those settings which are acceptable for their signals. The compression results and thruput of all three software products are identical – about 10 Msamp/sec using a Pentium-4. Samplify software compression results are bit-for-bit identical to those achieved by Samplify Systems' FPGA netlists and ASIC products, except for processing

Proceeding of the SDR 05 Technical Conference and Product Exposition. Copyright © 2005 SDR Forum. All Rights Reserved

speed, which is 100+ Msamp/sec in FPGA-based systems and higher for planned ASIC implementations.

5.2. Hardware Products: Netlists for FPGAs

For SDR applications where FGPA technology enables field upgradeability and expandability, Samplify Systems offers SignalZIP and Samplify compression and

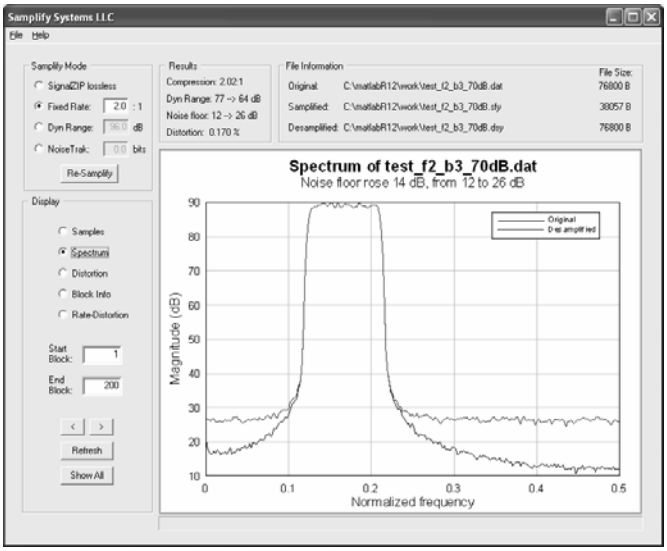


Figure 5: *Samplify for Windows, LabVIEW, and Matlab allow sampled data users to try SignalZIP and Samplify compression on their own data and to generate one-click rate-distortion curves.*

decompression blocks as encrypted netlists for both Altera and Xilinx FPGAs. All netlists operate at 100+ Msamp/sec (exact thruput is device-dependent), and all netlists support both SignalZIP lossless and Samplify lossy compression or decompression. Samplify Systems' encrypted netlists are of low complexity (CLB or LE count), as summarized in Table 1 below. Memory requirements of about 12k bits are similar for both Altera and Xilinx netlists, and for both compression and decompression netlists.

Table 1: *SignalZIP and Samplify compression and decompression functions are available as encrypted netlists for Altera and Xilinx FPGAs.*

FPGA function	*Logic complexity*	*Memory used*
Altera (compress)	***4100 LEs***	***12.3 kbits***
Altera (decompress)	***3000 LEs***	***12.3 kbits***
Xilinx (compress)	***2100 CLBs***	***12.3 kbits***
Xilinx (decompress)	***1500 CLBs***	***12.3 kbits***

The relatively small logic and memory footprint of SignalZIP and Samplify IP blocks make them ideally suited for SDR FPGA platforms. SignalZIP and Samplify FPGA netlists can also be parallelized to operate sampling rates above 100 Msamp/sec, with just a linear increase in hardware footprint. For example, if 300 Msamp/sec compression performance is required, simply multiply Table 1's LE or CLB count by 3.

5.3. Hardware Products: Low-cost ASICs

Samplify Systems is implementing the SignalZIP and Samplify algorithms at higher sampling rates and lower gate counts in low-cost ASICS that are targeting high-volume unit prices below $10. At this price, even low-cost USB and PCI products will be able to offer 2x to 6x more bandwidth at sampling rates up to 200 Msamp/sec (CMOS interfaces) or 1 Gsamp/sec (LVDS and SerDes interfaces).

Companies that develop SDR platforms should find the low cost of Samplify ASICs attractive. Such low-cost devices offer a compelling new technology that increases sampled data thruput across standard busses and networks. Samplify ASICs will make "virtual memory" for both snapshot RAM (signal capture) and arbitrary waveform RAM (signal generation) a cost-competitive reality when SDR systems require additional bandwidth and storage. As SDR sampling rates continue to increase, Samplify Systems' high-speed compression products provide a novel and low-cost way to increase SDR system bandwidth and to reduce SDR sampled data storage costs.

6. CONCLUSION

This paper has introduced the SignalZIP lossless and Samplify lossy compression algorithms for SDR applications. By integrating SignalZIP and Samplify in FPGAs, DSPs, and ASICs, SDR systems can increase A/D and D/A converter operating rates without replacing the legacy PCI, VME, and Ethernet communications links that support them. Similarly, SDR applications requiring snapshot memory or disk drive storage can realize both lower costs and higher performance by integrating compression. Samplify Systems compression products are offered as low-cost evaluation software, licensed FPGA netlists, low-cost ASICs, and via patent licenses.

10. REFERENCES

[1] D. Tuite, "Hardware Algorithm Fine-tunes Converters for Best Compression," *Electronic Design*, 20 September 2004, ED Online ID #8691.

[2] A. Wegener, "Samplify: Compression of A/D and D/A Converter Samples at 100 Msamp/sec," *GSPx Conference*, Santa Clara, CA, 30 Sep 2004.

[3] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, pp. 379-423 and 623-656, July and October, 1948.

Proceeding of the SDR 05 Technical Conference and Product Exposition. Copyright © 2005 SDR Forum. All Rights Reserved

Samplify:
High-speed Compression and Decompression of Bandlimited Signals for Software-Defined Radio

Al Wegener
Samplify Systems LLC
Portola Valley, CA
www.samplify.com

Samplify Systems

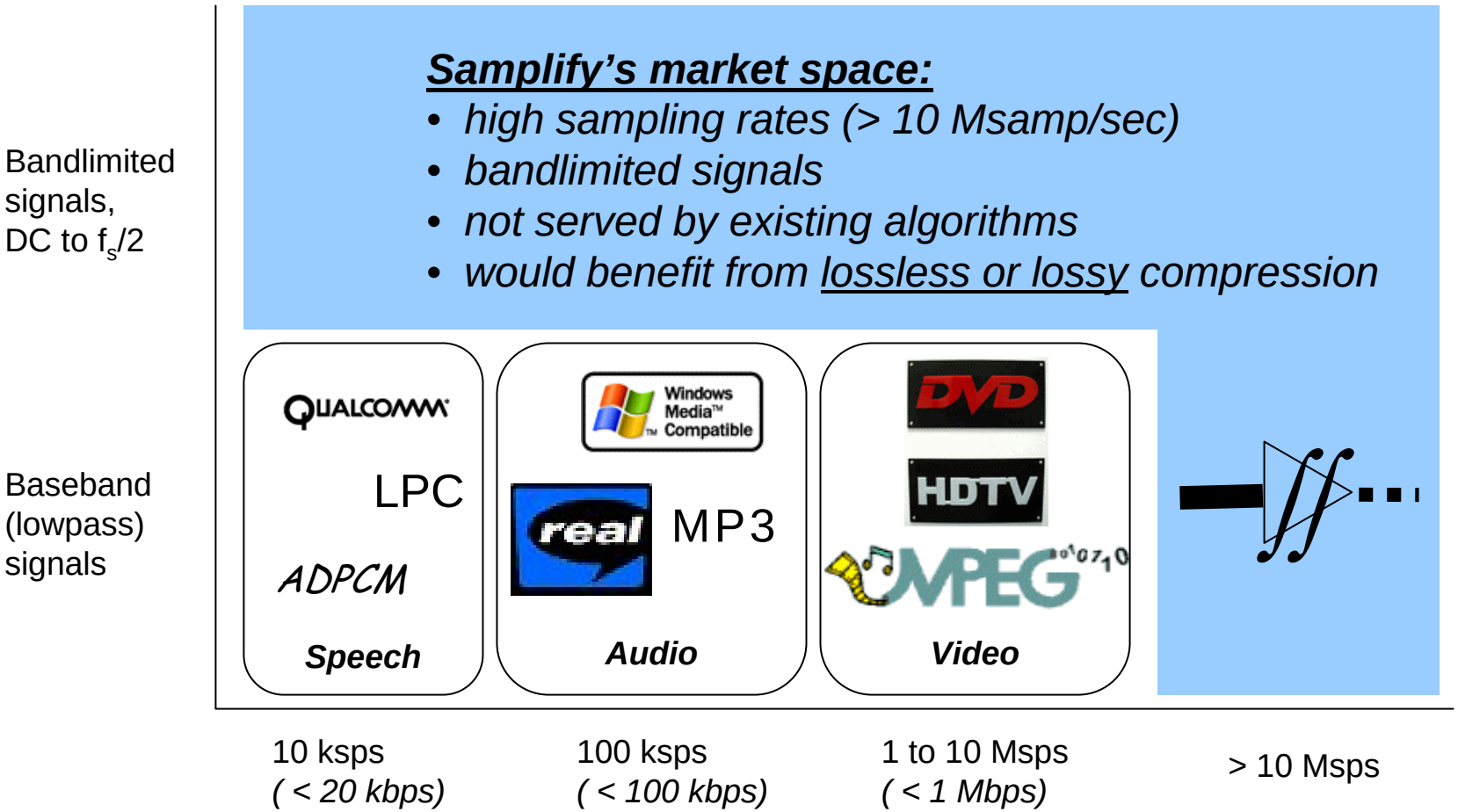
Agenda

- **Why compress SDR signals?**
- **Earlier compression methods & applications**
- **COTS Usage in SDR**
- **SignalZIP, Samplify, and NoiseTrak**
- ***Samplify for Windows* demo**
- **How does Samplify improve SDR systems?**
- **Product road map**
- **Patent status**
- **Samplify <u>your</u> SDR signals:**
  Samplify for Matlab, LabVIEW, FPGA & DSP

Why Compress SDR Signals?

- Wider bandwidth
  - Increase bandwidth of standard networks (PCIe, ATA, USB, etc.)
  - Reduce latency (smaller packets transfer faster)
- Lower product cost
  - Lower signal storage costs (stream to disk vs. snapshot in RAM)
  - Lower network transfer costs per MB transferred
- Improved performance / measurements
  - More samples → better results
  - Examples: improved spectral resolution, lower uncertainty
- *More bandwidth across legacy busses*

Industry Standard Compression Methods



COTS Usage in SDR

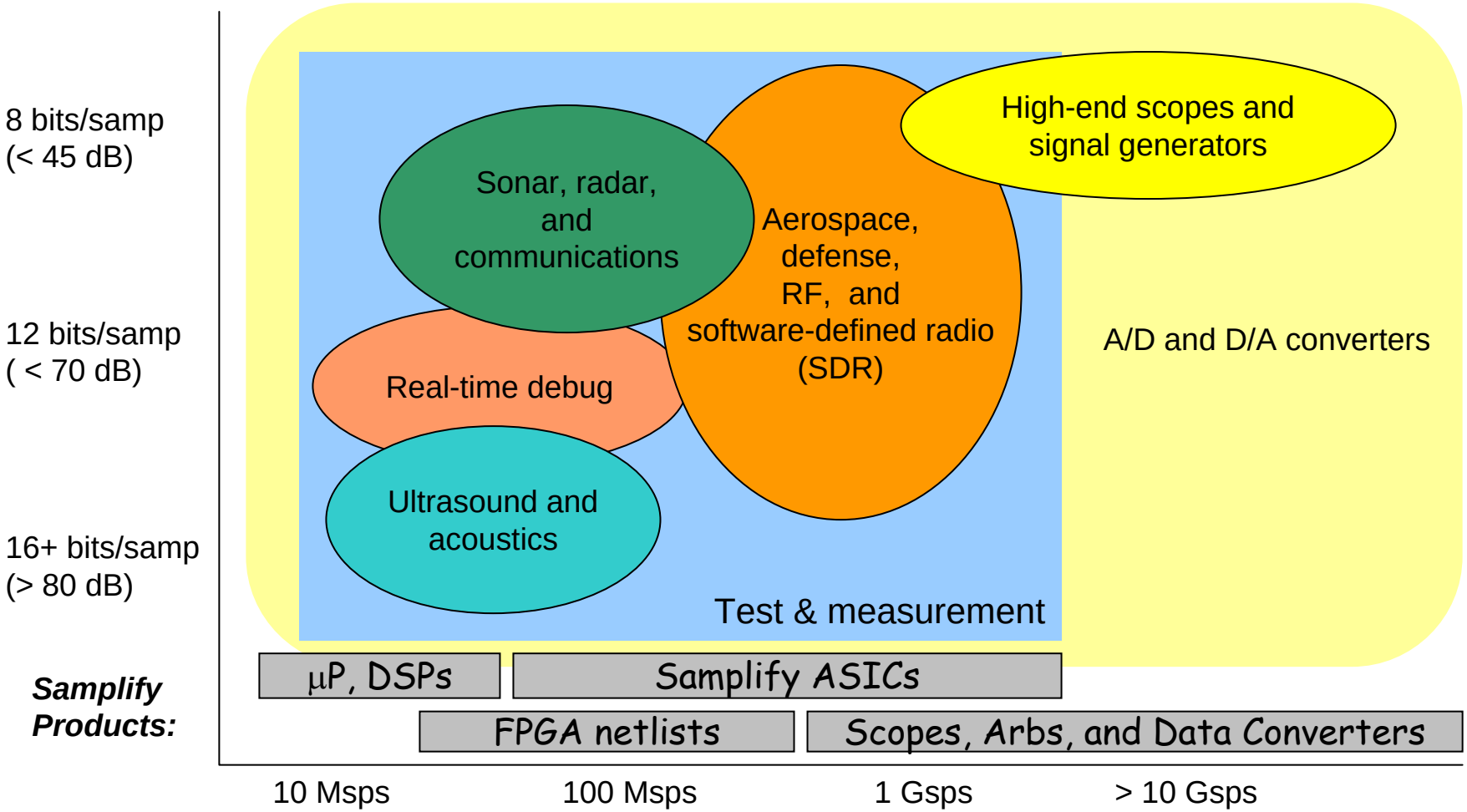
- FPGAs (Altera, Xilinx)
- DSPs (TI, Analog Devices, Freescale)

- Busses (VME, PCI, PCIe, etc.)
- Disk drive interfaces (IDE, ATA, SATA, SCSI, FibreChannel, etc.)
- Custom busses (PXI, VXI, ATCA, etc.)
- Networks (Ethernet, USB, RapidIO, FireWire/1394, etc.)
- Custom networks (FPDP, SkyChannel, Race++, etc.)
- Mezzanine cards (PMC, AMC, VIM)

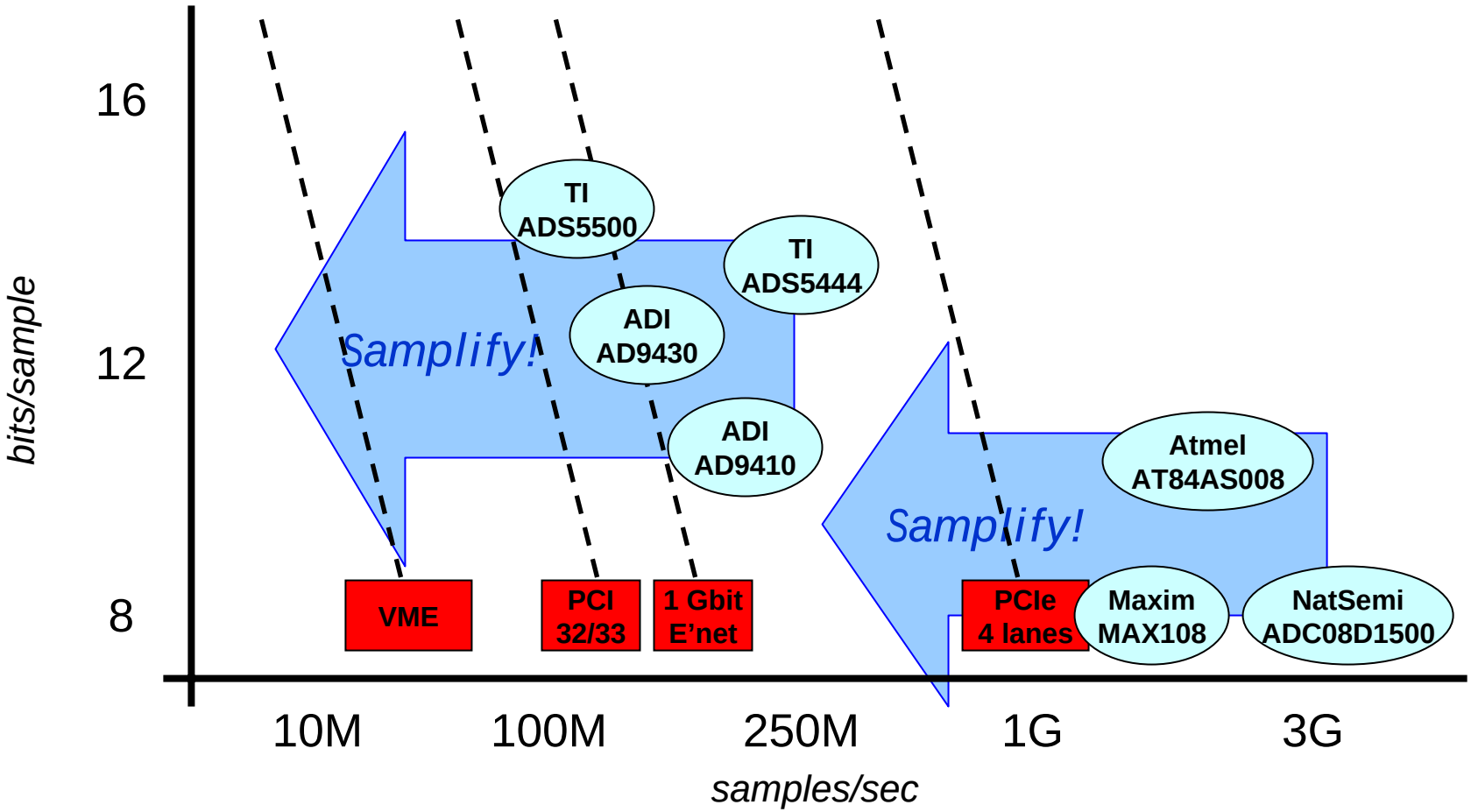
ALL of these COTS pipes have a limited bandwidth
As A/D and D/A rates increase, the pipes hit the wall

Samplify's value proposition:
Avoid or defer "hitting the wall" by using compression

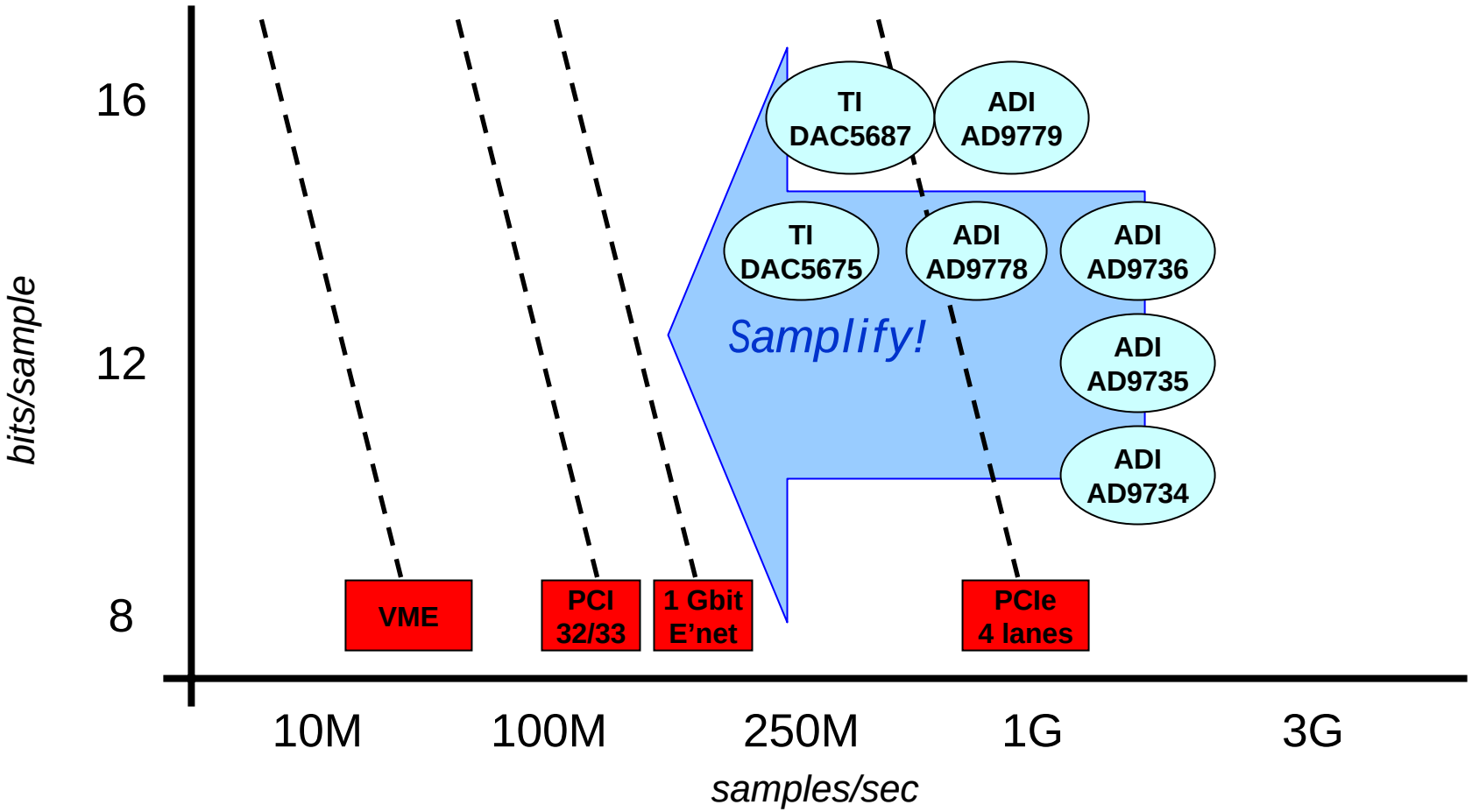
Samplify's Target Applications



A/D Converter Trends vs. Bus Speeds



D/A Converter Trends vs. Bus Speeds



Ideal Compression Characteristics for SDR

- Wide range of sampling rates (10 Msps to 40+ Gsps)
- Wide range of implementation platforms (PCs, DSP chips, FPGAs, ASICs)
- Bit widths: 8 to 16 bits per sample
- Center frequency f_c of bandlimited signals anywhere from DC to $f_s/2$
- User-selectable compression modes:
  - Lossless: bit-for-bit identical to the original sampled waveform
  - Lossy: choose the compression ratio or the distortion level
- Achievable compression ratios:
  - Lossless: 2:1 compression for narrowband signals
  - Lossy compression: user-selected compression ratio from 1.1:1 to 8:1, with 0.05 resolution

- Algorithm should *tell me how well my signal will compress*

Lossless Versus Lossy

Lossless compression guarantees the integrity of the data, but with signal-dependent bandwidth and storage savings.

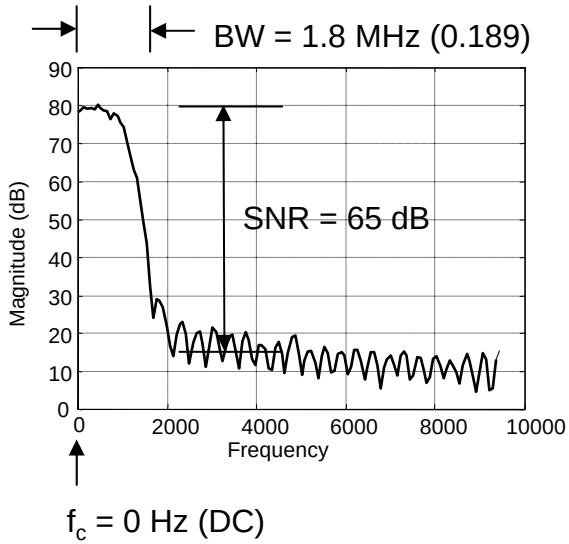
Lossy compression guarantees the bandwidth and storage savings, possibly with some user-controlled signal degradation.

The application suggests the appropriate choice.

The user makes the final decision.

Bandlimited Signal Parameters

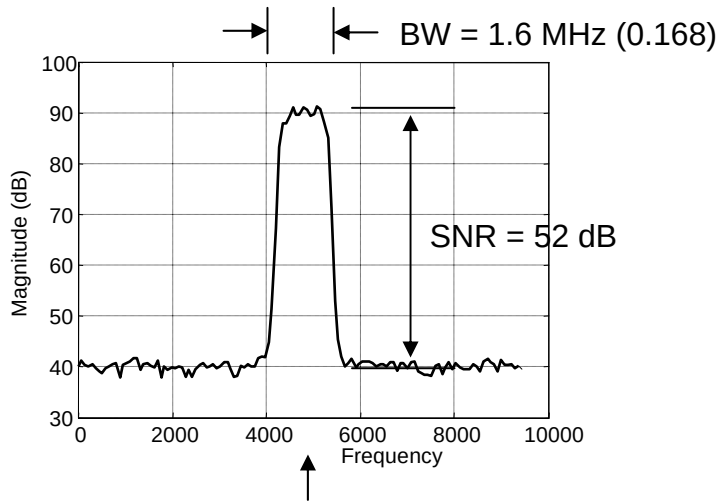
Example 1



f_c = 0 Hz (DC)

- ***Center freq: 0 (DC)***
- ***Bandwidth: 1.8 MHz***
- ***SNR: 65 dB***

Example 2



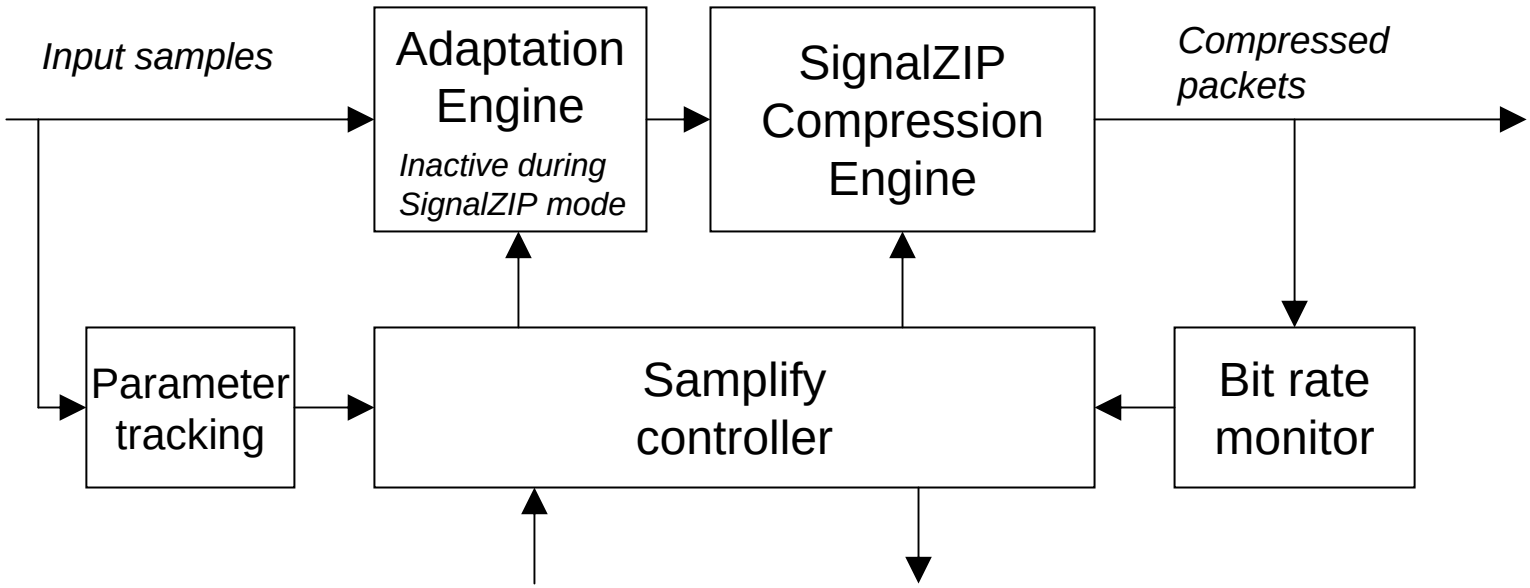
f_c = 5 MHz

- ***Center freq: 5 MHz***
- ***Bandwidth: 1.6 MHz***
- ***SNR: 52 dB***

Samplify Compression Modes

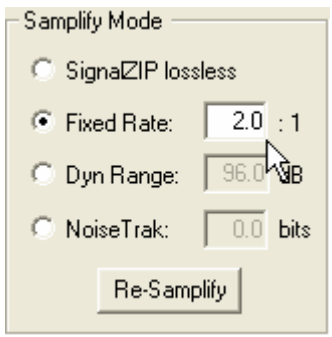
- **SignalZIP™** Lossless mode:
  - compression rate varies depending on the signal
  - ~2:1 compression on some bandlimited signals
- **Samplify™** Lossy mode:
  - Mode 1: User-specified compression ratio (ex: 1.5:1, 3.3:1)
  - Mode 2: User-specified dynamic range (ex: 64.5 dB, 75.2 dB)
- **NoiseTrak™** Lossy mode:
  - Track the sampled data's dynamic range and noise floor
  - Preserve only those parts of the signal that are "useful"

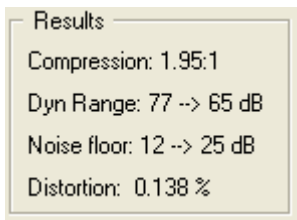
Adaptive Compression



MODE CONTROL:

User selects compression mode and desired goal parameter





RESULTS:

Samplify controller returns resulting bit rate, distortion, dynamic range, and noise floor estimates

Improving Measurements with Samplify

More samples → better measurements

Assumption: the measurement device captures samples in a fixed amount of memory (ex: digital storage scope, snapshot RAM board, disk drive)

Example 1: FFT resolution = f_s / N

the more samples collected, the better the resolution ***(linear improvement)***

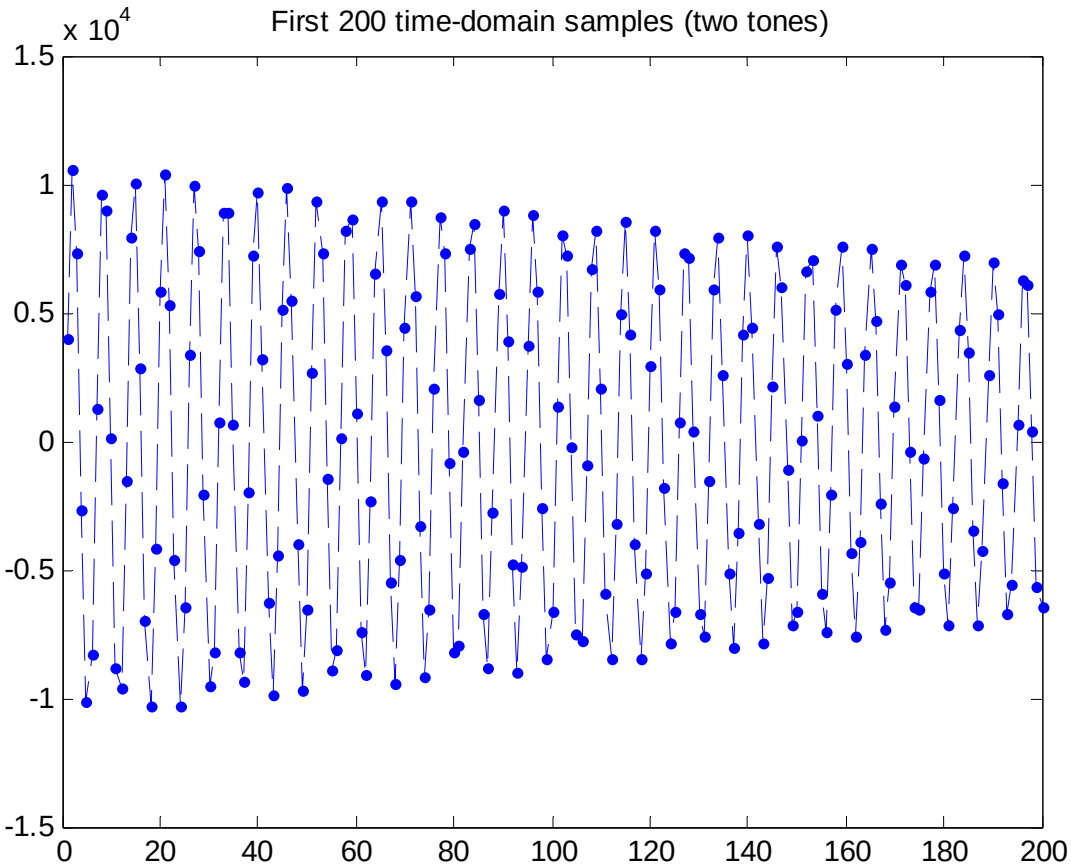
Example 2 (statistics):

$$E(\overline{X}) = \mu$$

$$sd\ (\overline{X}) = \frac{\sigma}{\sqrt{N}}$$

the more samples collected, the lower the standard error ***sd*** ***(square root improvement)***

Example 1: Two-tone test signal



Concept:
Original snapshot memory: 4k-pt
Samplify 2:1 enables 8k-pt FFT
Samplify 4:1 enables 16k-pt FFT

Example 1: Two tones

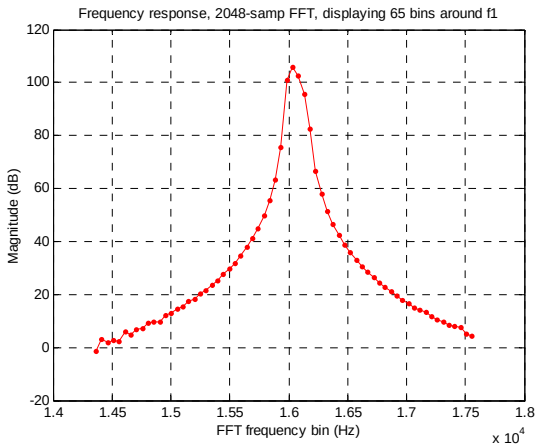
f1 = 15.95 MHz, mag = 8191
f2 = 16.05 MHz, mag = 4095
fs = 100 Msamp/sec, 16 bits/sample

Δ f: 16.05 – 15.95 MHz = 100 kHz

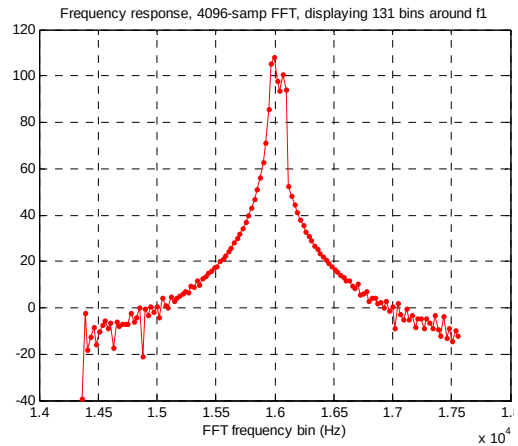
4k-pt FFT: 25 kHz resolution
8k-pt FFT: 12.5 kHz resolution
16k-pt FFT: 6.25 kHz resolution

Example 1: Improved FFT Resolution

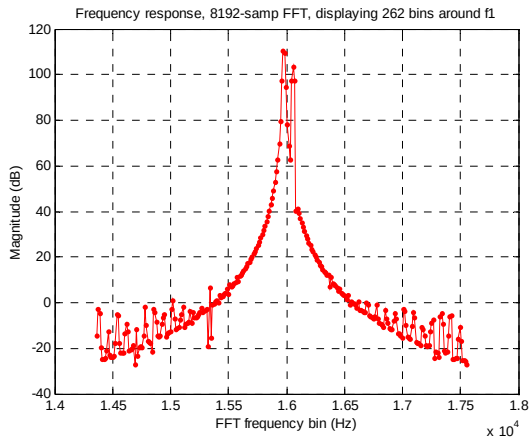
FFT resolution = f_s / N



N = 4096
before Samplify
FFT resolution:
100M / 4k = 25 kHz



N = 8192
Samplify @ 2:1
FFT resolution:
100M / 8k = 12.5 kHz



N = 16384
Samplify @ 4:1
FFT resolution:
100M / 16k = 6.25 kHz

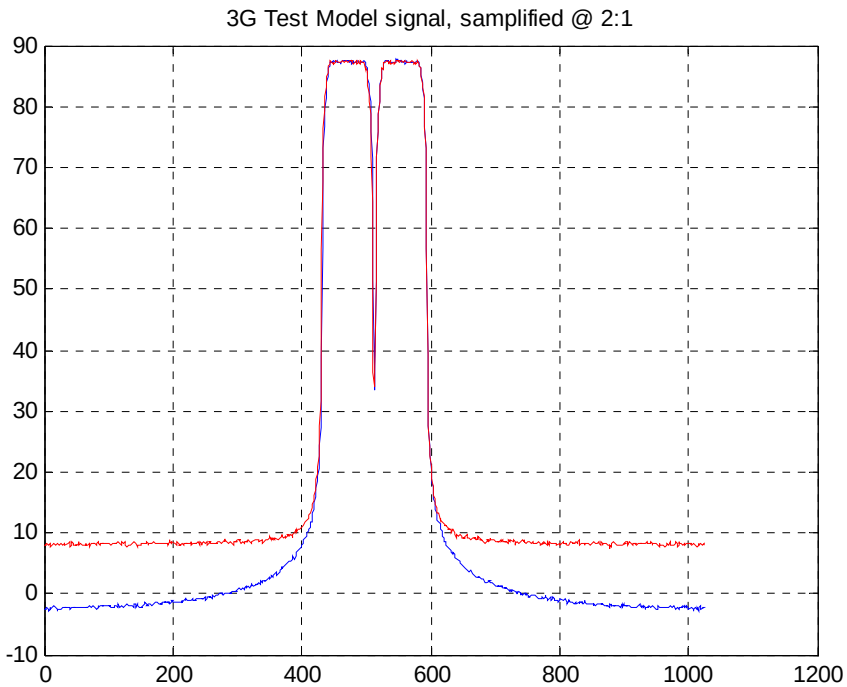
Samplify Systems

Example 2: Wider ARB Bandwidth

3G wireless ARB signal:

- Two-carrier downlink: Test Model 1
- 5 MHz/carrier (total 10 MHz BW)
- fs = 61.44 Msamp/sec (complex)
- 10 msec frame (614,400 samples)

SignalZIP (lossless)	**1.54:1**	**90 dB ACLR**
Samplify 2:1 lossy	**2.00:1**	**80 dB ACLR**



A Samplified ARB Test Model 1:

- ***stores 2 frames in the same physical memory, or***
- ***doubles stream-from-disk ARB bandwidth***

Samplify Increases ARB Thruput *(1 of 2)*

Product to be Samplified:

Agilent Baseband Studio for Waveform Streaming (N5110A)

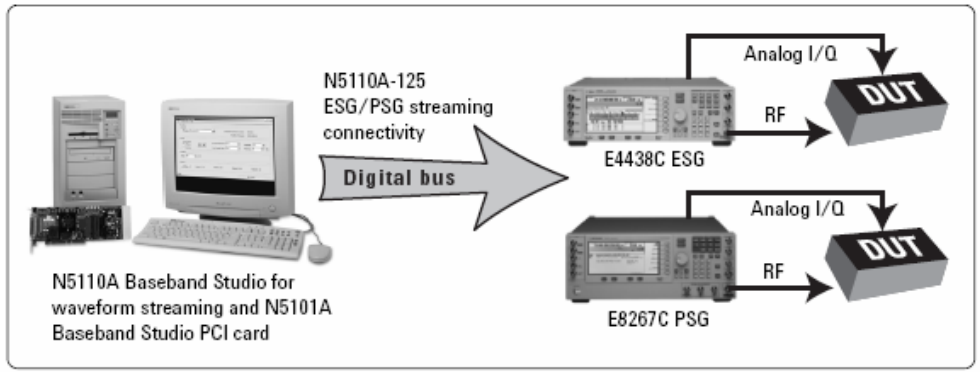


Figure 3. The Baseband Studio for waveform streaming solution

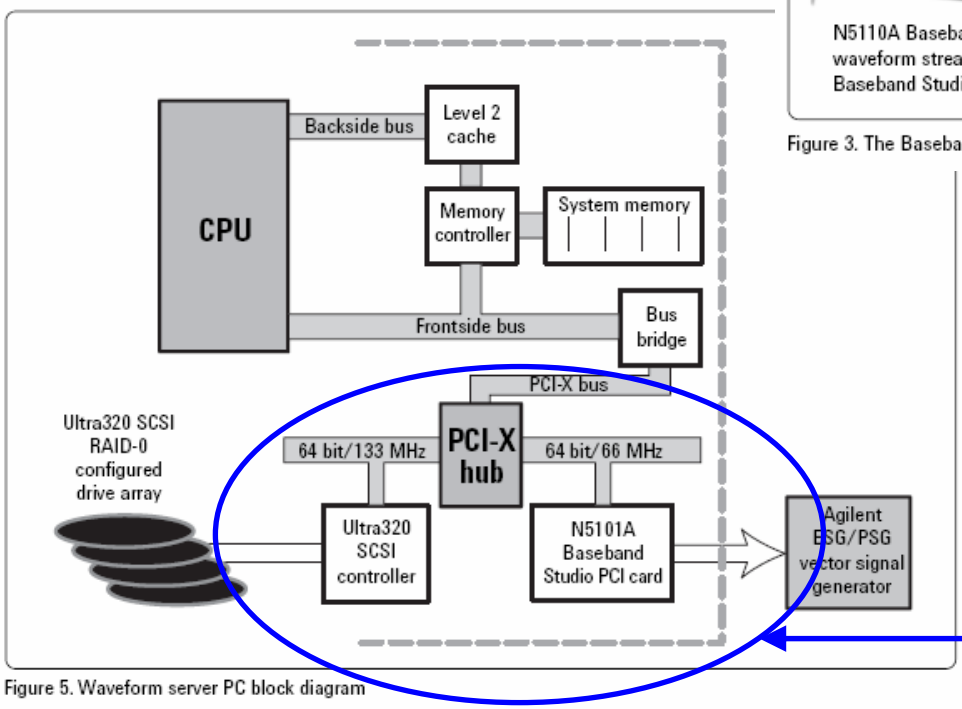


Figure 5. Waveform server PC block diagram

High data throughput

The Baseband Studio for waveform streaming solution can achieve data rates up to 40 MSa/s. This translates into 20 MHz of bandwidth for each of the I and Q channels and a total RF modulation bandwidth of 40 MHz. This wide bandwidth enables simulation of multiple channels of communication signals or generation of pulsed waveforms. The data rate is continuously displayed on the software graphical user interface (GUI) and is available on the API.

System bottleneck:

→ 160 MB/sec for 40 MHz bandwidth

Samplify Increases ARB Thruput *(2 of 2)*

Product to be Samplified: **Agilent Baseband Studio for Waveform Streaming (N5110A)**

Features and Specifications

Option	Maximum data rate	RF modulation bandwidth
Option 117	1 MSa/s	Up to 800 kHz
Option 118	5 MSa/s	Up to 4 MHz
Option 119	10 MSa/s	Up to 8 MHz
Option 120	20 MSa/s	Up to 16 MHz
Option 121	40 MSa/s	Up to 32 MHz
Example waveform playback time (4 bytes/sample)	50 s for a 4 GB file at 20 MSa/s 125 s for a 20 GB file at 40 MSa/s 5000 s for a 200 GB file at 10 MSa/s	
Markers	Selectable support for 0, 2, or 4 output markers	
Waveform resolution	16 bits without markers 15 bits with two markers set 14 bits with four markers set	

Samplify!

2x to 4x wider bandwidth!

2 to 4 Msa/s	1.6 to 3.2 MHz
10 to 20 Msa/s	8 to 16 MHz
20 to 40 Msa/s	16 to 32 MHz
40 to 80 Msa/s	32 to 64 MHz
80 to 160 Msa/s	64 to 128 MHz

Potential cost savings:

- **RAID → standard ATA drive**
- **PCI-X → standard PCI**
- **Smaller capacity disk drive**

Improving Legacy Products with Samplify

1. SDR equipment is **less expensive** if built using standard computer interfaces (PCI bus, Ethernet, USB 2.0, ATA/IDE drives, SCSI drives, SDRAM, Rambus RDRAM, etc.)

    - *National Instruments "virtual instruments"*
    - *"commercial, off-the-shelf" (COTS) trend for aerospace/defense*
    - *"synthetic instruments"; Agile Rapid Global Combat Support, Sep 2004* ; Synthetic Inst Working Group

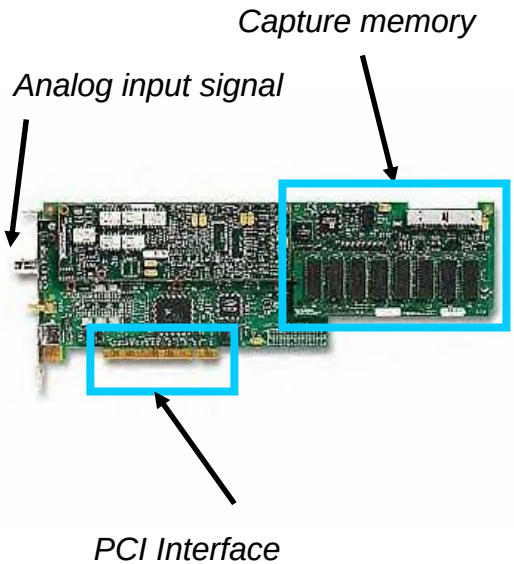
2. Standard computer interfaces have fixed, maximum thruput (MB/sec), which often (not always) become system bottlenecks.

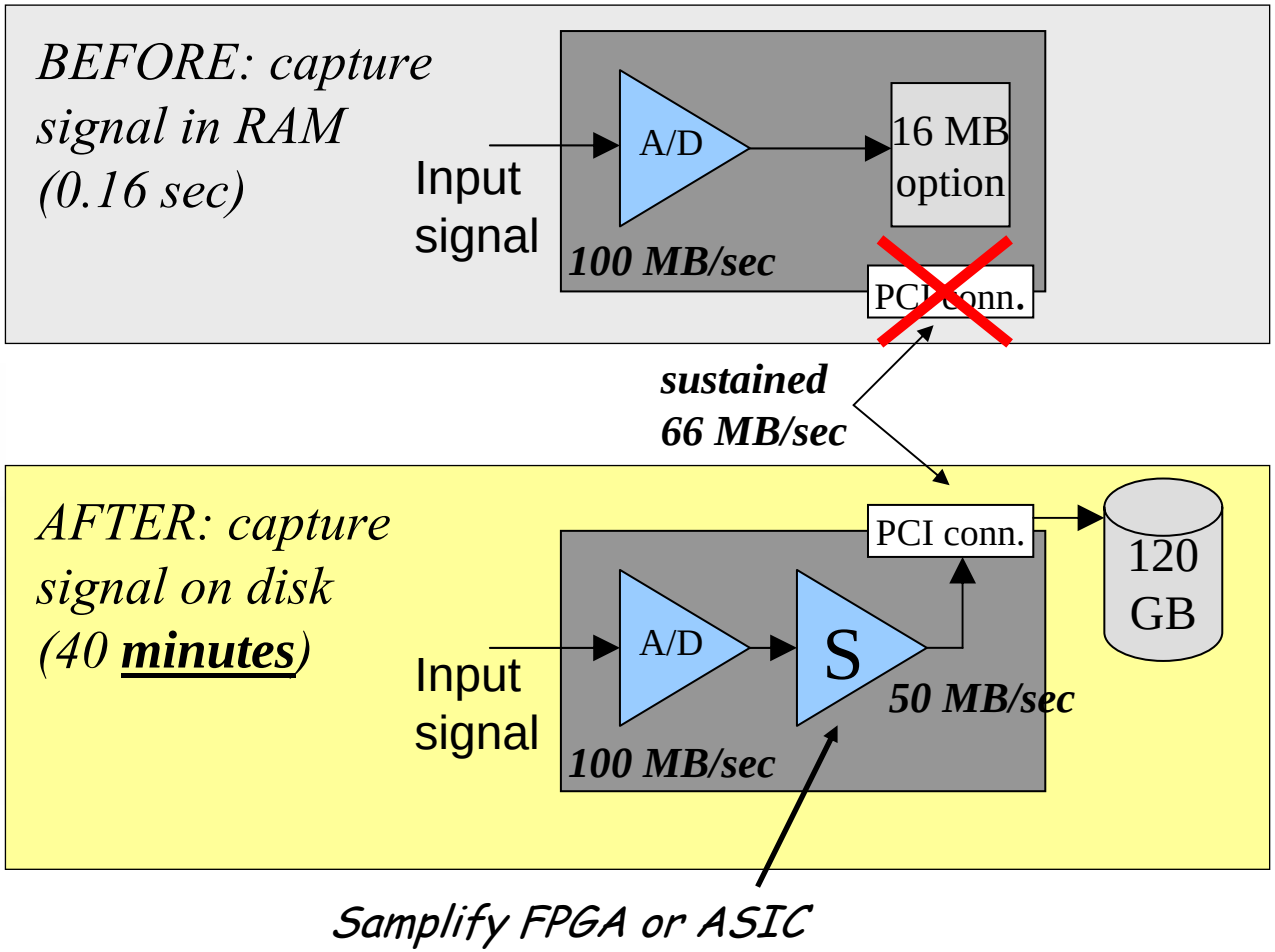
3. ***Samplify compression improves (by 1.3x to 6x) the thruput of all standard computer interfaces that carry sampled data.***

Example 3: Samplify Enables Streaming

Product to be Samplified:

National Instruments NI-5911 PC Data Capture Board

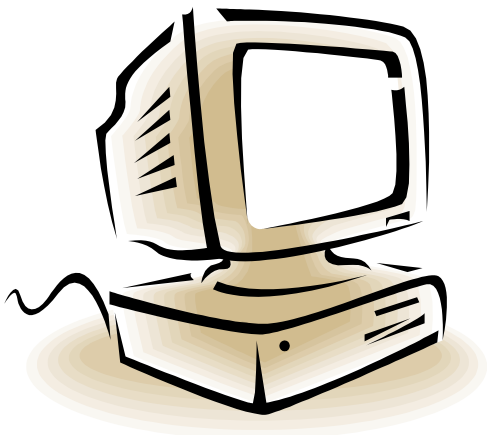




Samplify Software for PCs

Samplify Software

(identical results to Samplify ICs, just not as fast)



www.samplify.com

Samplify ***for Windows***

MATH BY MATLAB

LabVIEW

Availability:

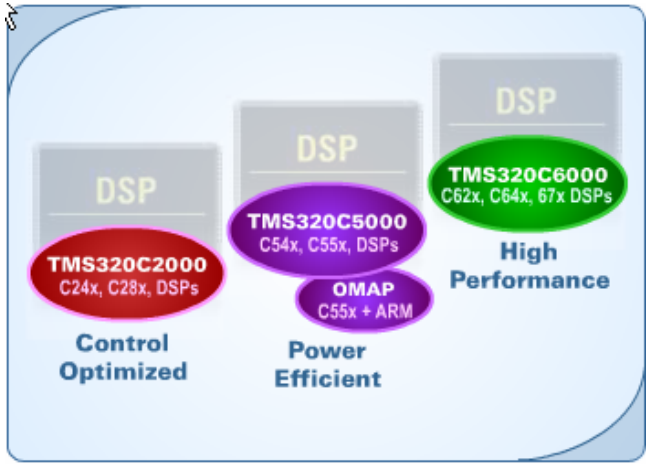
Now: $49

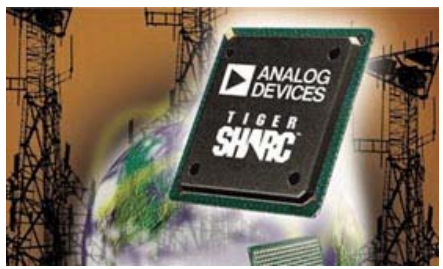
Now: $99

Dec 2005: $99

Samplify DSP Libraries

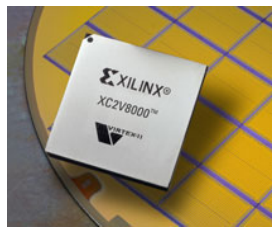
Samplify DSP Object Libraries

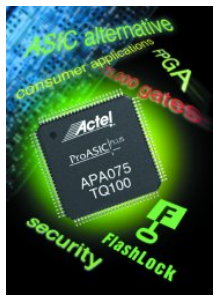


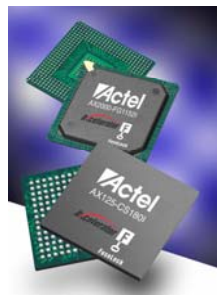


Samplify Encrypted FPGA Netlists

Samplify Encrypted FPGA Netlists







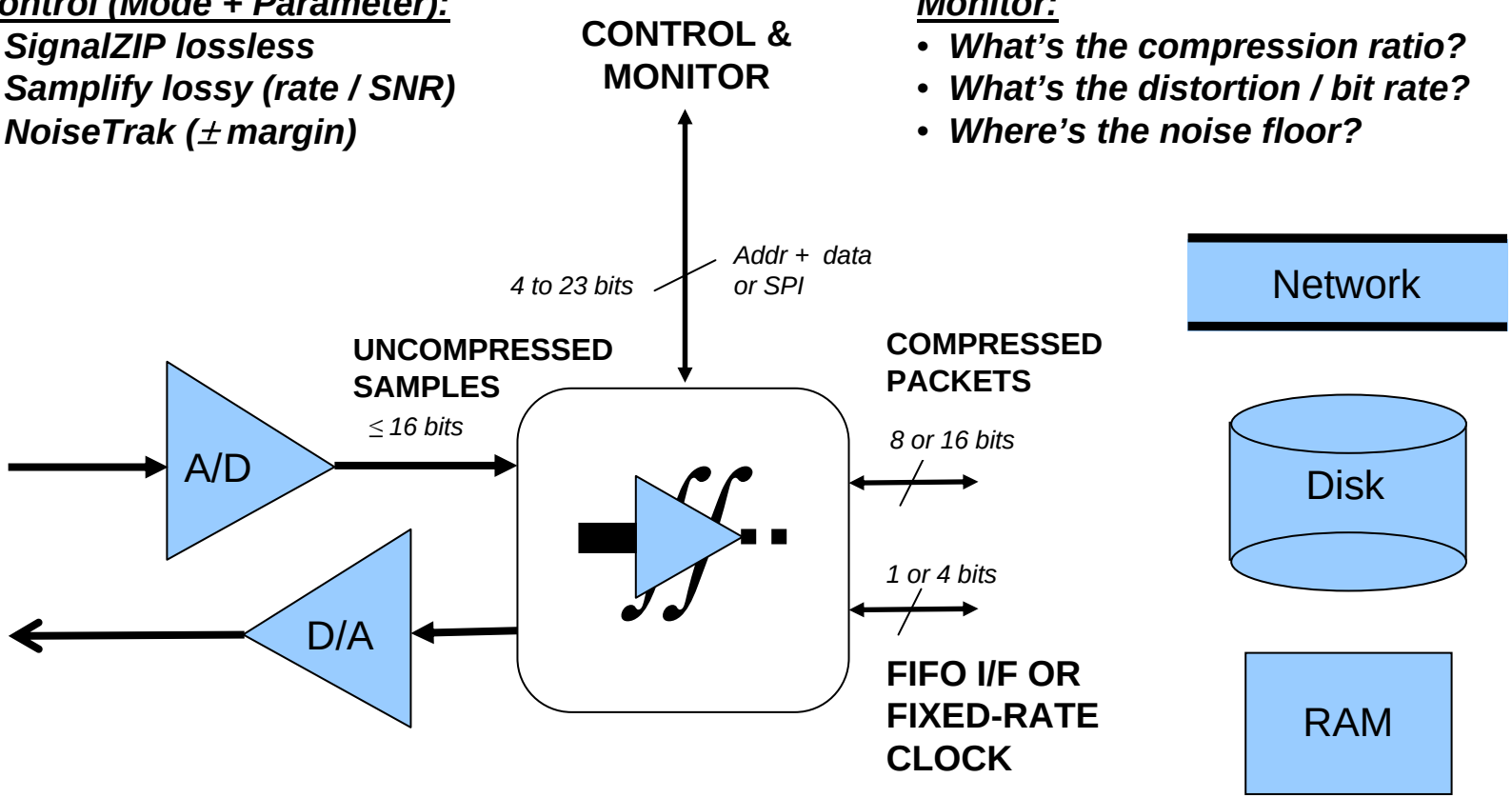
Available now for Xilinx, soon for Altera

- Flash-based
- Standard product
- Hard to reverse-engineer

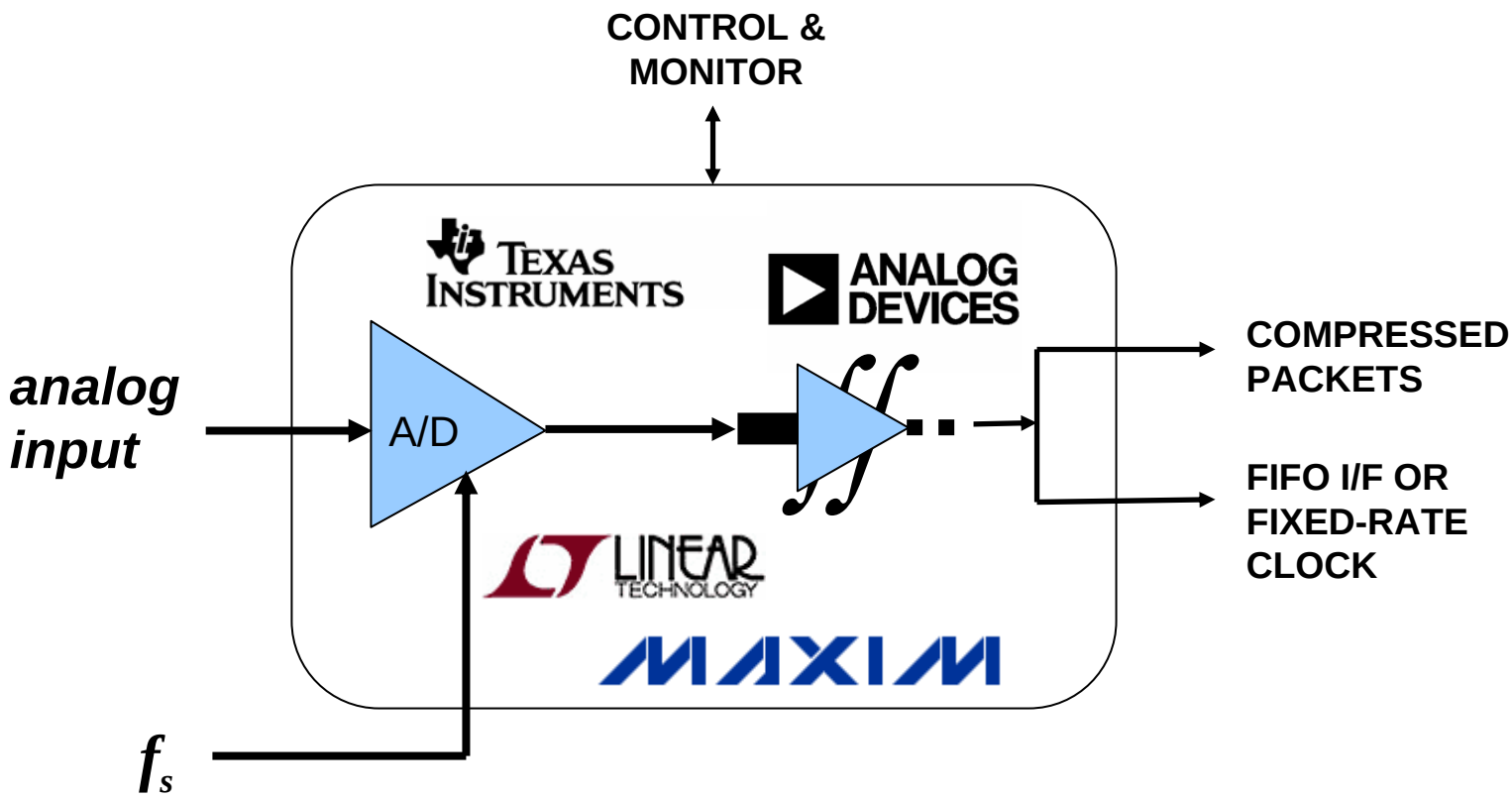
Samplify ASICs *(250+ Msps; low cost)*

Control (Mode + Parameter):
- ***SignalZIP lossless***
- ***Samplify lossy (rate / SNR)***
- ***NoiseTrak (± margin)***

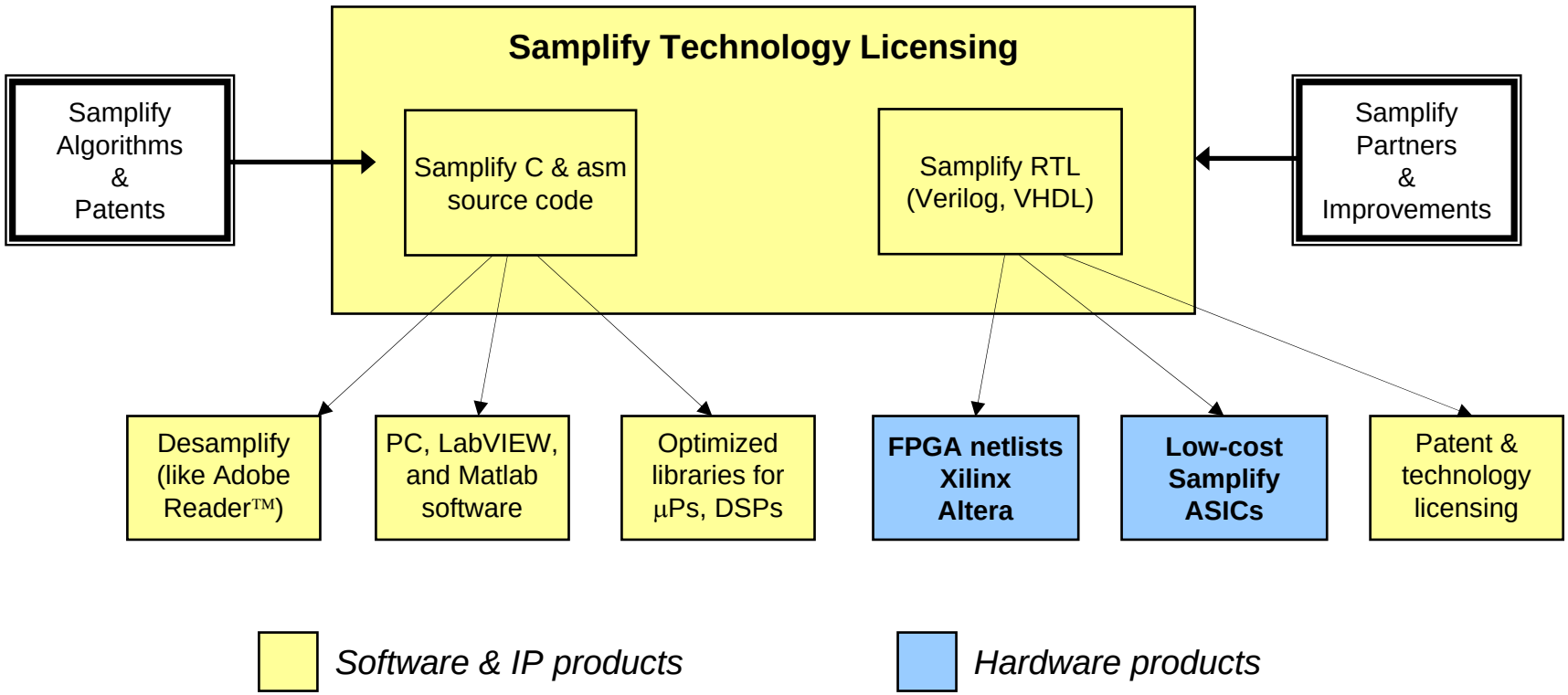
Monitor:
- ***What's the compression ratio?***
- ***What's the distortion / bit rate?***
- ***Where's the noise floor?***



Samplified Data Converters *(someday, we hope ☺)*



Samplify Product Family



Samplify Patents & Patent Apps

Granted Nov 1997

US 5,839,100: Lossless and loss-limited sampled data compression

Applied Feb 2004:

- App #1: Lossless and lossy bandlimited sampled data compression
- App #2: Enhanced test and measurement equipment using compression
- App #3: Enhanced data converters using compression

In process:

- 4 additional patent applications

Consider Samplify for SDR if…

- A standard bus/network has become your system bottleneck
- Your products process bandlimited, oversampled signals
- Your measurements are derived from imperfect (noisy) sources
- Your sampling rates preclude existing compression solutions
- You aren't sure how well your signal will compress
- You already use an FPGA or DSP to process each sample
- You could benefit from 2x to 6x lower network & storage costs

Visit our Web Site

www.samplify.com

End

Thank you!

...simply the bits that matter