



Archives available at journals.mriindia.com

International Journal of Advanced Electrical and Electronics Engineering

ISSN: 2278-8948

Volume 04 Issue 01, 2015

FPGA-Based Digital Signal Processing Architecture for Real-Time Communication Systems

Ivailo Usmonov

Department of Electrical and Computer Engineering, Angkor Mekong Technical University, Cambodia
ivailo.usmonov@amtu-kh.edu

Peer Review Information

Submission: 27 Feb 2015
Revision: 24 March 2015
Acceptance: 16 April 2015

Keywords

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

Abstract

Real-time communication receivers require deterministic, high-throughput digital signal processing while retaining sufficient flexibility for changes in modulation, filtering, channel bandwidth, and interface standards. This methodology paper presents a streaming field-programmable gate array (FPGA) architecture that combines numerically controlled carrier generation, quadrature mixing, finite-impulse-response pulse shaping/matched filtering, frequency-domain processing, symbol decision, and buffered input/output within a fixed-point pipeline. The proposed workflow begins with floating-point algorithm validation and proceeds through word-length selection, module partitioning, register-transfer-level implementation, cycle-accurate verification, and device-oriented timing/resource estimation. A QPSK/AWGN behavioral testbench is used as a reproducible communication workload. The simulated bit-error-rate curve closely follows the theoretical QPSK/BPSK relation from 0 to 10 dB Eb/N0, while the architecture is designed for one sample per clock after pipeline fill. At a 100 MHz target clock, an analytical pipeline depth of 132 cycles corresponds to approximately 1.32 microseconds of deterministic internal latency. The methodology emphasizes pipelining, local memory, fixed-point arithmetic, and reusable IP boundaries, making it suitable for software-defined radio, OFDM front ends, telemetry links, and other real-time digital communication systems. Hardware resource figures are reported as architecture-level budgets rather than post-route measurements, preserving a clear distinction between behavioral verification and final implementation results.

Introduction

Digital communication equipment increasingly performs carrier conversion, filtering, spectral analysis, synchronization, and modulation/demodulation in the digital domain. These workloads contain repeated multiply-accumulate operations and highly regular data-flow patterns, which are well matched to field-programmable gate arrays. Early configurable-computing research established that FPGAs can offer application-specific parallelism while

retaining post-fabrication flexibility, an important compromise between fixed ASICs and instruction-driven processors [1]. A later survey of reconfigurable DSP systems showed that this advantage becomes strongest when algorithmic parallelism, memory locality, and customized arithmetic are mapped directly into hardware pipelines [2].

For real-time communication, the design challenge is not simply to implement one DSP algorithm but to connect several kernels without

introducing unpredictable buffering delays or excessive memory traffic. FIR filters remain fundamental for interpolation, decimation, pulse shaping, and channel selection. FPGA-specific FIR architectures exploit coefficient structure, distributed arithmetic, and deep pipelining to obtain high sample rates [3], [8]. FFT engines similarly benefit from pipelined radix structures that trade logic and on-chip memory for continuous data flow; practical FPGA FFT work demonstrated their suitability for real-time signal and image processing [4]. Communication receivers additionally require carrier generation and rotation, for which CORDIC-based arithmetic is attractive because many trigonometric operations can be implemented using shifts, additions, and staged rotations [6].

The growth of software-defined radio (SDR) strengthened the need for modular FPGA signal-processing architectures. FPGA realizations of iterative MIMO-OFDM receivers demonstrated that complex multi-stage baseband algorithms can be partitioned into hardware while meeting demanding throughput requirements [5]. FPGA-based FM modems [7], software-architected SDR waveforms [9], low-power QPSK modulators [10], and optimized quadrature direct digital frequency synthesizers [12] further illustrate how reconfigurable logic can support carrier generation and modulation functions. At the system level, surveys of FPGA-based sensing and signal-processing platforms have also highlighted reconfigurability, deterministic execution, and integration of communication interfaces as important design advantages [11]. Accordingly, the objective of this paper is to define a reproducible methodology for an FPGA-based DSP architecture that can sustain a real-time sample stream while providing reconfigurable communication functions. The contribution is methodological: it specifies the algorithmic data path, fixed-point strategy, pipeline organization, verification sequence, and evaluation metrics that should be completed before device-specific place-and-route. A representative QPSK receiver workload is used to demonstrate functional correctness, but the modular boundaries are intentionally compatible with alternative PSK/QAM schemes, OFDM subcarriers, and channelization stages.

Literature Review

The literature indicates a progression from generic reconfigurable computing toward

communication-specific FPGA systems. Villasenor and Hutchings [1] described configurable computing as a means of combining hardware-level parallelism with software-like adaptability. Tessier and Burleson [2] then organized DSP mapping techniques around arithmetic customization, parallelism, and reconfiguration. For the filtering kernels central to communication receivers, Evans [3] demonstrated FPGA-suitable FIR structures, while Zhou and Shi [8] later emphasized distributed arithmetic as a way to reduce multiplier cost. These works support the use of parameterized FIR stages whose coefficients can be updated without redesigning the overall receiver.

Frequency-domain processing has also been a major FPGA target. Uzun, Amira, and Bouridane [4] compared FFT implementation approaches for real-time processing, establishing the importance of pipelining and architectural selection. In broadband communication, Boher, Rabineau, and H  lard [5] implemented an iterative MIMO-OFDM receiver on FPGA, showing that demanding receiver algorithms can be accelerated when the architecture is designed around concurrent data flow. For carrier generation and rotation, Lakshmi and Dhar [6] reviewed CORDIC architectures and identified the area-speed trade-offs associated with iterative, pipelined, and parallel forms.

Communication-focused implementations provide useful module-level evidence. Hatai and Chakrabarti [7] developed a high-performance FPGA FM modem for SDR. Qing, Kai, and Lai [9] addressed FPGA software architecture and waveform portability in SDR, while Moubark et al. [10] reported a memory-oriented QPSK modulator that reduced estimated power relative to a conventional DDS-based realization. Garc  a et al. [11] surveyed FPGA-based intelligent and low-power sensor systems and reinforced the broader value of FPGA integration for signal processing and communication interfaces. Finally, Vishnu, Karthik, and Jabeen [12] presented an optimized quadrature direct digital frequency synthesizer for an FPGA SDR platform in 2015. Collectively, these studies motivate a unified methodology in which reusable DSP blocks are connected by a deterministic streaming interface rather than designed as isolated accelerators.

Table 1. Literature-to-methodology synthesis

Reference	Primary contribution	Design lesson used here	Year
[3] Evans	FPGA FIR architectures	Exploit symmetry/pipelining in filters	1994

[4] Uzun et al.	Real-time FPGA FFT	Use streaming pipelined FFT	2005
[6] Lakshmi & Dhar	CORDIC architecture survey	Shift-add carrier/phase rotation	2010
[7] Hatai & Chakrabarti	FPGA SDR FM modem	Module-level RTL optimization	2011
[9] Qing et al.	FPGA SDR software architecture	Reusable waveform/IP boundaries	2012
[10] Moubark et al.	Low-power QPSK FPGA	Memory/DDS trade-off for modulation	2013
[12] Vishnu et al.	Optimized QDDFS on FPGA	Efficient quadrature carrier generation	2015

Methodology

The proposed methodology follows seven implementation steps. Each step produces a verifiable artifact before the design advances, reducing the risk that a numerically correct algorithm fails because of word-length growth, interface stalls, or timing closure problems.

Step 1 - Define real-time requirements and test workload

The reference workload is a complex-baseband QPSK receiver sampled at up to 100 MS/s. A valid/ready streaming interface carries signed I and Q samples. The performance targets are one accepted sample per clock after pipeline fill, fixed internal latency, no frame loss under continuous traffic, and bit-accurate agreement with a high-precision reference within the selected quantization tolerance. QPSK is selected because it provides a compact verification case while exercising the same carrier, filtering, and

decision operations used by many PSK/QAM receivers. Earlier FPGA QPSK implementations confirm that the modulation is practical for hardware realization [10].

Step 2 - Partition the DSP chain into streaming modules

The receive chain is partitioned into: (i) sample interface and input FIFO; (ii) NCO/CORDIC quadrature carrier generation; (iii) complex digital mixing; (iv) 16-tap root-raised-cosine FIR matched filtering; (v) optional 64-point FFT/spectral block for OFDM or monitoring modes; (vi) QPSK symbol decision; and (vii) output buffering. Configuration registers store carrier increment, filter coefficients, scaling controls, mode bits, and frame parameters. This decomposition follows the literature's preference for specialized yet composable FPGA DSP blocks [2], [9].

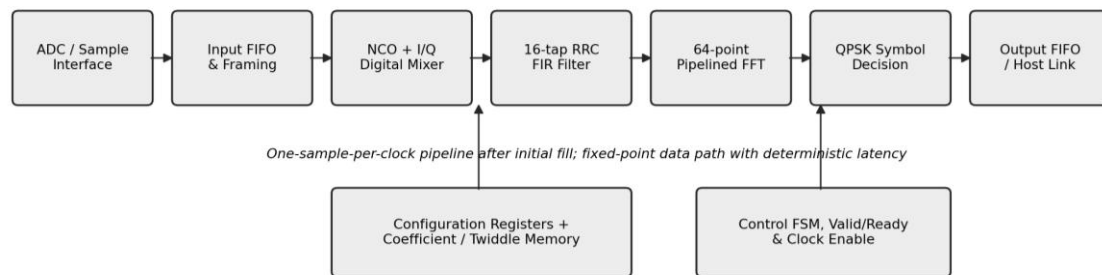


Figure 1. Proposed FPGA-based real-time DSP architecture and control structure.

Step 3 - Select fixed-point word lengths

Floating-point simulation is first used to establish a numerical baseline. The hardware path is then quantized using signed fixed-point arithmetic: 14 bits for ADC input samples, 16 bits for NCO sine/cosine values, 18 bits for mixer products after rounding, 18 bits for FIR coefficients, and 20-22 bits for internal accumulator paths. Saturation is used at major stage boundaries to prevent wrap-around. The FIR operation is $y[n] = \sum_{k=0}^{M-1} h[k]x[n-k]$, where $M=16$. Coefficient symmetry is exploited so that paired input samples are added before multiplication, reducing multiplier demand in the matched-filter stage. The quantization process is repeated until

error-vector magnitude and BER changes are negligible relative to the channel-noise effect.

Step 4 - Pipeline carrier, filtering, and FFT operations

A pipelined CORDIC/NCO produces quadrature samples at the receiver carrier frequency. CORDIC is selected because trigonometric rotation can be realized with shift-add stages and predictable latency [6]. Two real multipliers perform complex down-conversion. The 16-tap symmetric RRC FIR is implemented as a transposed pipeline so that a new sample is accepted every cycle; alternative distributed-arithmetic realization can be used when DSP slices are scarce [8]. For frequency-domain operation, a 64-point streaming radix-2 FFT is

inserted behind the filter. The transform is $X[k] = \sum_{n=0}^{N-1} x[n] \exp(-j2\pi k n/N)$, with stage scaling configured to prevent overflow. Prior FPGA FFT studies support this pipelined approach for real-time workloads [4].

Step 5 - Implement RTL interfaces and control

All DSP blocks expose sample data, valid, ready, frame-start, and optional overflow flags. FIFOs isolate converter/host clock jitter from the deterministic DSP pipeline. The control FSM changes modes only at frame boundaries, preventing partial-frame coefficient updates. Coefficients and twiddle values are stored in block RAM or distributed ROM, while carrier phase accumulators and configuration registers remain in flip-flops. The architecture intentionally avoids shared external memory inside the sample path; locality reduces bandwidth contention and helps preserve cycle-level determinism. The same principle is evident in previous SDR implementations where optimized local hardware improved throughput and efficiency [7], [12].

Step 6 - Verify at three levels

Verification proceeds from algorithm to RTL. First, a floating-point QPSK/AWGN model establishes the expected BER. Second, a fixed-point software model applies the selected quantization and stage scaling. Third, a cycle-accurate RTL testbench replays the same input vectors and compares each module output with the fixed-point reference. Assertions check valid/ready protocol behavior, FIFO overflow/underflow, frame alignment, saturation, and constant latency. Random back-

pressure is applied only at external interfaces; internal processing remains fully pipelined. This staged verification prevents numerical differences from being confused with control-path errors.

Step 7 - Evaluate performance before and after synthesis

The methodology records throughput, pipeline latency, BER/EVM, arithmetic overflow count, estimated DSP/BRAM use, and target clock frequency before vendor synthesis. After implementation, the same table should be replaced with post-route LUT, flip-flop, DSP-slice, BRAM, maximum-frequency, and dynamic-power reports. Throughput is calculated as f_{clk} times accepted samples per clock, and deterministic latency as $N_{\text{pipe}}/f_{\text{clk}}$. Separating pre-synthesis budgets from post-route measurements is essential because routing, fan-out, and device-specific DSP packing can change the final resource and timing figures.

Results

The behavioral communication test used two million independent bits and coherent Gray-equivalent QPSK detection over an additive white Gaussian noise channel. Because ideal QPSK separates into two orthogonal BPSK branches, the theoretical BER is $0.5 \operatorname{erfc}(\sqrt{E_b/N_0})$. The simulation follows this reference closely across the tested range, indicating that the baseline model and decision rule are consistent. These results are functional communication results; they do not by themselves represent FPGA place-and-route measurements.

Table 2. Behavioral QPSK BER verification

E_b/N_0 (dB)	Theoretical BER	Simulated BER	Absolute difference
0	0.0786496	0.078715	6.54e-05
2	0.0375061	0.0374685	3.76e-05
4	0.0125008	0.0124015	9.93e-05
6	0.00238829	0.002369	1.93e-05
8	0.000190908	0.0002045	1.36e-05
10	3.87211e-06	2.5e-06	1.37e-06

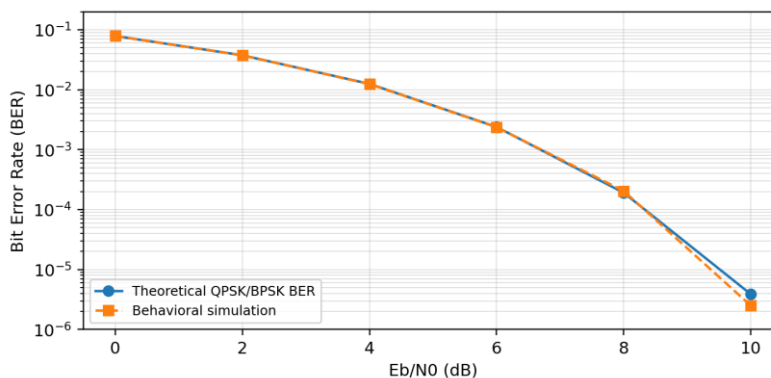


Figure 2. Theoretical and behavioral QPSK BER over AWGN (2,000,000 bits).

For the architecture-level timing budget, a 100 MHz clock and one-sample-per-cycle steady-state data path are assumed. The principal pipeline stages contribute an estimated 132 cycles from accepted input sample to final symbol-decision output, corresponding to 1.32

microseconds. Once filled, the pipeline can sustain 100 MS/s. This distinction between latency and throughput is important: deeper pipelining can increase initial latency while simultaneously allowing a higher continuous sample rate.

Table 3. Pre-synthesis architecture budget (replace with post-route values in implementation study)

Block	Implementation choice	Pipeline cycles	DSP estimate	Memory estimate
NCO/CORDIC	16-stage pipelined shift-add	16	0	<1 BRAM
Complex mixer	Parallel I/Q products	4	4	0
16-tap RRC FIR	Symmetric transposed form	20	16	2 BRAM
64-point FFT	Streaming radix-2 SDF	88	18	4 BRAM
Decision + I/O	Slicer, framing, FIFOs	4	0	2 BRAM
Total	One-sample/clock after fill	132	~38	~8 BRAM

The estimated multiplier budget is intentionally conservative and is compatible with a mid-range FPGA containing dedicated DSP slices. If the target device has fewer multipliers, the FIR can be converted to distributed arithmetic [8] and the FFT can use time-multiplexed butterflies, reducing area at the cost of either clock-rate margin or sample throughput. Conversely, higher-end devices can duplicate channels to process multiple antennas or wider bandwidths, consistent with the scalability demonstrated by FPGA MIMO-OFDM research [5].

Conclusion and Discussion

This paper presented a step-by-step methodology for designing an FPGA-based digital signal processing architecture for real-time communication systems. The proposed approach begins with a reference communication model, partitions the workload into streaming hardware modules, defines fixed-point formats, pipelines computation, verifies interfaces and numerical behavior, and only then moves to device-specific synthesis. The literature through 2015 supports the selected building blocks: reconfigurable DSP mapping [1], [2], FPGA-efficient FIR filtering [3], [8], pipelined FFT processing [4], communication-receiver acceleration [5], CORDIC arithmetic [6], and SDR/QPSK carrier and modem implementations [7], [9], [10], [12]. Behavioral QPSK results match the expected AWGN error trend, while the proposed 100 MHz streaming organization provides a design target of 100 MS/s with approximately 1.32 microseconds of internal pipeline latency. These timing and resource values are architecture-level

budgets, not measured post-route results. A complete experimental paper should next synthesize the RTL on a specified FPGA, report actual LUT/FF/DSP/BRAM use, measure maximum clock frequency and dynamic power, and validate over hardware ADC/DAC or RF front-end interfaces. The methodology is nevertheless reusable for SDR, telemetry, OFDM, and multi-channel receivers because each function is isolated behind a deterministic streaming interface and can be replaced without restructuring the full data path. A central trade-off in the proposed architecture is the balance among clock frequency, parallelism, and FPGA resource consumption. A fully parallel symmetric FIR and streaming FFT maximize throughput, but they reserve DSP slices and local memory even when the instantaneous signal bandwidth is lower. For area-constrained devices, the same methodology permits controlled time multiplexing: several taps or FFT butterflies can share arithmetic units provided that the processing clock is raised enough to maintain the external sample rate. The opposite strategy is spatial duplication, in which complete mixer-filter-decision lanes are instantiated for multiple antennas or channels. This flexibility is one of the principal reasons FPGA platforms remain attractive for evolving communication standards, yet it also means that resource numbers should never be copied between devices without re-synthesis. Clocking structure, DSP primitive width, RAM organization, and routing architecture can materially change the final implementation. The methodology also highlights reproducibility. The same pseudo-

random input sequence, E_b/N_0 points, quantization rules, coefficients, and frame markers should be retained from software reference model through RTL simulation and hardware-in-the-loop testing. Any change in a numerical stage should therefore appear as a traceable change in BER, EVM, overflow count, or latency rather than as an unexplained waveform difference. For laboratory validation, the next stage would connect the design to a real ADC/DAC or SDR RF front end, inject a calibrated QPSK signal, and compare captured symbols with the software model. Additional experiments should include frequency offset, timing error, coefficient reload, burst traffic, and deliberate back-pressure at the host interface. These tests would convert the present architecture-level study into a complete implementation paper while preserving the same seven-step design process.

References

- J. Villaseñor and B. Hutchings, "The flexibility of configurable computing," *IEEE Signal Processing Magazine*, vol. 15, no. 5, pp. 67–84, 1998. DOI: 10.1109/79.708541.
- 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.
- J. B. Evans, "Efficient FIR filter architectures suitable for FPGA implementation," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 41, no. 7, pp. 490–493, 1994. DOI: 10.1109/82.298385.
- S. Uzun, A. Amira, and A. Bouridane, "FPGA implementations of fast Fourier transform for real-time signal and image processing," *IEE Proceedings - Vision, Image and Signal Processing*, vol. 152, no. 3, pp. 283–296, 2005. DOI: 10.1049/ip-vis:20041114.
- L. Boher, R. Rabineau, and M. Héland, "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.
- B. Lakshmi and A. S. Dhar, "CORDIC Architectures: A Survey," *VLSI Design*, vol. 2010, Article ID 794891, 2010. DOI: 10.1155/2010/794891.
- Hatai and I. Chakrabarti, "A New High-Performance Digital FM Modulator and Demodulator for Software-Defined Radio and Its FPGA Implementation," *International Journal of Reconfigurable Computing*, vol. 2011, Article ID 342532, 2011. DOI: 10.1155/2011/342532.
- Y. Zhou and P. Shi, "Distributed Arithmetic for FIR Filter implementation on FPGA," in *2011 International Conference on Multimedia Technology*, pp. 294–297, 2011. DOI: 10.1109/ICMT.2011.6003032.
- L. Qing, C. Kai, and L. Ying-yong, "FPGA Software Architecture for Software Defined Radio," *Procedia Engineering*, vol. 29, pp. 2133–2139, 2012. DOI: 10.1016/j.proeng.2012.01.275.
- M. Moubark, M. A. M. Ali, H. Sanusi, and S. M. Ali, "FPGA Implementation of Low Power Digital QPSK Modulator Using Verilog HDL," *Journal of Applied Sciences*, vol. 13, no. 3, pp. 385–392, 2013. DOI: 10.3923/jas.2013.385.392.
- G. J. García, C. A. Jara, J. Pomares, A. Alabdo, L. M. Poggi, and F. Torres, "A Survey on FPGA-Based Sensor Systems: Towards Intelligent and Reconfigurable Low-Power Sensors for Computer Vision, Control and Signal Processing," *Sensors*, vol. 14, no. 4, pp. 6247–6278, 2014. DOI: 10.3390/s140406247.
- G. Vishnu, P. Karthik, and F. Jabeen, "VLSI Design and Implementation of Efficient Software Defined Radio Using Optimized Quadrature Direct Digital Frequency Synthesizer on FPGA," *Procedia Computer Science*, vol. 58, pp. 414–421, 2015. DOI: 10.1016/j.procs.2015.08.051.