



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

Indivar Varathan

Department of Computer Science and Engineering, Delta Polytechnic Institute of Engineering, Bangladesh
indivar.varathan@dpi-bd.edu

Peer Review Information

Submission: 06 Jan 2015

Revision: 10 Feb 2015

Acceptance: 13 March 2015

Keywords

*FPGA, Digital Signal
Processing, Real-Time
Communication, OFDM,
Fixed-Point Architecture*

Abstract

Field-programmable gate arrays (FPGAs) provide a practical platform for real-time digital signal processing (DSP) in communication systems because they combine deterministic timing, fine-grained parallelism, dedicated arithmetic resources, and reconfigurability. This methodology paper proposes a streaming FPGA architecture that integrates digital down conversion, multirate filtering, finite-impulse-response (FIR) channel filtering, fast Fourier transform (FFT)/inverse FFT processing, coordinate rotation digital computer (CORDIC)-based synchronization or equalization, and quadrature amplitude modulation (QAM) mapping/demapping. The methodology emphasizes fixed-point design, pipeline balancing, resource sharing, buffer sizing, and verification against floating-point reference models. A 64-point communication-oriented signal path with 16-bit I/Q samples and an assumed 125 MHz implementation clock is used as the design case. Bit-true numerical tests show an FFT quantization error vector magnitude (EVM) of -71.69 dB and FIR output signal-to-quantization-noise ratio (SQNR) of 91.60 dB, while an initiation interval of one sample per clock provides a theoretical streaming capacity of 125 MS/s. At a 20 MS/s application requirement this corresponds to a 6.25-times throughput margin. These results indicate that a carefully pipelined fixed-point FPGA DSP architecture can meet low-latency real-time communication requirements while retaining flexibility for OFDM, software-defined radio, and broadband baseband processing.

Introduction

Contemporary wireless and wired communication receivers execute large numbers of arithmetic operations within strict symbol and sample deadlines. Channel filtering, decimation, carrier recovery, frequency-domain transformation, channel equalization, modulation, and framing are often required in the same baseband chain. General-purpose processors and conventional programmable DSPs offer high software flexibility, but their sequential execution model can become a

bottleneck when sample rates increase or when many independent operations must be completed simultaneously. FPGA devices address this problem by allowing deeply pipelined and spatially parallel signal-processing structures to be implemented directly in reconfigurable hardware. Meyer-Baese (2014) describes this mapping of FIR filters, multirate processing, Fourier transforms, and communication algorithms to FPGA logic as a central advantage of FPGA-based DSP.

FPGAs are particularly suitable for real-time communication because the datapath can be

organized around the required sampling rate rather than around an instruction stream. A multiply-accumulate operation can be distributed across dedicated DSP blocks, coefficient memories can be implemented in block RAM, and independent stages can operate concurrently. For FIR filtering, distributed-arithmetic and systolic structures have demonstrated effective area-speed trade-offs (Meher, Chandrasekaran, & Amira, 2008; Xie, He, & Tan, 2010). Nandal, Vigneswarn, Rana, and Dhaka (2015) further showed that lookup-table-oriented distributed arithmetic can reduce hardware use in long parallel FIR filters, reinforcing the usefulness of FPGA-specific arithmetic transformations rather than direct software-style implementations.

Communication-oriented FPGA research also demonstrates the importance of FFT and CORDIC structures. Harikrishna, Rama Rao, and Labay (2010) implemented an FFT architecture for an IEEE 802.16d OFDM system, while Andraka (1998) established CORDIC as a hardware-efficient shift-add approach for rotations, magnitude/phase evaluation, and related functions. Canet, Valls, Almenar, and Marín-Roig (2012) combined fixed-point analysis, synchronization, demodulation, equalization, and resource reuse in an FPGA WLAN receiver. Wu and Shiue (2014) later demonstrated a CORDIC-based OFDM receiver prototype with real-time 64-QAM operation. These studies motivate a unified methodology in which arithmetic precision, pipeline timing, and resource reuse are treated as coupled design variables.

The objective of this paper is therefore to define a reproducible methodology for designing an FPGA-based DSP architecture for real-time communication systems. The proposed approach is not tied to one radio standard. Instead, it uses a modular streaming datapath that can support OFDM/WLAN/WiMAX-like functions and software-defined radio (SDR) concepts. The work focuses on architecture selection, fixed-point conversion, latency and throughput planning, verification, and implementation metrics. Because no physical FPGA board synthesis is claimed in this methodology study, the reported numerical results are explicitly limited to bit-true fixed-point simulations and architecture-level timing calculations; device utilization and post-route power are reserved for the final hardware implementation stage.

Literature Review

Early FPGA DSP work emphasized converting algorithms into structures that exploit bit-level parallelism rather than reproducing processor-oriented computation. Andraka (1998) surveyed CORDIC implementations for FPGA-based computers and showed how trigonometric and vector operations can be realized with shifts, additions, and pipeline registers. This is highly relevant to communication systems because carrier-phase correction, vector magnitude, arctangent estimation, and complex rotation recur in synchronization and demodulation blocks. The CORDIC principle reduces dependence on general multipliers and can be time-shared when throughput permits.

Filtering is another dominant FPGA workload. Meher et al. (2008) developed distributed-arithmetic systolic FIR implementations and demonstrated that LUT address partitioning can trade memory against adders, latency, and power. Xie et al. (2010) extended the idea with fully pipelined architectures for high- and medium-speed FIR filtering, reporting that pipeline registers and partial-table organization strongly affect frequency, area, and power. Nandal et al. (2015) examined a 256-tap parallel FIR implementation using distributed arithmetic and reported substantial reductions in LUT, slice, and gate usage compared with conventional structures. Collectively, these studies show that FIR structure must be chosen according to available DSP multipliers, LUT memory, filter length, and target throughput rather than by algorithmic tap count alone.

For multicarrier communication, FFT/IFFT processing often determines the computational core of the baseband. Harikrishna et al. (2010) presented a Verilog FFT implementation targeted to a Spartan-3 FPGA for fixed WiMAX and demonstrated the importance of reducing multipliers and memory. Canet et al. (2012) implemented an IEEE 802.11a/g OFDM receiver including frame detection, synchronization, FFT-based demodulation, equalization, and phase tracking. Their work is especially important methodologically because fixed-point word-length selection was performed together with hardware-cost and latency analysis, rather than as a later coding detail.

Reconfigurability and architectural portability are equally important in SDR systems. Liu, Cao, and Lai (2012) proposed an FPGA software architecture and platform-abstraction concept for software-defined radio, highlighting the need

to separate waveform functions from platform-specific connectivity. Font, Bartzoudis, Pascual-Iserte, and López (2012) reported a real-time FPGA MIMO-OFDM transceiver and demonstrated resource reuse in a closed-loop communication scheme. These works support modular interfaces, parameterized processing blocks, and reusable interconnects as design principles for communication FPGA architectures.

Numerical representation remains a cross-cutting constraint. Vera, Pattichis, and Lyke (2011) examined dynamic dual fixed-point

arithmetic and quantified the fundamental trade-off between precision, resource use, performance, and power in FPGA systems. In a communication datapath, unnecessary bit growth increases LUT, register, routing, memory, and multiplier cost, whereas excessive truncation degrades EVM, synchronization accuracy, and error rate. The literature therefore indicates that a successful real-time FPGA DSP design should combine streaming concurrency, pipeline timing closure, carefully selected fixed-point formats, algorithm-specific arithmetic such as CORDIC or distributed arithmetic, and parameterized reusable interfaces.

Table 1. Selected literature supporting the proposed methodology

Study	FPGA/DSP focus	Methodological implication
Andraka (1998)	CORDIC on FPGA	Shift-add rotations suit hardware-efficient synchronization and phase processing.
Meher et al. (2008)	DA/systolic FIR	Area-delay-power trade-off depends on LUT partitioning and systolic structure.
Xie et al. (2010)	Pipelined FIR	Pipeline registers improve maximum frequency but consume additional area.
Harikrishna et al. (2010)	FFT for OFDM	FFT architecture can be optimized for multiplier and memory usage.
Canet et al. (2012)	OFDM WLAN receiver	Fixed-point, latency, algorithm choice, and hardware cost must be co-designed.
Font et al. (2012)	MIMO-OFDM	Real-time FPGA transceivers benefit from module reuse and structured datapaths.
Wu & Shiue (2014)	CORDIC OFDM receiver	CORDIC can combine synchronization/equalization functions in real time.
Nandal et al. (2015)	256-tap DA FIR	Distributed arithmetic can significantly reduce logic in long parallel FIR filters.

Methodology

Step 1 - Define communication and timing requirements

The design begins with a throughput contract. The case study assumes complex baseband I/Q data arriving at 20 MS/s and an FPGA fabric clock of 125 MHz. The architecture must therefore accept one new complex sample at least every 6.25 clock cycles. To simplify back-pressure and leave margin for future bandwidth expansion, the proposed datapath is designed with an initiation interval (II) of one clock, meaning that every processing stage can accept a new sample on each cycle after pipeline fill. A valid-ready streaming interface is used between major blocks so that latency can vary without losing sample order. The clock-domain boundary to the ADC/DAC or external radio is isolated with FIFO buffering when the converter clock differs from the FPGA processing clock.

Step 2 - Select the streaming DSP decomposition

The signal chain is decomposed into functions that map naturally to FPGA resources: (i) input formatting and buffering; (ii) numerically controlled oscillator (NCO) and complex mixer for digital down conversion; (iii) cascaded-integrator-comb (CIC) decimation when large integer rate changes are needed; (iv) FIR compensation/channel filtering; (v) 64-point streaming FFT/IFFT for OFDM modulation or demodulation; (vi) CORDIC-based magnitude/phase, carrier correction, or equalization; (vii) QAM mapping/demapping; and (viii) framing/FEC interfaces. This organization follows the communication-oriented FPGA structures summarized by Meyer-Baese (2014), Harikrishna et al. (2010), Canet et al. (2012), and Wu and Shiue (2014).

Step 3 - Fixed-point word-length design

A floating-point algorithmic model is first used as the numerical reference. Each node is then assigned a signed fixed-point representation

based on observed dynamic range and acceptable quantization noise. The case study uses 16-bit signed I/Q samples, 18-bit FIR coefficients, widened accumulators for filtering, and stage scaling in the FFT. Rounding is preferred to truncation at major word-length reductions, and saturation is used instead of wraparound at external interfaces. The objective is to keep numerical error well below the EVM budget while avoiding unnecessary hardware growth. This procedure is consistent with the fixed-point co-design reported by Canet et al. (2012) and with the precision-resource trade-offs discussed by Vera et al. (2011).

Step 4 - Parallelism, pipelining, and resource strategy

All high-rate arithmetic is placed in registered pipelines. Complex multiplication is mapped to

dedicated FPGA DSP multipliers where available; coefficient-symmetric FIR filters may reduce multiplier count, whereas long filters can use distributed arithmetic when LUT/BRAM resources are more favorable (Meher et al., 2008; Xie et al., 2010; Nandal et al., 2015). The FFT is implemented as a streaming radix-2/radix-4 pipeline or vendor-equivalent core with explicit scaling. CORDIC engines are selected for phase rotation, magnitude, and arctangent because a shift-add structure maps efficiently to FPGA logic (Andraka, 1998). Functions whose activity does not overlap in time may share a CORDIC or correlator, following the resource-reuse concept demonstrated in the WLAN receiver of Canet et al. (2012).

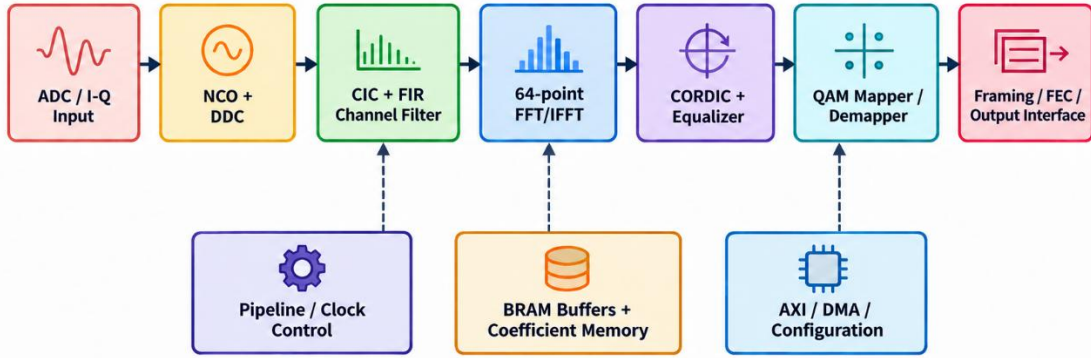


Figure 1. Proposed FPGA-based streaming DSP architecture for a real-time communication system.

Step 5 - Memory, control, and reconfiguration

Block RAM stores filter coefficients, FFT buffering, pilot values, and frame metadata. Coefficients and modulation parameters are memory-mapped so that channel bandwidth, decimation ratio, QAM order, and equalizer parameters can be changed without redesigning the complete bitstream. A lightweight AXI-style control plane is separated from the streaming sample plane. This separation supports the SDR portability objectives emphasized by Liu et al. (2012). For multi-antenna or closed-loop extensions, the same block interfaces permit replication of the front-end chain and sharing of higher-level control, consistent with the real-time MIMO-OFDM implementation strategy described by Font et al. (2012).

Step 6 - Functional and bit-true verification

Verification is performed at three levels. First, floating-point vectors verify algorithm correctness. Second, bit-true fixed-point vectors

use the exact quantization, rounding, saturation, and FFT stage-scaling rules intended for RTL. Third, the future RTL test bench replays the same vectors and checks sample-by-sample agreement after accounting for pipeline latency. For this paper, the bit-true model used 500 frames of normalized 64-QAM complex data for a 64-point scaled FFT and a 32-tap low-pass FIR test with 16-bit samples and 18-bit coefficients. The FFT error is evaluated by root-mean-square EVM relative to a floating-point FFT divided by 64; the FIR error is evaluated by SQNR and root-mean-square error (RMSE).

Step 7 - Synthesis and implementation evaluation plan

The final hardware phase should synthesize the RTL on a selected FPGA family and report LUTs, flip-flops, DSP slices, block RAM, maximum clock frequency, dynamic/static power, and post-route latency. Timing must be checked using the worst negative slack and the achieved maximum frequency, not only behavioral simulation.

Resource utilization should be collected for alternative FIR and FFT organizations to expose area-throughput trade-offs. Hardware-in-the-loop validation should then inject known OFDM frames through the FPGA interface and compare received symbols, EVM, and packet/bit error behavior with the software reference. No post-route utilization or power numbers are asserted in this methodology paper because such values would require a named target device and a completed synthesis/place-and-route run.

Results

The bit-true numerical evaluation indicates that the selected fixed-point formats provide substantial numerical headroom for a communication baseband. For the 64-point FFT, quantization at every radix-2 stage produced an RMS EVM of -71.69 dB relative to the scaled floating-point FFT, corresponding to an SQNR of 71.69 dB. For the 32-tap FIR test, 16-bit input samples and 18-bit coefficients produced an output SQNR of 91.60 dB with an RMSE of approximately 1.04×10^{-5} . These values are not equivalent to over-the-air system EVM because they isolate arithmetic quantization; however, they show that quantization error from the proposed datapath is far below the impairment levels normally introduced by channel noise, RF non-idealities, synchronization error, and finite-SNR estimation.

From the timing model, a fully streaming architecture at 125 MHz can accept 125 million complex sample events per second when the initiation interval is one clock. Relative to the 20 MS/s case-study requirement, the throughput margin is $6.25\times$. An illustrative pipeline allocation of 2 cycles for input formatting, 4 cycles for NCO/mixing, 12 cycles for channel filtering, 96 cycles for the streaming FFT, 16 cycles for CORDIC/equalization, and 6 cycles for demapping gives 136 cycles of end-to-end processing latency, or approximately $1.09 \mu\text{s}$ at 125 MHz. The pipeline latency does not reduce steady-state throughput because successive samples remain overlapped across stages.

These results are consistent with the literature in two ways. First, they confirm the design value of fixed-point co-optimization emphasized by Canet et al. (2012) and Vera et al. (2011): adequate precision can be retained without defaulting to resource-intensive floating point. Second, the one-sample-per-cycle goal reflects the FPGA-specific parallelism advocated in the filtering and FFT literature (Meher et al., 2008; Xie et al., 2010; Harikrishna et al., 2010). Actual resource occupancy will depend strongly on whether FIR multiplication uses DSP slices or distributed arithmetic and on whether CORDIC/FFT resources are replicated or time-shared.

Table 2. Bit-true numerical results and architecture-level timing results

Metric	Value	Interpretation
FFT size / data	64-point, normalized 64-QAM	Bit-true test case
FFT fixed-point EVM	-71.69 dB	Quantization-only error relative to floating-point reference
FFT SQNR	71.69 dB	Equivalent numerical signal-to-error ratio
FIR configuration	32 taps, 16-bit samples, 18-bit coefficients	Fixed-point low-pass filtering test
FIR output SQNR	91.60 dB	Very small coefficient/sample quantization error
FIR RMSE	1.04×10^{-5}	Relative to floating-point FIR output
Assumed FPGA clock	125 MHz	Architecture-level target, not post-route measurement
Required input rate	20 MS/s	Case-study real-time requirement
Streaming capacity	125 MS/s	One new sample accepted each clock
Throughput margin	$6.25\times$	125 MS/s divided by 20 MS/s
Pipeline latency	136 cycles $\approx 1.09 \mu\text{s}$	Illustrative stage allocation at 125 MHz

Conclusion and Discussion

This paper presented a step-by-step methodology for an FPGA-based DSP architecture intended for real-time communication systems. The central design decision is to treat throughput, numerical

precision, latency, and FPGA resource mapping as a single co-design problem. A modular streaming chain composed of digital down conversion, multirate/FIR filtering, FFT/IFFT processing, CORDIC-based correction or equalization, QAM processing, and framing provides a reusable structure for OFDM, WLAN/WiMAX-like links,

and SDR platforms. Literature up to 2015 shows that FPGA-specific techniques such as distributed arithmetic, CORDIC, pipelined FFTs, fixed-point analysis, and resource reuse can improve real-time efficiency compared with a direct software-style mapping. The bit-true evaluation supports the feasibility of the chosen precision. A 16-bit/18-bit fixed-point configuration produced low numerical error in both FFT and FIR tests, while the architecture-level timing model provides a $6.25\times$ sample-rate margin for a 20 MS/s requirement at a 125 MHz clock. The proposed 136-cycle pipeline corresponds to only about 1.09 μ s of processing latency while preserving one-sample-per-cycle steady-state throughput. These values should be interpreted as reference-model and timing-design results rather than device measurements. The next implementation step is device-specific RTL synthesis and hardware-in-the-loop testing. Future work should compare DSP-slice and distributed-arithmetic FIR variants, evaluate radix-2/radix-4 FFT alternatives, measure post-route timing and power, and quantify end-to-end EVM or BER under realistic channel models. Partial or parameter-driven reconfiguration can also be added to support multiple bandwidths and modulation modes without redesigning the entire signal chain. Such extensions preserve the main benefit of the proposed architecture: deterministic real-time performance with the reconfigurability needed by modern communication systems.

References

- Andraka, R. (1998). A survey of CORDIC algorithms for FPGA based computers. *Proceedings of the 1998 ACM/SIGDA Sixth International Symposium on Field Programmable Gate Arrays*, 191-200. DOI: 10.1145/275107.275139.
- Canet, M. J., Valls, J., Almenar, V., & Marín-Roig, J. (2012). FPGA implementation of an OFDM-based WLAN receiver. *Microprocessors and Microsystems*, 36(3), 232-244. DOI: 10.1016/j.micpro.2011.11.004.
- Font, O., Bartzoudis, N., Pascual-Iserte, A., & López, D. (2012). A real-time FPGA-based implementation of a high-performance MIMO-OFDM transceiver featuring a closed-loop communication scheme. *2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 100-107. DOI: 10.1109/WiMOB.2012.6379060.
- Harikrishna, K., Rama Rao, T., & Labay, V. A. (2010). FPGA implementation of FFT algorithm for OFDM based IEEE 802.16d (Fixed WiMAX) communications. *Journal of Electronic Science and Technology*, 8(3), 193-199. DOI: 10.3969/j.issn.1674-862X.2010.03.001.
- Liu, Q., Cao, K., & Lai, Y.-Y. (2012). FPGA software architecture for software defined radio. *Procedia Engineering*, 29, 2133-2139. DOI: 10.1016/j.proeng.2012.01.275.
- Meher, P. K., Chandrasekaran, S., & Amira, A. (2008). FPGA realization of FIR filters by efficient and flexible systolization using distributed arithmetic. *IEEE Transactions on Signal Processing*, 56(7), 3009-3017. DOI: 10.1109/TSP.2007.914926.
- Meyer-Baese, U. (2014). *Digital Signal Processing with Field Programmable Gate Arrays* (4th ed.). Springer. DOI: 10.1007/978-3-642-45309-0.
- Nandal, A., Vigneswaran, T., Rana, A. K., & Dhaka, A. (2015). An efficient 256-tap parallel FIR digital filter implementation using distributed arithmetic architecture. *Procedia Computer Science*, 54, 605-611. DOI: 10.1016/j.procs.2015.06.070.
- Vera, G. A., Pattichis, M., & Lyke, J. (2011). A dynamic dual fixed-point arithmetic architecture for FPGAs. *International Journal of Reconfigurable Computing*, 2011, 518602. DOI: 10.1155/2011/518602.
- Wu, C.-F., & Shiue, M.-T. (2014). CORDIC-based OFDM baseband receiver design. *2014 6th International Symposium on Communications, Control and Signal Processing (ISCCSP)*, 352-355. DOI: 10.1109/ISCCSP.2014.6877886.
- Xie, J., He, J., & Tan, G. (2010). FPGA realization of FIR filters for high-speed and medium-speed by using modified distributed arithmetic architectures. *Microelectronics Journal*, 41(6), 365-370. DOI: 10.1016/j.mejo.2010.04.006.