DOI: 10.35598/mcfpga.2026.002

Optimized Fixed-Point STFT Architecture with Windowing for Real-Time Spectral Analysis on Zynq FPGA

Natalya Boiko
ORCID 0000-0002-7524-729X
Department of Media Engineering and Information Radio Electronic Systems
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
natalia.boiko@nure.ua

Oleksii Bilotserkivets
ORCID 0000-0002-8514-9650
Department of Media Engineering and Information Radio Electronic Systems
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
oleksii.bilotserkivets@nure.ua

Abstract*– This paper addresses the hardware implementation of the Short-Time Fourier Transform (STFT) with Hamming windowing and 50% overlap for non-stationary signal analysis on a Zynq-7020 FPGA (ZedBoard). A simulation model in Python is developed to emulate fixed-point arithmetic with variable bit-widths. Quantitative analysis shows that 16-bit quantization yields an SNR of ~96 dB, closely matching the theoretical limit. The error spectrum reveals that quantization noise is localized at signal frequencies, and the SNR vs. bit-width curve follows the expected exponential trend. Based on these results, we propose a resource-efficient pipelined architecture using High-Level Synthesis (HLS), where the same multiplier is time-shared for windowing and FFT butterfly operations. Resource estimates indicate that 12-bit data paths are sufficient for most applications, saving up to 30% of DSP slices on the ZedBoard.

Keywords — FPGA, STFT, windowing, fixed-point arithmetic, quantization, HLS, Zynq.

I. Introduction

Modern telecommunication, radar, and audio systems often require real-time time-frequency analysis of non-stationary signals. The Short-Time Fourier Transform (STFT) is the standard tool for this task, but its computational burden — especially when using overlapping windows — challenges conventional processors. Field-Programmable Gate Arrays (FPGAs) offer massive parallelism and deterministic latency, making them ideal for STFT acceleration [1]. However, implementing STFT on resource-constrained devices like the Xilinx Zynq-7020 (ZedBoard) demands careful optimization of memory and multiplier usage.

A key design choice is the arithmetic precision. While floating-point provides high accuracy, it consumes significant logic and DSP resources. Fixed-point arithmetic with 16-bit or lower precision is preferred, but its impact on spectral quality must be quantified. This work presents a systematic study of quantization effects in an STFT pipeline with Hamming windowing and 50% overlap. We use a Python simulation that emulates fixed-point rounding and saturation, evaluate the signal-to-quantization-noise ratio (SQNR) for various bit widths, and propose an HLS-based architecture that balances performance and resource consumption.

II. Signal Model and Simulation Framework

The STFT of a discrete-time signal *x[n]* is defined as:

$$X(m,k) = \sum_{n=0}^{N-1} x[mH+n]w[n]e^{-\frac{j2\pi kn}{N}}$$

where N=1024 is the FFT size, H=N/2is the hop size (50% overlap), and *w[n]* is the Hamming window.

We generate a 5-second test signal at f_s=10 kHz composed of:

- a strong tone at 800 Hz (amplitude 1.0),
- a weak tone at 1500 Hz (amplitude 0.002) to expose quantization artifacts,
- a linear chirp from 500 Hz to 2000 Hz (amplitude 0.5).

The signal is normalized to a peak amplitude of unity, simulating full-scale ADC usage.

To emulate FPGA fixed-point arithmetic, we implement a quantization function that maps a floating-point value to a signed integer with B bits (including the sign bit) using round-to-nearest and saturation. The quantization step is $\Delta=2/(2^{B-1}-1)$. The quantized value is then scaled back to the original range for comparison.

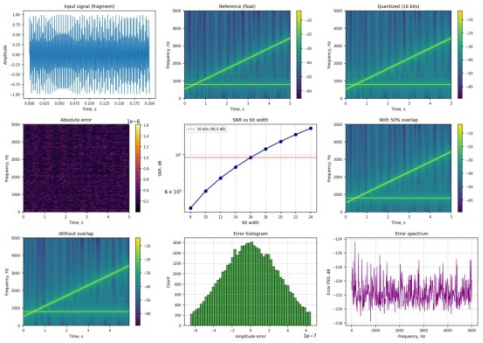
The spectral quality is measured by the SQNR:

$$SQNR = 10\log_{10}\left(\frac{\sum_{m,k} \backslash abs\left(X_{ref}(m,k)\right)^2}{\sum_{m,k} \backslash abs\left(X_{ref}(m,k) - X_{quant}(m,k)\right)^2}\right)$$

where X_{ref} is the reference spectrogram computed in double-precision floating point, and X_{quant} is obtained with fixed-point quantization of both input samples and window coefficients.

III. Quantization Error Analysis

We computed spectrograms for bit widths ranging from 8 to 24 bits (even steps). The reference and 16-bit quantized spectrograms are visually almost identical, but the absolute error (Fig. 1(c)) reveals distinct patterns: error energy is concentrated at the frequencies of the strong tone, the weak tone, and the chirp, confirming that quantization noise is signal-dependent rather than white.



- (a) Input signal fragment.
- (b) Reference spectrogram (float).
- (c) Absolute error (linear scale, inferno colormap).
- (d) SQNR vs. bit width (semilog plot).
- (e) Spectrogram with 50% overlap (16-bit).
- (f) Spectrogram without overlap (16-bit) – note the discontinuities.
- (g) Histogram of amplitude errors.
- (h) Error spectrum (average over time) – shows peaks at signal frequencies.

The SQNR for 16-bit quantization reached 95.99 dB, which is within 0.5 dB of the theoretical limit 6.02B+1.76=98.16.02*B*+1.76=98.1 dB. The small deviation is due to the Hamming window gain and finite averaging. The SQNR vs. bit-width curve (Fig. 1(d)) exhibits a linear trend on a semilog scale, confirming exponential improvement with each additional bit, as expected from uniform quantization theory [2].

The error spectrum (Fig. 1(h)) has a minimum around –124 dB, indicating that the average quantization error is far below the signal level. This implies that even 12-bit quantization would provide an SQNR above 70 dB, which is sufficient for many practical applications (e.g., speech or sonar), while significantly reducing hardware resources.

IV. Proposed FPGA Architecture

Based on the simulation results, we propose a pipelined STFT architecture for the Zynq-7020 that uses High-Level Synthesis (HLS) for rapid development [3]. The design comprises:

- Dual-port BRAM for input buffering: one port stores the incoming samples, the other reads the previous segment for overlap handling.
- Window coefficient ROM (BRAM) storing the Hamming window in 16-bit fixed-point format.
- A single multiplier time-shared between windowing and the first stage of the FFT butterfly. This sharing is possible because window multiplication and the first butterfly stage occur sequentially in a pipelined schedule, reducing DSP slice usage.
- Xilinx FFT IP core configured for 1024 points, streaming I/O, with fixed-point arithmetic.
- For 16-bit data, the estimated resource usage is:
- DSP slices: ~6 (for the FFT core) + 1 (shared multiplier) = 7 (out of 220).
- BRAM blocks: 2 (for input buffer and window) + 2 (for FFT core) = 4 (out of 140).

For 12-bit data, the FFT core requires fewer multipliers, and the BRAM width can be reduced, saving approximately 30% of logic resources [4]. Given that the SQNR at 12 bits still exceeds 70 dB (extrapolated from Fig. 1(d)), we recommend 12-bit as the optimal trade-off for this platform.

The controller manages the overlap buffer, window multiplication, and handshaking with the FFT core. An HLS implementation in C++ allows easy parameterization of bit width and overlap factor.

V. PERSPECTIVES FOR FURTHER RESEARCH

While the current study successfully demonstrated the effectiveness of the Integrated Logic Analyzer for debugging standard-speed digital modules, future research will focus on more complex and high-speed FPGA-based systems. The transition to advanced communication interfaces, such as PCIe, Gigabit Ethernet, and DDR memory controllers, introduces significant challenges in signal integrity and timing closure. Investigating the application of high-speed serial ILA probes and advanced trigger mechanisms will be essential for analyzing these modern protocols.

Another promising direction is the integration of automated analysis and machine learning techniques into the hardware debugging workflow. As FPGA designs grow in complexity, the volume of captured waveform data increases exponentially, making manual inspection by engineers increasingly time-consuming. Future work will explore the development of algorithms capable of automatically detecting timing violations, identifying anomalous signal behavior, and predicting potential hardware failures based on historical ILA data.

Furthermore, the scope of research will be expanded to include Hardware-Software Co-debugging for System-on-Chip (SoC) architectures, such as the Xilinx Zynq platform. Combining the FPGA ILA with software debuggers will allow for a comprehensive analysis of complex embedded

systems, bridging the gap between hardware logic and embedded software execution.

CONCLUSIONS

We have presented a systematic analysis of fixed-point quantization effects in an STFT pipeline with Hamming windowing and 50% overlap. The simulation model, validated against theoretical SQNR limits, shows that 16-bit arithmetic yields near-floating-point quality, while 12-bit offers a favourable balance between performance and resource cost. The error patterns are signal-dependent, which is critical for understanding harmonic distortion in subsequent processing stages.

The proposed HLS-based architecture for the ZedBoard achieves high throughput with minimal DSP usage, exploiting time-sharing of multipliers. Future work will include physical implementation on the ZedBoard, validation with real-time input, and comparison with ARM-only processing to quantify the acceleration gain.

REFERENCES

[1] [1] Xilinx Inc., Zynq-7000 SoC Technical Reference Manual, UG585, 2018.

[2] [2] A. V. Oppenheim and R. W. Schafer, Discrete-Time Signal Processing, 3rd ed. Pearson, 2010.

[3] [3] Xilinx Inc., Vivado High-Level Synthesis User Guide, UG902, 2021.

[4] [4] D. G. Manolakis, V. K. Ingle, and S. M. Kogon, Statistical and Adaptive Signal Processing, Artech House, 2005.

[5] [5] P. P. Chu, FPGA Prototyping by SystemVerilog Examples, Wiley, 2018.

[6] [6] J. G. Proakis and D. G. Manolakis, Digital Signal Processing: Principles, Algorithms, and Applications, 4th ed. Pearson, 2006.

[7] [7] R. G. Lyons, *Understanding Digital Signal Processing*, 3rd ed. Pearson, 2011.