IAES Journal of Electrical and Electronics Engineering
Vol. 4, No. 2, December 2017, pp. 36~40
ISSN: 2338-7432, DOI: 10.11591/jeee.v4i2.pp36-40

High-performance signal processing with FPGA: design strategies, optimization, and real-time applications

Aiman Zakwan Jidin, Auzani Jidin
Faculty of Engineering Technology, Universiti Teknikal Malaysia Melaka, Malacca, Malaysia

Article Info

Article history:

Received Oct 29, 2016
Revised Aug 31, 2017
Accepted Oct 31, 2017

Keywords:

Digital filters
Digital signal processing
Field-programmable gate array
Hardware acceleration
Parallel architecture
Real-time processing

ABSTRACT

The increasing demand for high-speed digital signal processing (DSP) has encouraged the development of hardware platforms capable of delivering high throughput, low latency, and predictable execution. Conventional digital signal processors provide programmability, but their sequential processing structures may become restrictive in computationally intensive applications. Field-programmable gate arrays (FPGAs) offer an alternative through configurable logic, dedicated arithmetic resources, and spatial parallelism. This review examines FPGA-based high-performance signal processing research, focusing on architectural design, computational optimization, programming methodologies, and real-time applications. It discusses how pipelining, parallel processing, fixed-point arithmetic, resource sharing, and specialized DSP blocks contribute to implementation efficiency. Representative applications include digital filtering, Fourier transforms, radar processing, image analysis, and heterogeneous signal-processing platforms. The review also considers the practical trade-offs among throughput, latency, hardware utilization, numerical accuracy, and power consumption. Existing studies indicate that FPGA performance depends not simply on the availability of logic resources but on how algorithms, data movement, and hardware structures are coordinated. The review concludes that systematic design-space exploration and hardware-aware optimization are essential for developing efficient and dependable real-time FPGA signal-processing systems.

This is an open access article under the CC BY-SA license.

1. INTRODUCTION

Digital signal processing (DSP) is fundamental to modern electronic systems that acquire, analyze, transform, and interpret information from physical environments. Applications such as wireless communication, radar, medical instrumentation, image processing, and industrial monitoring require increasingly high processing rates and reliable response times. Conventional digital signal processors provide a convenient programming environment, but their performance may become limited when applications require extensive parallel computation or deterministic low-latency operation. These requirements have encouraged the use of field-programmable gate arrays (FPGAs) as flexible platforms for implementing specialized signal-processing hardware.

FPGAs differ from conventional processors because their internal logic can be configured to implement application-specific datapaths. Multiple arithmetic operations can be executed concurrently, while pipelining allows successive data samples to move through processing stages without waiting for an entire computation to finish. Modern FPGA devices also include dedicated DSP blocks, embedded memory, and high-speed interfaces that support computationally intensive workloads. These architectural characteristics make FPGAs attractive for applications where throughput and predictable execution are more important than the flexibility of a purely software-based implementation.

However, achieving high performance on an FPGA is not simply a matter of translating an algorithm into hardware. Designers must balance parallelism, pipeline depth, numerical precision, memory

bandwidth, and available logic resources. A highly parallel implementation may provide excellent throughput but consume excessive hardware resources, whereas a compact architecture may fail to meet real-time processing requirements. This review therefore examines three connected research problems: how FPGA architectures support high-performance DSP, which design and optimization strategies improve implementation efficiency, and how these techniques are applied in real-time signal-processing systems.

2. REVIEW SCOPE AND APPROACH

This article presents a narrative review of representative FPGA-based DSP publications available no later than 2016. The selected literature covers hardware architectures, digital filtering, linear signal transforms, DSP-block mapping, high-level synthesis, and real-time applications. Priority is given to peer-reviewed publications with traceable bibliographic information and identifiable DOI records. The purpose is to synthesize design principles and implementation challenges rather than to establish a numerical ranking of FPGA devices or commercial platforms.

3. FPGA ARCHITECTURES FOR HIGH-PERFORMANCE DSP

3.1. Configurable logic and dedicated DSP resources

The architecture of an FPGA provides the foundation for implementing parallel signal-processing operations. Configurable logic blocks and lookup tables can implement control functions, arithmetic operations, and data-routing structures, while registers support sequential processing and pipeline stages. Dedicated DSP blocks provide optimized arithmetic resources, particularly for multiplication and multiply-accumulate operations that frequently occur in filtering and transform algorithms. The combination of general configurable logic and specialized arithmetic resources enables designers to select hardware structures according to the computational characteristics of an application.

The effective use of these resources requires careful mapping of algorithmic operations onto the FPGA fabric. Ronak and Fahmy [1] demonstrated that synthesis tools do not always map arithmetic graphs onto FPGA DSP blocks in a way that achieves maximum throughput. Their work proposed a mapping approach that considers the internal pipeline structure of DSP blocks when scheduling arithmetic operations. This result highlights the importance of architecture-aware mapping rather than relying exclusively on conventional synthesis optimization.

3.2. Parallelism and pipelining

Parallel processing is one of the principal advantages of FPGA-based DSP. Independent operations can be assigned to separate hardware units, allowing several data samples or computational stages to be processed concurrently. Pipelining divides a complex operation into sequential stages, enabling a new input to enter the processing structure before earlier inputs have completed all stages. These techniques can substantially increase throughput, although their effectiveness depends on data dependencies, resource availability, and the timing characteristics of the circuit.

Wang and McAllister [2] introduced a streaming accelerator architecture for FPGA signal and image processing. Their approach uses fine-grained streaming elements to construct accelerator networks for applications including fast Fourier transforms, motion estimation, matrix multiplication, and edge detection. The study demonstrates that carefully designed streaming architectures can provide real-time performance while reducing the limitations associated with software-programmable soft processors. It also illustrates how execution models and memory addressing mechanisms influence the efficiency of FPGA accelerators.

3.3. Algorithm-to-hardware transformation

A further challenge is selecting an appropriate hardware structure for a given signal-processing algorithm. Linear transforms such as the discrete Fourier transform can be implemented through different algorithmic decompositions and datapath organizations. The best implementation depends on the required throughput, latency, hardware budget, and data organization. Milder et al. [3] addressed this problem through the SPIRAL hardware-generation framework, which automatically explores transform algorithms and produces synthesizable hardware descriptions.

Automated generation is particularly valuable when an algorithm has many possible hardware realizations. Rather than manually implementing and testing every architecture, designers can specify performance and structural requirements and allow a design framework to explore alternatives. This approach can reduce development effort and provide implementations that compete with manually designed circuits. Nevertheless, the generated architecture must still be evaluated against the timing, resource, and verification requirements of the target application.

4. DESIGN STRATEGIES AND OPTIMIZATION TECHNIQUES

4.1. Digital filter architecture

Digital filters are among the most established applications of FPGA-based signal processing. Finite impulse response (FIR) filters are attractive because they can provide stable filtering characteristics and are suitable for pipelined implementations. Their conventional direct-form architectures may require substantial multiplier resources when the filter order increases. Therefore, efficient FPGA filter design often involves coefficient symmetry, arithmetic restructuring, resource sharing, and alternative filter representations.

Vandenbussche *et al.* [4] investigated multiplicative FIR filter structures and their implementation on modern FPGA devices. Their study examined different filter responses and showed how fixed-point arithmetic can be used to construct practical filter architectures. It also discussed the relationship between hardware utilization, clock speed, and filter characteristics. This work demonstrates that filter selection and arithmetic representation should be treated as architectural decisions rather than purely mathematical design choices.

4.2. Fixed-point arithmetic and numerical precision

Numerical representation has a direct influence on the hardware cost and performance of an FPGA implementation. Floating-point arithmetic can provide a wide dynamic range and simplify algorithm development, but it generally requires more hardware resources than carefully selected fixed-point arithmetic. Fixed-point representations can reduce multiplier complexity, memory usage, and power consumption, making them attractive for embedded DSP applications. However, the word length must be selected carefully because quantization errors can affect signal quality and algorithmic accuracy.

A reliable fixed-point design process should begin with the numerical requirements of the target application. Designers need to examine input ranges, coefficient precision, intermediate accumulation growth, and the acceptable output error. Reducing word length without evaluating its effect on the complete processing chain can lead to unacceptable numerical degradation. Consequently, simulation and hardware verification should be performed together to establish an appropriate balance between precision and implementation cost.

4.3. Resource mapping and high-level synthesis

Efficient FPGA design also depends on how arithmetic operations are assigned to available hardware resources. DSP blocks, lookup tables, registers, and embedded memories have different performance and capacity characteristics. A design that uses too many dedicated arithmetic units may exhaust the available DSP resources, while a design that relies heavily on general logic may experience longer critical paths. Ronak and Fahmy's [1] DSP-block mapping strategy illustrates how improved resource assignment can increase operating frequency without requiring a complete redesign of the signal-processing algorithm.

High-level synthesis (HLS) provides another approach to improving design productivity. It allows designers to describe algorithmic behavior using higher-level programming languages, after which synthesis tools generate hardware structures. Nane et al. [5] evaluated FPGA HLS tools and showed that different tools can produce different results in performance and resource utilization, even when given similar source descriptions. This indicates that high-level abstraction does not eliminate the need for hardware-aware optimization and careful evaluation of generated circuits.

4.4. Summary of design strategies

The preceding studies show that high-performance FPGA signal processing requires coordinated decisions across algorithm, architecture, and implementation. Table 1 summarizes the principal strategies and their expected benefits, together with the limitations that designers must consider. The table is intended as a qualitative comparison rather than a universal ranking because the effectiveness of each technique depends on the application and target FPGA. Its comparison provides a practical framework for selecting optimization methods during the design process.

As indicated in Table 1, no single optimization method addresses every performance limitation. Pipelining and parallelism primarily improve throughput, while fixed-point arithmetic and resource mapping influence hardware cost and numerical efficiency. Streaming architectures address the organization of continuous data processing, whereas automated generation and HLS focus on development productivity. A practical FPGA design often combines several of these techniques to meet its application-specific requirements.

Table 1. FPGA design strategies for high-performance signal processing

Design strategy	Main purpose	Potential benefit	Main consideration
Pipelining	Divide computation into stages	Higher throughput	Pipeline latency and register overhead
Parallel processing	Execute independent operations concurrently	Increased processing capacity	Logic and DSP-block utilization
Fixed-point arithmetic	Reduce numerical hardware complexity	Lower resource and power requirements	Quantization error
DSP-block mapping	Assign arithmetic operations to dedicated resources	Improved arithmetic throughput	Limited number of DSP blocks
Streaming architecture	Process continuous data flows	Reduced processing delays	Data synchronization and buffering
Automated hardware generation	Explore algorithm and datapath alternatives	Reduced design effort	Quality of generated implementation
High-level synthesis	Raise programming abstraction	Improved design productivity	Dependence on compiler optimization

5. REAL-TIME APPLICATIONS OF FPGA-BASED DSP

5.1. Radar signal processing

Radar systems require rapid processing of received signals to identify targets, estimate motion, and suppress unwanted clutter. These operations may involve filtering, spectral analysis, and repeated arithmetic computations that must be completed within strict timing constraints. Yousuf et al. [6] presented an FPGA-based moving target indication (MTI) processing module for radar applications. Their implementation used two-pulse and three-pulse cancellers and demonstrated the potential for combining MATLAB-based algorithm development with hardware optimization on an FPGA platform.

The radar example illustrates why predictable execution is important in real-time DSP. A processing delay may reduce the usefulness of information when targets are moving or when the system must respond immediately to changing conditions. FPGA parallelism allows several processing operations to be implemented as dedicated hardware structures, reducing dependence on sequential instruction execution. However, the overall system must also account for data acquisition, buffering, and communication with other processing components.

5.2. Image and video processing

Image and video processing systems often require continuous processing of large data streams. Operations such as edge detection, motion estimation, filtering, and feature extraction can be organized into pipelined stages that operate on successive pixels or image blocks. Wang and McAllister [2] demonstrated streaming accelerator structures for image-processing operations, including motion estimation and Sobel edge detection. Their results highlight the suitability of FPGA streaming architectures for applications that demand continuous data processing and real-time response.

5.3. Digital filtering and communication systems

Digital filters are widely used in communication receivers, audio processing, instrumentation, and sensor systems. Their implementation must often satisfy requirements for frequency response, sampling rate, latency, and numerical accuracy. FPGA-based FIR and multiplicative FIR structures provide flexibility in selecting filter architectures and arithmetic representations [4]. This flexibility is valuable in systems that need to support different signal characteristics or adjustable processing configurations.

5.4. Application-oriented design considerations

The requirements of real-time DSP systems differ considerably across application domains. Radar processing emphasizes predictable latency and reliable detection, while image processing often prioritizes throughput and continuous data handling. Communication systems may require a combination of high sampling rates, precise filtering, and efficient resource utilization. These differences indicate that FPGA architecture should be selected according to the dominant system constraint rather than based on peak clock frequency alone.

6. RESEARCH CHALLENGES AND FUTURE DIRECTIONS

One major challenge is balancing throughput against hardware utilization. Increasing parallelism can improve processing capacity, but it also consumes additional logic, registers, and DSP blocks. Deep pipelining may increase maximum clock frequency while adding latency and storage overhead. Future design methodologies should therefore incorporate application-level performance targets and systematic design-space exploration instead of optimizing isolated hardware metrics.

Another challenge concerns memory access and data movement. Arithmetic units can remain underutilized when data cannot be delivered at the required rate, particularly in applications involving large

images or continuous sensor streams. Efficient buffering, data reuse, and streaming interfaces are therefore essential for achieving sustained throughput. Hardware architectures should be evaluated not only by their computational capacity but also by their ability to maintain continuous data flow.

Design productivity and verification also remain important research concerns. Manual RTL design offers detailed control but can require substantial development and debugging effort. HLS and automated hardware-generation frameworks can reduce this burden, although their output quality depends on compiler behavior and the structure of the source description [3], [5]. Future tools should provide more reliable performance estimation, improved resource mapping, and stronger support for hardware verification.

Finally, power efficiency must be considered alongside processing speed. High-performance architectures can increase switching activity and hardware utilization, potentially raising total energy consumption. Fixed-point arithmetic, efficient memory organization, and selective resource sharing can help reduce unnecessary hardware activity. The most effective real-time FPGA system is therefore not necessarily the one with the highest theoretical throughput, but the one that satisfies its timing and accuracy requirements within an acceptable resource and energy budget.

7. CONCLUSION

FPGAs provide a flexible and effective platform for high-performance DSP because they support configurable datapaths, parallel execution, and dedicated arithmetic resources. Research published through 2016 demonstrates their usefulness in digital filtering, Fourier transforms, radar processing, and image analysis. Pipelining, parallelism, fixed-point arithmetic, DSP-block mapping, streaming architectures, and automated hardware generation are important strategies for improving implementation performance. Their effectiveness, however, depends on the relationship between algorithmic requirements, hardware resources, data movement, and real-time constraints. The review indicates that successful FPGA-based DSP design requires a system-level optimization approach. Throughput, latency, numerical precision, resource utilization, and power consumption must be evaluated together rather than treated as independent objectives. HLS and automated design frameworks can improve productivity, but they do not remove the need for hardware-aware analysis and verification. Continued progress in these areas can support more efficient and dependable FPGA-based signal-processing systems for communication, radar, imaging, and other real-time applications.

REFERENCES

[1] B. Ronak and S. A. Fahmy, "Mapping for maximum performance on FPGA DSP blocks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 4, pp. 573–585, Apr. 2016, doi: 10.1109/TCAD.2015.2474363.

[2] P. Wang and J. McAllister, "Streaming Elements for FPGA Signal and Image Processing Accelerators," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 6, pp. 2262–2274, Jun. 2016, doi: 10.1109/TVLSI.2015.2504871.

[3] P. Milder, F. Franchetti, J. C. Hoe, and M. Püschel, "Computer generation of hardware for linear digital signal processing transforms," *ACM Transactions on Design Automation of Electronic Systems*, vol. 17, no. 2, pp. 1–33, Apr. 2012, doi: 10.1145/2159542.2159547.

[4] J. J. Vandenbussche, P. Lee, and J. Peuteman, "Multiplicative finite impulse response filters: Implementations and applications using field programmable gate arrays," *IET Signal Processing*, vol. 9, no. 5, pp. 449–456, Jul. 2015, doi: 10.1049/iet-spr.2014.0143.

[5] R. Nane *et al.*, "A Survey and Evaluation of FPGA High-Level Synthesis Tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 10, pp. 1591–1604, Oct. 2016, doi: 10.1109/TCAD.2015.2513673.

[6] M. Yousuf, A. Aziz, and R. Mahmud, "Area Efficient Implementation of MTI Processing Module on a Reconfigurable Platform," *Chinese Journal of Engineering*, vol. 2014, pp. 1–7, Jan. 2014, doi: 10.1155/2014/167184.