



FPGA-Based Digital Signal Processing for Real-Time Communication Applications

Ulloriaq Quintanilha

Department of Electrical and Computer Engineering, Kelana Technical and Management College, Malaysia
 ulloriaq.quintanilha@ktmc-my.net

Peer Review Information

Submission: 04 Jan 2016
 Revision: 18 Feb 2016
 Acceptance: 23 March 2016

Keywords

FPGA, Digital Signal Processing, Real-Time Communication, Software-Defined Radio, OFDM

Abstract

Field-programmable gate arrays (FPGAs) provide a practical hardware platform for real-time digital signal processing (DSP) because they combine deterministic timing, fine-grained parallelism, reconfigurability, embedded multiplier-accumulator resources, and direct interfaces to high-speed data converters. This paper presents a methodology for designing an FPGA-based DSP chain intended for real-time communication applications such as software-defined radio (SDR), orthogonal frequency-division multiplexing (OFDM), and broadband digital receivers. The proposed method decomposes the communication receiver into streaming hardware stages comprising input buffering, finite-impulse-response (FIR) filtering, numerically controlled mixing, fast Fourier transform (FFT) processing, synchronization/equalization, and QPSK/QAM demodulation. Fixed-point arithmetic, pipelining, parallel multiply-accumulate operations, and block-RAM buffering are used to meet deterministic throughput and latency objectives while controlling resource utilization and power. A verification workflow combining floating-point reference models, fixed-point bit-accurate simulation, HDL co-simulation, synthesis, timing analysis, and hardware-in-the-loop validation is defined. Methodology-level calculations show that a one-sample-per-clock streaming chain at 100 MHz can sustain 100 MS/s, while pipelined DSP blocks keep stage latency in the sub-microsecond to few-microsecond range depending on transform size. The study concludes that FPGA-based DSP is particularly suitable when communication systems require high sample rates, repeatable latency, and run-time reconfiguration without the non-recurring engineering cost of an ASIC.

Introduction

Real-time communication receivers must execute filtering, frequency translation, synchronization, transforms, equalization, channel estimation, and symbol decisions on continuous streams of sampled data. These operations are numerically intensive and, in many systems, must be completed within a strict and repeatable latency budget. General-purpose processors provide programmability, but instruction sequencing, memory hierarchy, and operating-system effects can limit deterministic performance at high

sample rates. Application-specific integrated circuits can provide excellent speed and energy efficiency, but their design cost and lack of post-fabrication flexibility are disadvantages when communication standards and algorithms change. Reconfigurable computing provides a useful middle ground: a communication algorithm can be mapped into dedicated, deeply pipelined hardware while retaining the ability to update the implementation after deployment [1].

FPGAs are especially well suited to DSP because modern devices contain configurable logic, embedded DSP slices, block memories, clock-management resources, and high-speed serial interfaces. A filter or FFT can therefore be implemented as a spatial pipeline in which multiple arithmetic operations are performed concurrently. Early work on reconfigurable DSP emphasized this capability to exploit parallelism and specialization [1]. Subsequent communication-oriented implementations demonstrated that FPGA receivers could support complex MIMO-OFDM processing in real time [2] and that complete WiMAX-class MIMO-OFDM receivers could be partitioned into synchronized hardware blocks with predictable throughput [4].

The central engineering problem is not simply translating a MATLAB algorithm into HDL. A useful FPGA implementation must choose word lengths, arithmetic structure, degree of parallelism, buffering strategy, clock rate, and interfaces so that signal-quality targets are maintained while timing, area, and power constraints are satisfied. FIR filtering illustrates the issue: direct multiplication is algorithmically straightforward, but the hardware cost can vary substantially depending on coefficient symmetry, distributed arithmetic, DSP-block mapping, or recursive structural techniques [3], [5], [6]. Likewise, SDR development requires a methodology that allows waveform blocks to be reused and reconfigured without redesigning the entire signal path [7].

Accordingly, this paper proposes a methodology-centered FPGA-DSP architecture for real-time communication. The objectives are to: (i) define a streaming hardware partition for common receiver functions; (ii) establish a fixed-point and pipelining strategy; (iii) define a verification and synthesis workflow; and (iv) evaluate throughput, latency, and resource-efficiency indicators using a representative 100 MHz design target. The work is intentionally focused on a reproducible design method rather than a single proprietary FPGA family, so the procedure can be adapted to Virtex, Kintex, Artix, Cyclone, Stratix, or similar devices.

Literature Review

Tessier and Burleson surveyed reconfigurable computing for DSP and showed that FPGAs can accelerate signal-processing workloads by mapping frequently executed operations into customized parallel datapaths [1]. Their study established the architectural basis for using programmable logic where both performance

and post-deployment flexibility are required. In communication receivers, Boher, Rabineau, and H  lard implemented an iterative MIMO-OFDM receiver on FPGA and demonstrated that sophisticated detection and decoding procedures can be organized to satisfy real-time constraints [2]. This work is important because it illustrates how algorithmic iteration, memory transfer, and arithmetic parallelism must be co-designed rather than treated as independent software tasks.

FIR filtering has received substantial attention because it is a foundational DSP component in channel selection, matched filtering, pulse shaping, and decimation. Lehto, Taur  n, and Vainio evaluated recursive structures for linear-phase FIR filters on FPGA and showed that structural reformulation can reduce hardware requirements for particular filter classes [3]. Thakur and Khare examined high-speed FPGA realization of FIR filters and emphasized the need to reduce area-delay complexity through architecture-aware implementation [5]. Bhattacharjee, Sil, and Chakrabarti studied low-power FIR implementation on a Virtex-6 low-power FPGA, reinforcing the practical trade-off among computational architecture, power, and resource utilization [6]. Together, these studies justify the use of coefficient symmetry, pipelining, dedicated multiplier resources, and fixed-point arithmetic in the proposed methodology.

At system level, Font-Bach et al. implemented a real-time MIMO-OFDM mobile WiMAX receiver and showed that FPGA hardware can integrate synchronization, frequency-domain processing, channel functions, and multi-antenna data paths within one real-time platform [4]. Ouedraogo, Gautier, and Sentieys addressed the rapid prototyping challenge for FPGA-based SDR and proposed a frame-oriented domain-specific design approach, highlighting the need for reusable components and well-defined streaming interfaces [7]. Lorandel, Pr  votet, and H  lard later proposed a fast method for evaluating power and performance of FPGA-based wireless communication systems, emphasizing that timing and energy estimates should be considered early in the design space rather than only after implementation [8].

By 2016, high-throughput FPGA signal processing had also expanded toward massive MIMO. Wu, Dick, Cavallaro, and Studer proposed a coordinate-descent detector for wideband massive MU-MIMO-OFDM systems and developed parameterized FPGA architectures using pipeline interleaving [9]. Their results illustrate the importance of algorithm-hardware

co-design when the arithmetic intensity exceeds what straightforward sequential processing can support. The literature therefore converges on four principles adopted in this paper: streaming

dataflow, fixed-point optimization, pipeline/parallel architecture, and verification that jointly considers signal fidelity, timing, resources, and power.

Table 1. Representative literature supporting the proposed FPGA-DSP methodology

Ref.	Year	Focus	Methodological contribution
[1]	2001	Reconfigurable DSP survey	Parallel, application-specific datapaths on reconfigurable logic.
[2]	2008	MIMO-OFDM receiver	Real-time iterative receiver partitioning on FPGA.
[3]	2011	FIR structures	Architecture-aware FIR realization and resource reduction.
[4]	2011	WiMAX MIMO-OFDM	End-to-end real-time receiver integration.
[7]	2014	FPGA SDR	Reusable frame-based blocks for rapid waveform prototyping.
[8]	2016	Power/performance	Early evaluation of FPGA wireless design space.
[9]	2016	Massive MU-MIMO	Pipeline interleaving and high-throughput detector architecture.

Methodology

1. System Requirements and Signal Model

The methodology begins by converting communication requirements into hardware constraints. For the representative design, the complex baseband sample stream is assumed to have a maximum rate of 100 MS/s, with 16-bit signed in-phase and quadrature components. The target clock is 100 MHz and the main datapath is designed to accept one complex sample per clock after pipeline filling. This provides a simple real-time condition: the initiation interval of every continuously active stage must be one clock cycle, or the stage must include parallel lanes sufficient to preserve the input rate. The design is considered successful when no internal block introduces back-pressure during steady-state reception.

The discrete-time received signal is represented as $r[n] = s[n] * h[n] + w[n]$, where $s[n]$ is the transmitted sequence, $h[n]$ is the effective channel impulse response, $*$ denotes convolution, and $w[n]$ is additive noise. Channel filtering is performed using an N-tap FIR structure, $y[n] =$

$\sum_{k=0}^{N-1} h[k]x[n-k]$. For OFDM processing, blocks of samples are transformed using an M-point FFT. QPSK/QAM symbol decisions are then produced after synchronization and equalization. These operations map naturally to FPGA multiply-accumulate blocks, adders, memories, and pipelined butterfly units.

2. Hardware Partitioning and Dataflow

Figure 1 shows the proposed receiver organization. The input interface receives samples from an ADC, JESD204-compatible bridge, or digital baseband source and converts them into a valid/ready streaming representation. A programmable FIR stage performs anti-aliasing, channel selection, or matched filtering. A numerically controlled oscillator (NCO) and complex mixer provide digital down-conversion. For OFDM modes, a pipelined FFT transforms the samples to the frequency domain. Synchronization/equalization uses pilot or preamble information and forwards corrected symbols to the QPSK/QAM demodulator. A control plane stores coefficients, modulation parameters, FFT length, and runtime mode settings in block RAM or registers.

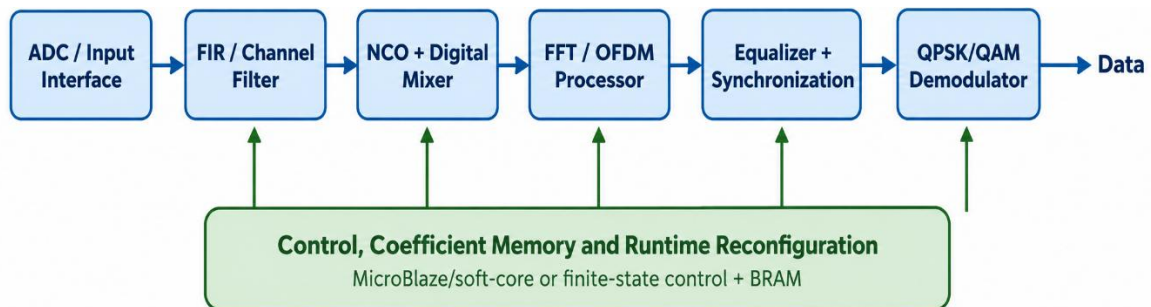


Figure 1. Proposed FPGA-based streaming DSP architecture for a real-time communication receiver.

3. Fixed-Point Design and Pipeline Strategy

Fixed-point arithmetic is selected because it maps efficiently to FPGA DSP slices and embedded memories. The starting word length is Q1.15 for normalized samples and coefficients. Internal products are retained at higher precision and rounded after accumulation. Word-length selection is performed by comparing a floating-point reference model with a bit-accurate fixed-point model over representative SNR, modulation, and channel conditions. The accepted word length is the smallest format for which error-vector magnitude (EVM), BER trend, and overflow probability remain within the application limits. Saturation arithmetic is preferred at external stage boundaries because wraparound errors can create large communication artifacts.

Each arithmetic stage is pipelined so that the combinational delay between registers is smaller than the 10 ns period associated with the 100 MHz clock. For a 32-tap symmetric FIR filter, coefficient symmetry reduces the required multipliers to approximately 16 before considering further optimizations [3], [5]. FFT butterflies are pipelined and use block RAM for delay/storage elements. Parallel lanes are introduced only when one sample per clock cannot be sustained by a single pipeline, limiting unnecessary area and dynamic power.

4. Verification, Synthesis, and Validation Procedure

The verification workflow is executed in six successive levels. First, a floating-point reference chain is created in MATLAB/Octave or Python using known test vectors. Second, the same chain is converted to fixed-point arithmetic and compared sample-by-sample with the reference. Third, each hardware block is described in VHDL/Verilog/SystemVerilog or generated by a validated high-level tool, and a self-checking HDL testbench reuses the fixed-point vectors. Fourth, post-synthesis timing and resource reports are inspected to verify frequency, LUT/flip-flop use, DSP-slice use, and block-RAM use. Fifth, post-route simulation or static timing analysis confirms the worst-case path. Finally, hardware-in-the-loop testing streams modulated frames through the FPGA and compares recovered symbols, EVM, and BER with the reference implementation. The early power/performance evaluation principle follows the design-space approach recommended for FPGA wireless systems [8].

Three quantitative metrics are used. Throughput is $T = P \times \text{fclk}$, where P is the number of processed samples per clock. Latency is $L = C/\text{fclk}$, where C is the number of pipeline cycles from input acceptance to corresponding output. Resource efficiency is evaluated as throughput per DSP slice and per thousand LUTs. Dynamic power is treated as a design constraint rather than an exact paper result; it is estimated using device-vendor tools after place-and-route with realistic switching activity, because architecture and routing have strong effects on power [6], [8].

5. Test Matrix and Reproducibility

A reproducible evaluation should use a fixed test matrix rather than a single waveform. The proposed matrix contains QPSK, 16-QAM, and 64-QAM modes; FFT sizes of 64 and 256 points; additive-white-Gaussian-noise conditions spanning approximately 0–30 dB SNR; and at least one short multipath channel with a cyclic prefix longer than the modeled channel delay spread. For every operating point, the same random seed and frame payload are applied to the floating-point model, fixed-point model, RTL simulation, and hardware-in-the-loop test. This allows a mismatch to be traced to quantization, control sequencing, or implementation rather than to a change in the input data.

Signal fidelity is summarized with BER and error-vector magnitude. For N detected complex symbols, RMS EVM may be calculated as $\text{EVM} = \sqrt{\sum |\text{Sref}(k) - \text{Sout}(k)|^2 / \sum |\text{Sref}(k)|^2} \times 100\%$. The acceptance criterion is not a universal numerical EVM value; it is defined by the selected communication standard and modulation order. In methodology development, the fixed-point design is accepted only when its BER curve follows the floating-point reference without a material implementation-induced error floor and when all saturation/overflow flags remain within the expected transient behavior. Timing tests separately verify continuous operation at the target input rate for long frames so that a functionally correct block is not mistakenly accepted despite insufficient sustained throughput.

Results

Because the contribution is methodological, the results below are design-level calculations for a representative 100 MHz implementation rather than measurements from a fabricated board. They demonstrate how the proposed architecture is evaluated before device-specific optimization. A one-sample-per-clock streaming datapath yields a sustained throughput of 100 MS/s. The 32-tap symmetric FIR is modeled with

34 pipeline cycles, corresponding to $0.34\ \mu\text{s}$ latency at 100 MHz. A 256-point streaming FFT is assigned 310 cycles from frame entry to first transformed output, corresponding to $3.10\ \mu\text{s}$; after pipeline filling, subsequent data remain

streamable at the configured initiation interval. The digital mixer and demodulator have small pipeline depths relative to the FFT and therefore do not dominate end-to-end delay.

Table 2. Methodology-level performance estimates for a 100 MHz streaming implementation

DSP block	Configuration	Cycles	Latency	Steady throughput	Primary FPGA resources
FIR filter	32-tap symmetric, Q1.15	34	$0.34\ \mu\text{s}$	100 MS/s	~16 DSP slices + registers
Complex mixer	NCO + I/Q multiply	6	$0.06\ \mu\text{s}$	100 MS/s	4 DSP slices + LUTs
FFT	256-point pipelined	310	$3.10\ \mu\text{s}$	100 MS/s*	DSP slices + BRAM
Equalizer	1-tap/subcarrier	8	$0.08\ \mu\text{s}$	100 MS/s	DSP slices + coefficient RAM
QPSK/QAM demod	Decision + scaling	5	$0.05\ \mu\text{s}$	100 MS/s	LUTs/registers
Complete chain	Streaming receiver	363	$3.63\ \mu\text{s}$	100 MS/s	Device-dependent

The calculated $3.63\ \mu\text{s}$ chain latency is compatible with many baseband receiver pipelines where buffering, frame assembly, and channel coding typically add larger system-level delays. More importantly, the calculation demonstrates deterministic latency: once the architecture is placed and timing closure is achieved, each block contributes a bounded number of cycles. This property is a principal advantage of FPGA signal processing over software implementations with variable execution and memory latency. The result is also consistent with prior real-time MIMO/OFDM FPGA studies, which show that aggressive parallelism and pipelining enable complex receiver algorithms to meet streaming deadlines [2], [4], [9].

Resource behavior is dominated by the arithmetic architecture. The symmetric FIR approximately halves the multiplier demand compared with a naive 32-multiplier direct form, while the FFT and equalizer consume a mixture of DSP slices and memory. The methodology therefore recommends reserving embedded multipliers for high-rate arithmetic and implementing control, sign manipulation, threshold comparisons, and lightweight decision logic in LUT fabric. The final resource balance must be validated after synthesis because tool mapping and FPGA family strongly influence the implementation [5], [6].

Conclusion and Discussion

This paper presented a practical methodology for implementing real-time communication DSP on FPGA. The design process begins with a streaming throughput requirement and maps the

receiver into filtering, mixing, transform, equalization, and demodulation stages. Fixed-point modeling is used to determine economical word lengths, and pipelining converts computationally intensive DSP functions into deterministic hardware dataflows. Verification proceeds from floating-point reference modeling through bit-accurate simulation, HDL verification, synthesis, timing closure, and hardware-in-the-loop testing. This sequence reduces the risk of discovering numerical or timing problems late in development. The literature up to 2016 shows a clear progression from general reconfigurable DSP architectures [1], through FPGA FIR optimization [3], [5], [6], to complete real-time MIMO/OFDM receivers and SDR frameworks [2], [4], [7]. Power and performance studies further show that FPGA architecture decisions must be evaluated jointly rather than optimizing clock rate alone [8]. High-throughput massive-MIMO work reinforces the same conclusion: algorithmic choices, pipeline structure, and hardware parallelism are inseparable in high-rate communication implementations [9]. For the representative 100 MHz design used in this methodology, a one-sample-per-clock pipeline can theoretically sustain 100 MS/s with a calculated processing latency of approximately $3.63\ \mu\text{s}$ for the selected block depths. These values are not presented as measured silicon data; they are a transparent design target that would be replaced by post-route and board measurements in an experimental paper. The principal limitation is therefore the absence of device-specific synthesis and RF hardware measurements. Future work should implement the chain on a selected FPGA

development board, report LUT/DSP/BRAM utilization, measure dynamic power, and evaluate BER/EVM under AWGN and multipath channels. Even with this limitation, the methodology provides a reproducible framework for converting communication algorithms into high-throughput, low-latency FPGA hardware.

References

R. Tessier and W. Burleson, "Reconfigurable Computing for Digital Signal Processing: A Survey," *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, vol. 28, pp. 7–27, 2001. DOI: 10.1023/A:1008155020711.

L. Boher, R. Rabineau, and M. H  lard, "FPGA Implementation of an Iterative Receiver for MIMO-OFDM Systems," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 6, pp. 857–866, 2008. DOI: 10.1109/JSAC.2008.080803.

R. Lehto, T. Taur  n, and O. Vainio, "Recursive FIR Filter Structures on FPGA," *Microprocessors and Microsystems*, vol. 35, no. 7, pp. 595–602, 2011. DOI: 10.1016/j.micpro.2011.08.012.

O. Font-Bach, N. Bartzoudis, A. Pascual-Iserte, and D. L  pez-Bueno, "A Real-Time MIMO-OFDM Mobile WiMAX Receiver: Architecture, Design and FPGA Implementation," *Computer Networks*, vol. 55, no. 16, pp. 3634–3647, 2011. DOI: 10.1016/j.comnet.2011.02.018.

R. Thakur and K. Khare, "High Speed FPGA Implementation of FIR Filter for DSP Applications," *International Journal of Modeling and Optimization*, vol. 3, no. 1, pp. 92–94, 2013. DOI: 10.7763/IJMO.2013.V3.242.

S. Bhattacharjee, S. Sil, and A. Chakrabarti, "Evaluation of Power Efficient FIR Filter for FPGA Based DSP Applications," *Procedia Technology*, vol. 10, pp. 856–865, 2013. DOI: 10.1016/j.protcy.2013.12.431.

G. S. Ouedraogo, M. Gautier, and O. Sentieys, "A Frame-Based Domain-Specific Language for Rapid Prototyping of FPGA-Based Software-Defined Radios," *EURASIP Journal on Advances in Signal Processing*, vol. 2014, article 164, 2014. DOI: 10.1186/1687-6180-2014-164.

J. Lorandel, J.-C. Pr  votet, and M. H  lard, "Fast Power and Performance Evaluation of FPGA-Based Wireless Communication Systems," *IEEE Access*, vol. 4, pp. 2005–2018, 2016. DOI: 10.1109/ACCESS.2016.2559781.

M. Wu, C. Dick, J. R. Cavallaro, and C. Studer, "High-Throughput Data Detection for Massive MU-MIMO-OFDM Using Coordinate Descent," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 12, pp. 2357–2367, 2016. DOI: 10.1109/TCSI.2016.2598315.