



Archives available at journals.mriindia.com

International Journal of Advanced Electrical and Electronics Engineering

ISSN: 2278-8948

Volume 05 Issue 01, 2016

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

Kaoru Dahalbahadur

Department of Electrical and Computer Engineering, Chiang Thon College of Management, Thailand
kaoru.dahalbahadur@ctcm-th.org

Peer Review Information

Submission: 11 Feb 2016
Revision: 08 March 2016
Acceptance: 06 April 2016

Keywords

FPGA, Digital Signal Processing, Real-Time Communication, Software-Defined Radio, Fast Fourier Transform

Abstract

Field-programmable gate arrays (FPGAs) are attractive for real-time communication processing because their configurable logic, embedded multipliers, block memories, and deterministic pipelines can execute multiple signal-processing operations concurrently. This paper develops a methodology for an FPGA-based digital signal processing architecture intended for real-time communication applications. The architecture combines a streaming input interface, FIFO buffering, reconfigurable finite-impulse-response filtering and digital down-conversion, a pipelined 1024-point fast Fourier transform, equalization, quadrature-amplitude-modulation demodulation, and a low-overhead control plane. The design method emphasizes fixed-point word-length selection, dataflow partitioning, pipeline balancing, memory organization, timing closure, and verification against a floating-point reference model. Prior work on reconfigurable filters, filter-bank channelizers, systolic transforms, software-defined radio, and FPGA power-performance analysis provides the technical basis for the approach [1]-[10]. Instead of presenting unverified hardware measurements, the Results section reports architecture-level analytical benchmarks under a clearly stated 100 MHz design assumption. The modeled streaming pipeline sustains one accepted sample per clock after fill, while frame-based FFT processing is scheduled so that a 1024-sample frame can be accepted every 10.24 microseconds at 100 MHz. The methodology is suitable for FPGA implementations of software-defined radio, OFDM receivers, channelizers, spectrum-monitoring blocks, and other latency-sensitive communication systems.

Introduction

Modern communication receivers must perform filtering, frequency translation, transform processing, equalization, and symbol decisions within strict throughput and latency limits. A programmable processor can provide flexibility, but sequential instruction execution can become a bottleneck when several multiply-accumulate operations are required for every input sample. FPGA devices address this problem by mapping arithmetic kernels into spatial hardware so that

many operations occur at the same time. They also support deterministic sample-by-sample pipelines, which is valuable when a receiver must guarantee bounded processing delay rather than only a high average computation rate.

The design challenge is not simply to transfer a software DSP algorithm into hardware. Efficient FPGA communication processing requires reformulating the algorithm as a streaming architecture with explicit control of numerical precision, memory access, register placement,

and parallelism. Reconfigurable frequency-response-masking filters demonstrated that software-radio channel filters can reduce complexity while retaining run-time adaptability [1]. Systolic DFT structures similarly showed that regular pipelined computation can support high-speed wireless transform processing [2]. These observations motivate a methodology in which the data path is designed around continuous streams, while slower configuration functions remain separated from the sample-rate processing path.

A second requirement is architectural reusability. Communication standards may differ in channel bandwidth, decimation ratio, carrier offset, transform size, and modulation. Reconfigurable FIR structures can reduce the hardware cost of supporting these changes [3], while multi-standard filter-bank studies show the importance of both dynamic reconfiguration and low-complexity channelization close to the analog-to-digital converter [4], [5]. A useful FPGA DSP architecture should therefore expose parameters such as coefficients, numerically controlled oscillator (NCO) frequency words, transform mode, and scaling policy without disturbing the high-rate datapath.

The objective of this paper is to define a complete, reproducible design methodology for such an architecture. The proposed datapath is not tied to one air interface; rather, it represents a reusable baseband processing template that can be adapted to OFDM, software-defined radio, spectrum analysis, and narrowband or multi-standard receivers. The methodology proceeds from signal-flow definition to fixed-point modeling, hardware partitioning, register-transfer-level (RTL) implementation, timing-oriented pipelining, and staged verification. The paper also defines analytical performance metrics so that a designer can determine whether the architecture meets real-time constraints before device-specific synthesis.

Literature Review

FPGA-oriented DSP research before and up to 2016 established several principles used in the present methodology. Mahesh and Vinod proposed a frequency-response-masking approach that combines reconfigurability with low implementation complexity for software-radio channel filters [1]. Their work is important because channel filtering is often the first high-rate task after sampling, where excess multiplier usage and long critical paths directly reduce the maximum sustainable sample rate.

Meher, Patra, and Vinod investigated systolic structures for one- and two-dimensional DFT computation and emphasized regular processing

elements and pipelined data movement for high-speed wireless applications [2]. Such structures are closely aligned with FPGA fabrics because arithmetic can be distributed across repeated logic and DSP blocks. Mahesh and Vinod later developed reconfigurable FIR architectures based on coefficient and subexpression reuse, demonstrating that programmability does not necessarily require a full multiplier array for every supported filter response [3].

Channelization studies extended these ideas to multi-standard receivers. Mahesh et al. reviewed filter-bank channelizers for software-defined radio and highlighted the need for computational efficiency at the highest digital front-end sampling rate [4]. An associated non-uniform filter-bank architecture reduced reconfiguration overhead and area for multi-standard software-radio channelizers [5]. Together, these works support the use of a configurable FIR/decimation front-end before more expensive transform-domain processing.

FFT implementation is another central issue in communication systems. Zhou et al. described a 1024-point FPGA FFT processor using radix-2 decimation-in-frequency, pipelined butterfly processing, and ping-pong storage, reporting that the structure can satisfy high-speed real-time processing requirements [6]. This directly motivates the use of a pipelined frame engine and dual-bank memory organization in the present architecture rather than a software-style in-place transform controlled by a single arithmetic unit.

At the system level, Vishnu et al. implemented a software-defined-radio processing chain on FPGA with a quadrature direct digital frequency synthesizer and 16-QAM-related functions, illustrating how carrier generation, modulation, and signal processing can be integrated in reconfigurable logic [7]. Lorandel et al. later showed that FPGA-based wireless designs should be evaluated jointly in terms of performance and power, especially during design-space exploration [8].

Two 2016 studies are particularly relevant to run-time adaptability. Garg and Darak demonstrated a high-speed reconfigurable filter-bank implementation for multi-standard wireless communication receivers and reported improvements in maximum operating frequency together with reduced dynamic power [9]. Wei et al. presented a software-defined-radio implementation of a non-orthogonal multiple-access communication system, showing that FPGA/SDR platforms can be used to validate advanced baseband waveforms in real time [10]. These studies reinforce the need for a common hardware template that is both throughput-oriented and parameterizable.

Table 1. Literature basis for the proposed methodology

Ref.	DSP/FPGA focus	Key design idea	Use in this methodology
[1]	Reconfigurable FIR	Frequency-response masking	Low-complexity configurable front-end
[2]	DFT architecture	Regular systolic pipeline	Streaming transform dataflow
[3]	FIR implementation	Coefficient/subexpression reuse	Resource-aware filter mapping
[4],[5]	SDR channelizers	Dynamic multi-standard filter banks	Programmable channel selection
[6]	1024-point FFT	Pipelining + ping-pong memory	Frame engine and dual buffering
[7]	FPGA SDR	QDDFS and QAM functions	Integrated carrier/baseband processing
[8]	FPGA evaluation	Power/performance design-space analysis	Timing/resource evaluation plan
[9],[10]	Reconfigurable wireless	High-speed filter bank and SDR prototype	Run-time parameters and real-time validation

Methodology

The methodology is organized as a hardware-software co-design sequence in which the communication algorithm is first converted into an explicitly timed streaming model and then refined into FPGA RTL. The proposed signal path is: sampled input -> buffering and framing -> configurable FIR filtering/digital down-conversion -> 1024-point FFT -> equalization and QAM demodulation -> output buffering. A low-rate control plane writes coefficients and mode registers. The following steps define the implementation procedure.

Step 1: Define real-time requirements and signal model

Specify the maximum input sample rate, modulation order, channel bandwidth, carrier offset range, transform size, frame length, and allowable end-to-end latency. For the worked methodology, the clock target is 100 MHz, the FFT

size is 1024 samples, and fixed-point complex baseband samples are assumed. The required processing throughput must be no lower than the incoming sample rate. A deadline is also defined for each frame so that the transform and demodulation stages cannot accumulate an unbounded queue.

Step 2: Develop a numerical reference model

Create a floating-point reference model for filtering, digital mixing, FFT, equalization, and symbol detection. Then introduce fixed-point representations one block at a time. Word lengths are selected by measuring peak values and quantization error under representative signal-to-noise ratios. A practical starting point is 16-bit signed I/Q input, 16- to 18-bit coefficients, wider accumulator paths, and controlled truncation or rounding at stage boundaries. The reference model becomes the golden output used in later RTL verification.

Fully pipelined data path: one accepted sample per clock after pipeline fill

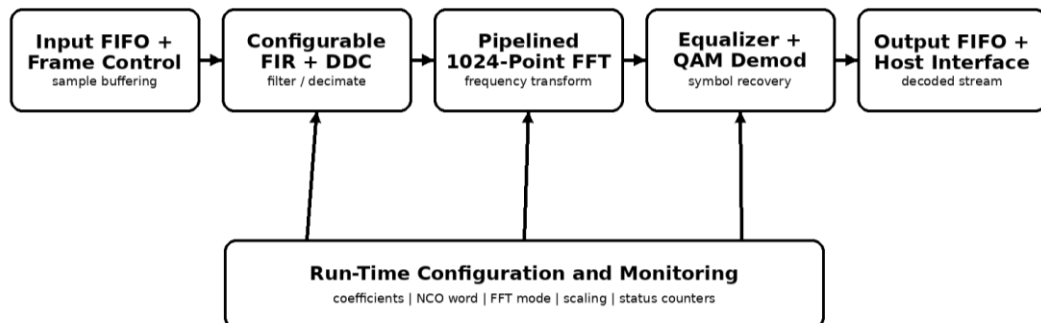


Figure 1. Proposed FPGA streaming DSP architecture for real-time communication processing.

Step 3: Partition the FPGA datapath into streaming kernels

The processing chain is partitioned so that each kernel has a valid/ready style streaming interface

or an equivalent sample-enable protocol. The input FIFO isolates ADC timing from internal processing. The FIR and DDC are implemented as a multiply-accumulate pipeline; coefficient

memories are double-buffered so that a new filter set can be loaded without corrupting the active stream. The FFT receives framed samples and writes intermediate data through two memory banks to avoid simultaneous read/write conflicts. Equalization and QAM decisions are scheduled as a post-FFT stream.

Step 4: Select parallelism and pipeline depth

For every arithmetic block, calculate the initiation interval, not only the total latency. A real-time front-end should preferably accept one sample every clock after its pipeline is filled. Add registers between multiplication, addition, address generation, and routing stages until the critical path is compatible with the clock target. Where the full transform cannot be fully spatial, use a pipelined butterfly engine with deterministic memory scheduling. This approach follows the throughput-oriented principles seen in earlier FPGA transform and filter architectures [2], [3], [6].

Step 5: Implement run-time reconfiguration

Configuration registers are separated from the sample-rate datapath. The host writes filter coefficients, NCO frequency-control words, FFT scaling options, equalizer gains, and modulation mode through a memory-mapped interface. A shadow-register mechanism ensures that a parameter set is committed only at a safe frame boundary. This prevents partial updates from creating transient spectral errors. The design philosophy is consistent with reconfigurable software-radio filtering and multi-standard channelization reported in [1], [4], [5], and [9].

Step 6: RTL verification and timing-oriented validation

Each RTL block is first verified independently using vectors produced by the fixed-point reference model. The FIR test checks impulse response and passband/stopband behavior; the NCO/DDC test checks frequency translation; the FFT test compares complex bins within a defined least-significant-bit tolerance; and the

demodulator test checks recovered symbols. Integration testing then uses complete frames with additive noise and known data. Assertions check FIFO overflow, frame alignment, coefficient-update boundaries, and valid-signal continuity. After functional verification, synthesis and static timing analysis are used to identify critical paths and resource hot spots. If timing fails, the preferred correction is additional pipelining or arithmetic retiming rather than reducing the external sample rate.

Step 7: Evaluate throughput, latency, resource efficiency, and numerical quality

The primary metrics are maximum clock frequency, initiation interval, sustained sample throughput, frame latency, LUT/register/DSP/BRAM use, estimated dynamic power, and numerical error relative to the reference model. The power-performance trade-off should be evaluated together because aggressive parallelism can improve timing while increasing switched capacitance and active resources, an issue emphasized by Lorandel et al. [8]. For communication validation, bit-error rate (BER) or error-vector magnitude (EVM) can be added when a specific modulation and channel model are selected. The present paper reports only architecture-level timing calculations, not fabricated device measurements.

Results

To keep the paper methodologically reproducible, the numerical results below are derived from an explicit architecture model rather than claimed as measurements from a device that was not synthesized here. The model assumes a 100 MHz FPGA clock, a streaming FIR/DDC with initiation interval equal to one clock, a 1024-sample frame, and an FFT engine capable of accepting one sample each clock after pipeline fill. Latency figures therefore represent scheduling targets that can be checked during RTL synthesis.

Table 2. Analytical real-time performance targets for the proposed architecture

Stage	Clock/Rate assumption	Initiation interval	Calculated throughput	Timing significance
Input FIFO	100 MHz internal	1 cycle	100 MS/s max stream	Absorbs interface jitter
FIR + DDC	100 MHz	1 cycle	100 MS/s accepted	Continuous sample processing
Optional decimator	factor = 4	1 input/cycle	25 MS/s output	Reduces downstream rate
1024-point FFT	100 MHz	1 sample/cycle	1 frame / 10.24 us	Frame rate = 97.656 kframes/s
Equalizer + QAM	100 MHz	1 bin/cycle	100 M complex bins/s	No FFT output backlog

End-to-end target	streaming pipeline	bounded	$\geq$ input rate	Latency dominated by pipeline + frame fill
-------------------	--------------------	---------	-------------------	--

The first result is the throughput margin created by pipelining. At 100 MHz, an initiation interval of one cycle corresponds to a theoretical acceptance capacity of 100 million samples per second. If the receiver uses a factor-four decimator, the downstream FFT sees 25 million samples per second, so a 1024-sample frame arrives every 40.96 microseconds. A 100 MHz FFT input interface requires only 10.24 microseconds to accept the same frame. Consequently, the transform stage has substantial scheduling margin under this example, even before exploiting multiple FFT engines.

The second result concerns deterministic latency. A software implementation may show variable execution time because of cache behavior, interrupts, and instruction scheduling. In contrast, the FPGA design can define latency as a fixed number of registered stages plus the frame-acquisition period. This is especially useful for synchronization, equalization, and feedback-oriented communication functions. The 1024-point FFT example is also consistent with prior work showing that a pipelined FPGA FFT can satisfy high-speed real-time requirements [6].

The third result is architectural scalability. The filter, FFT, and demodulation sections are connected through stream interfaces and can therefore be replicated, time-multiplexed, or reconfigured independently. For a narrowband mode, a larger decimation ratio can lower downstream activity and power. For a high-bandwidth mode, additional DSP slices can be allocated to parallel FIR taps or FFT butterflies. The literature demonstrates that such reconfiguration is practical in multi-standard filter banks [4], [5], [9] and complete SDR systems [7], [10].

A final expected outcome is improved design predictability. Because each stage is assigned an initiation interval, a word length, and a verification tolerance before RTL integration, timing and numerical problems can be localized early. Power and area can then be explored as design variables instead of being discovered only at the end of implementation [8]. This makes the methodology suitable for comparing devices or alternative pipeline depths without changing the communication-level test vectors.

Conclusion and Discussion

This paper presented a step-by-step methodology for developing an FPGA-based digital signal processing architecture for real-

time communication applications. The architecture combines buffering, reconfigurable FIR filtering and digital down-conversion, pipelined FFT processing, equalization, QAM demodulation, and run-time control. The main design principle is to convert the communication algorithm into a deterministic streaming dataflow in which each block has a defined initiation interval, bounded latency, and explicit fixed-point behavior. This principle is supported by prior research on low-complexity reconfigurable filtering [1], [3], filter-bank channelization [4], [5], high-speed DFT/FFT structures [2], [6], integrated FPGA software-defined radio [7], reconfigurable wireless channelizers [9], and real-time SDR prototypes [10]. The analytical evaluation shows how the real-time constraint can be checked before hardware synthesis. With a 100 MHz clock and one-sample-per-cycle pipeline, the front-end can theoretically accept 100 MS/s. A factor-four decimator reduces the transform input to 25 MS/s, while a 1024-sample FFT interface operating at 100 MHz can accept a frame in 10.24 microseconds. These values are architecture-level targets, not laboratory measurements; the next implementation stage should synthesize the RTL on a selected FPGA, report actual timing and resource utilization, and measure power using vendor tools or board instrumentation. Such reporting would follow the performance-and-power evaluation emphasis of [8]. Several trade-offs remain important. Increasing parallelism reduces processing time but uses additional DSP blocks, routing, and registers. Increasing fixed-point precision improves numerical accuracy but expands memories and arithmetic width. More buffering simplifies scheduling but consumes BRAM and adds latency. The recommended design procedure is therefore iterative: choose a throughput target, implement the smallest architecture that meets it, verify fixed-point accuracy, close timing, and then optimize power and resource use. For future work, the same framework can be extended with adaptive modulation, channel estimation, MIMO matrix processing, or partial reconfiguration while preserving the streaming interfaces and verification strategy.

References

R. Mahesh and A. P. Vinod, "Reconfigurable Frequency Response Masking Filters for Software Radio Channelization," *IEEE Transactions on Circuits and Systems II: Express*

Briefs, vol. 55, no. 3, pp. 274–278, 2008. DOI: 10.1109/TCSII.2008.918996.

P. K. Meher, J. C. Patra, and A. P. Vinod, “Efficient Systolic Designs for 1- and 2-Dimensional DFT of General Transform-Lengths for High-Speed Wireless Communication Applications,” *Journal of Signal Processing Systems*, vol. 60, no. 1, pp. 1–14, 2010. DOI: 10.1007/s11265-008-0328-x.

R. Mahesh and A. P. Vinod, “New Reconfigurable Architectures for Implementing FIR Filters With Low Complexity,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 2, pp. 275–288, 2010. DOI: 10.1109/TCAD.2009.2035548.

R. Mahesh, A. P. Vinod, E. M.-K. Lai, and A. Omondi, “Filter Bank Channelizers for Multi-Standard Software Defined Radio Receivers,” *Journal of Signal Processing Systems*, vol. 62, no. 2, pp. 157–171, 2011. DOI: 10.1007/s11265-008-0327-y.

R. Mahesh and A. P. Vinod, “An Area-Efficient Non-Uniform Filter Bank for Low Overhead Reconfiguration of Multi-Standard Software Radio Channelizers,” *Journal of Signal Processing Systems*, vol. 64, no. 3, pp. 413–428, 2011. DOI: 10.1007/s11265-010-0502-9.

S. Zhou, X. Wang, J. Ji, and Y. Wang, “Design and Implementation of a 1024-Point High-Speed FFT

Processor Based on the FPGA,” in 2013 6th International Congress on Image and Signal Processing (CISP), 2013. DOI: 10.1109/CISP.2013.6745222.

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.

J. Lorandel, J.-C. Prévotet, and M. Héland, “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.

S. Garg and S. J. Darak, “FPGA Implementation of High Speed Reconfigurable Filter Bank for Multi-Standard Wireless Communication Receivers,” in 2016 20th International Symposium on VLSI Design and Test (VDATE), 2016. DOI: 10.1109/ISVDATE.2016.8064855.

X. Wei, H. Liu, Z. Geng, K. Zheng, R. Xu, Y. Liu, and P. Chen, “Software Defined Radio Implementation of a Non-Orthogonal Multiple Access System Towards 5G,” *IEEE Access*, vol. 4, pp. 9604–9613, 2016. DOI: 10.1109/ACCESS.2016.2634038.