

This document describes the advantages of using floating-point processing in FPGAs for digital signal processing (DSP) in radar applications.

Introduction

Modern radar systems process high-frequency signals at over 100 GHz. Modern array radar systems have various modes enabled by digital signal processing, including modes for searching, identification, tracking, targeting, and surveillance. The majority of these radar systems, whether steered mechanically or electronically, now process signals digitally to improve system flexibility with multiple modes using software-driven waveforms.

Many military applications have physical limitations of board space and power consumption. FPGAs offer the best performance and size, weight, and power (*SWaP*) characteristics for these DSP-driven applications. In particular, Altera® FPGAs provide the following signal processing advantages:

- Efficient floating-point DSP for superior system range and observability
- Parallel DSP processing with expanded memory capacity and I/O bandwidth
- Variable-precision DSP enabling efficient integration near the antenna
- A complete DSP solution unprecedented in the industry
- Low static and dynamic power

In addition to the above list, Altera floating-point FPGAs enable higher-precision processing nearer to the antenna, which improves system dynamic range and reduces losses. Floating-point processing also scales numbers by tracking and moving the binary decimal point in the mantissa with minimal risk of overflow. ⁽¹⁾

Radar Digital Processing Requirements

Modern radars calculate an enormous amount of information in *real-time*. That means they process the information without delays. Real-time processing demands stringent requirements of the signal processing device. These systems also have substantial space, power, and heat requirements. The following technologies support real-time DSP processing:

- FPGAs
- Standalone DSPs
- General-purpose Graphical Processing Units (GPUs)
- Multi-core processors

While all of these devices offer flexible, software-defined digital processing, only Altera FPGAs offer superior SWaP, true floating-point, and parallel signal processing.

A critical element of signal processing is the measure of performance versus power, often measured in GFLOPs per Watt for floating-point operation. Table 1 shows the range of GFLOPs per Watt for various products:

Table 1. Digital Signal Processing Efficiency

Product Type	GFLOPs per Watt
High-end CPU	< 3
General-purpose GPU	< 5
High-end DSP	< 8
Stratix® IV FPGA	5 – 7
Stratix V FPGA	12 - 15

Table 1 shows that Altera FPGAs can perform floating-point operations more than 10 times as efficiently as many CPUs. Altera’s 28-nm Stratix V device family is expected to double the GFLOP performance compared to previous generation FPGAs.

Designers can extend this efficiency by combining the capabilities of Stratix IV or Stratix V FPGAs with Bittware Anemone DSPs. The combination of Stratix series FPGAs with Anemone DSP is expected to achieve even higher performance, and can use standard ANSI C software code for the digital signal processor with acceleration in the FPGA. Floating-point FPGAs provide the most space and power efficient solution for a given performance requirement.

Real-time DSP solutions must perform thousands of calculations in parallel. Only FPGAs can provide this capability because they provide superior customized memory access (a topic known as data locality) and extremely wide internal bandwidth, unlike hard-coded processors. Modern FPGAs contain 1000s of DSP elements operating in parallel, over 50 Mb of memory on chip, and over 500 Gbps of I/O bandwidth. The DSP elements also include highly efficient features such as pre-adder elements that reduce the elements needed for processing digital filters.

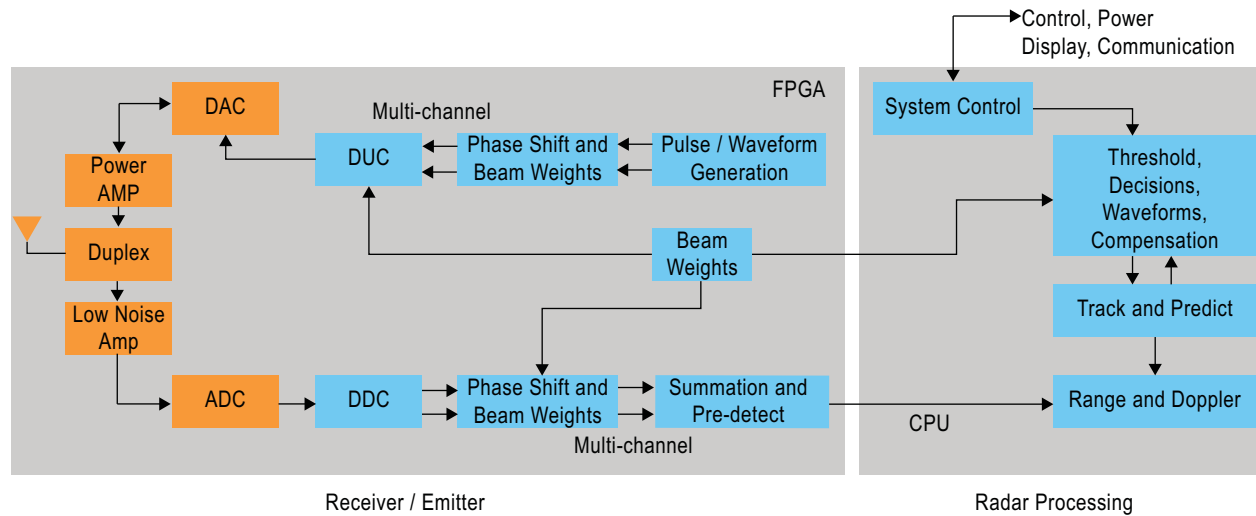
Radar systems must operate with significant dynamic range. The power of the transmitted signal decays with the square of the distance on the way to the target and again on the return, resulting in decay with the 4th power of the distance. Discriminating distant and low-observable targets, without being “blinded” by high intensity, results in exorbitant data widths if implemented in fixed point. Typically single or even double precision floating-point processing is used in at least part of the overall radar processing chain to solve this problem. A very limited number of DSP and multi-core processors have the ability to support these elements of floating-point processing. Only Altera offers true single- and double-precision floating-point methodology in an FPGA.

Radar Digital Processing Architecture

Modern radars have an analog interface to the antenna or antenna elements, but the analog signals are converted to digital signals for processing. The receiver typically includes downconversion and beamforming elements, as shown in Figure 1.

The emitter includes pulse generation, beamforming, and digital up conversion. The radar processing element processes information to enhance the signal, remove environmental effects, detect target location and velocity, and perform other tasks such as system control. The following sections describe these elements in detail.

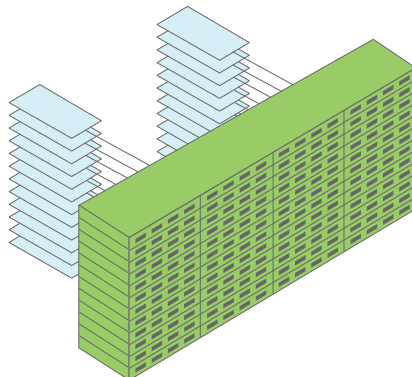
Figure 1. Typical Digital Radar Architecture



Beamform Filtering

In radar systems, beamforming is the forming of beams in a physical direction when radiating or receiving energy, also known as “steering.” Beamforming is a spatial filtering method of gaining higher sensitivity in a direction. On receiving, the beamforming element locates the arrival angle of the signal. This is particularly useful with phased array antennas that have multiple elements, or large antenna arrays, as shown in Figure 2.

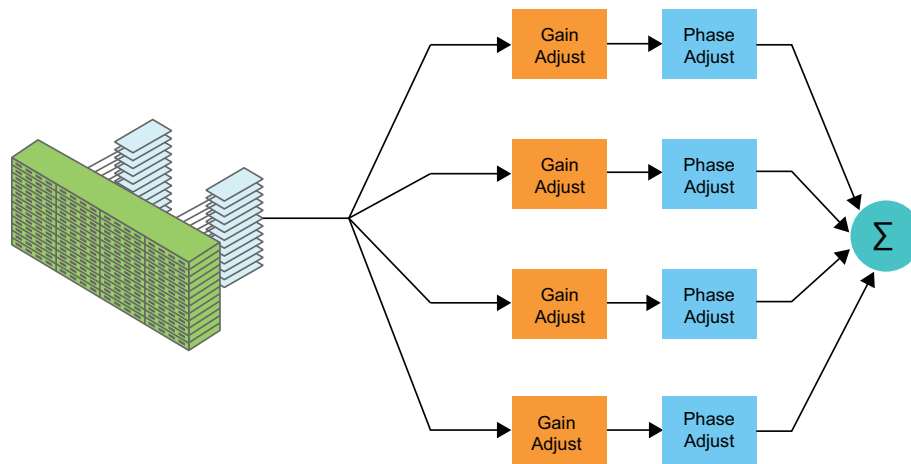
Figure 2. Array Antenna



When radars receive energy at an antenna or array element (that is, antennas), interference energy from unintended targets and the environment can have a detrimental effect on the capabilities of the radar system. To resolve these effects, radar designers create a system that does spatial and temporal filtering. Spatial filtering focuses on the distance and direction in space, while temporal filtering focuses on the time element (or frequency) of the signal.

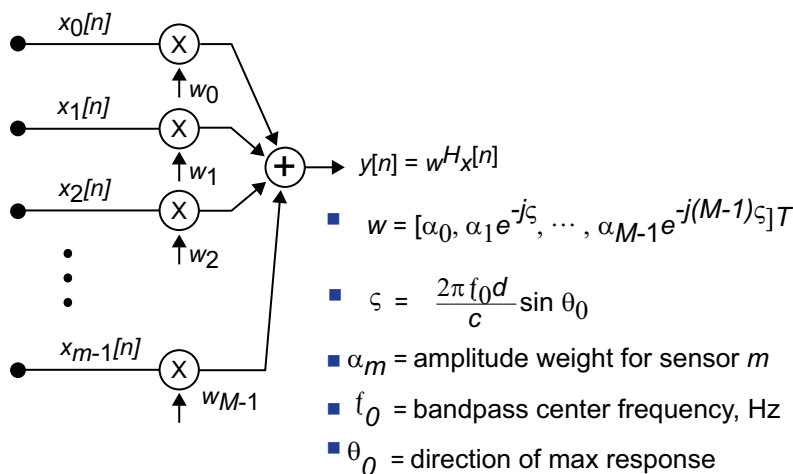
Beamforming looks at multiple antennas in a multi-channel environment and aligns signal delays along one direction by adjusting the phase and gain, such that signals from the different antennas constructively add in the region of interest, and cancel out in unwanted directions, as shown in Figure 3.

Figure 3. Beamforming Radar Array Model



Weights are also applied to control the beam shape, as represented in Figure 4 and Figure 5

Figure 4. Beamforming Map for Narrowband Phased Array (2)



Mathematically, narrowband beamforming corresponds with a FIR filter method. Traditionally, beamforming is a fixed-point processing effort. The use of floating-point FPGAs can simplify the task of converting MATLAB system-level models from floating point to fixed point. Figure 4 and Figure 5 show beamforming for narrowband and broadband applications, respectively.

Figure 5. Beamforming Map for Broadband Delay-Sum Array (2)

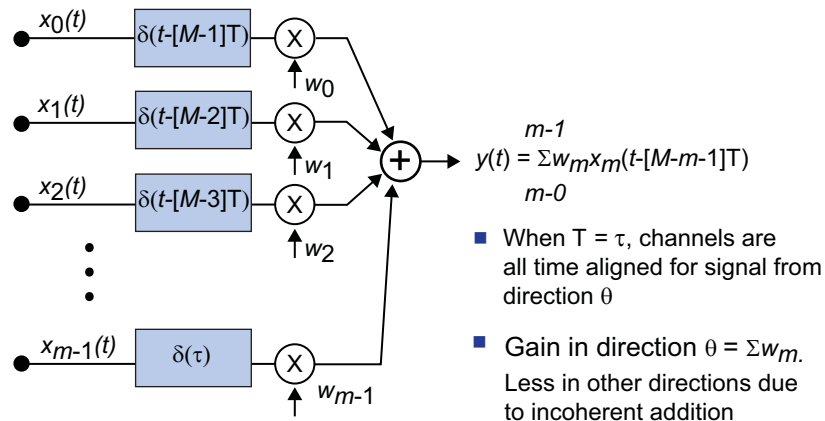
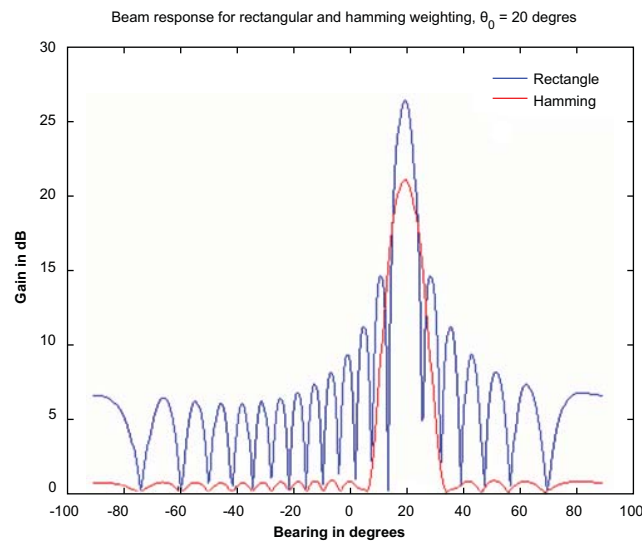


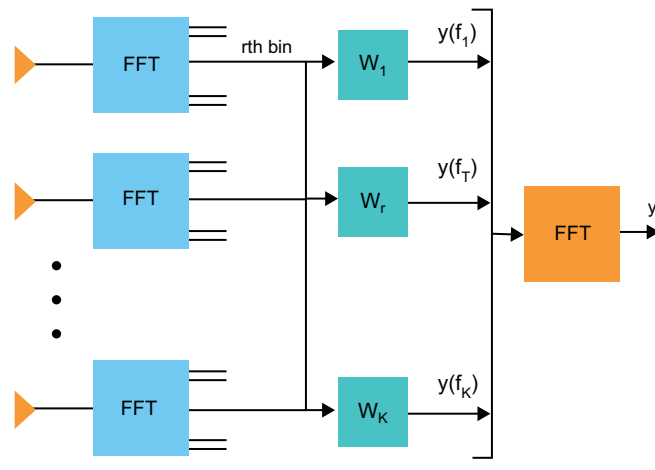
Figure 6 shows an FIR filtered beam response that controls sidelobe levels. In this example, the weights are applied to the beam to determine a 20 degree peak angle.

Figure 6. Filtered Beam Response (2)



The time domain analysis works well with a single beam filter, but many arrays filter multiple beams at once. For multiple beams in broadband applications, Fourier transform analysis is shown in Figure 7.

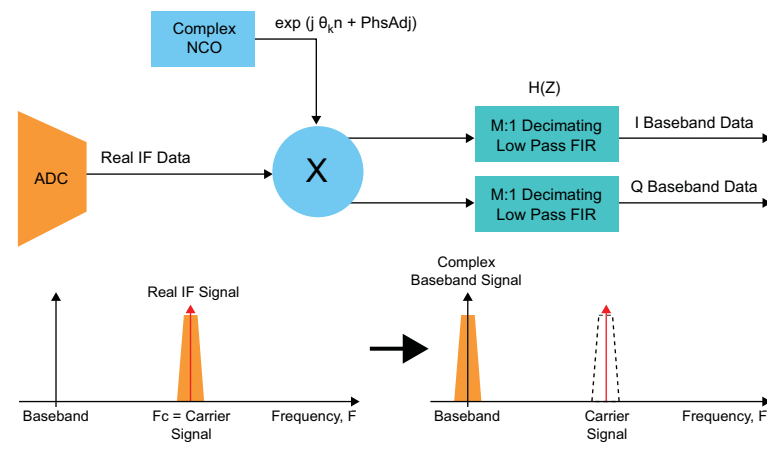
Figure 7. Fourier Transform Analysis in Broadband Beamformer



Beamforming in Altera FPGAs

Figure 8 shows a traditional downconversion system, where an incoming signal is shifted to a lower frequency. The incoming signal is mixed with another locally generated frequency creating a heterodyne output that is filtered and down sampled. A heterodyne is the generation of new frequencies by mixing (multiplying) two oscillating waveforms. The baseband data is then ready for signal and data processing at later stages in the system.

Figure 8. Traditional Downconversion (3)



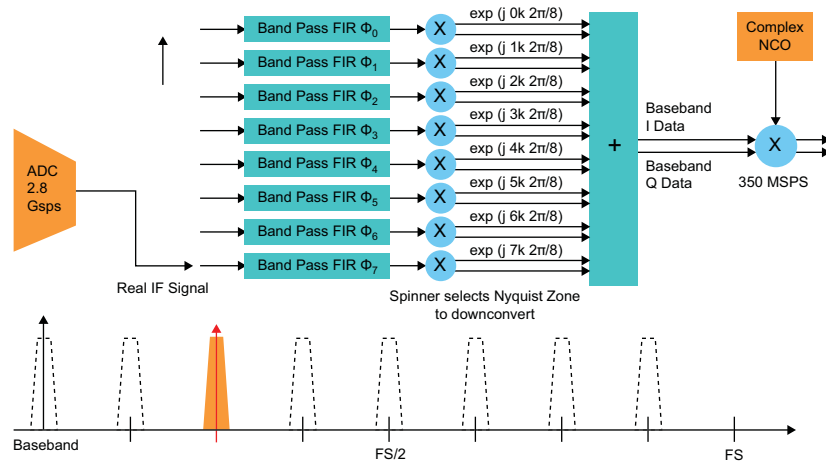
To work well, digital beamformers should have the following characteristics:

- High-bandwidth input connections that can be targeted at a number of interface types
- A simple architecture that is well suited for real-time filtering in parallel
- Customizable and sufficient memory to locate beam weight data near the processor element in time and/or space
- Customizable time-domain or frequency-domain processing at a data rate sufficient to process all signals
- High-performance in a platform that uses low-power and size
- Wide-bandwidth for continued on-chip processing or high-bandwidth output to migrate data for further processing in the system

The following section demonstrates how the Altera beamformer performs against these needs.

Altera provides a radar front end beamforming design example that helps designers get started with beamform processing. This design example uses the aliased polyphase digital downconversion (DDC) shown in [Figure 9](#), due to its efficient use of resources⁽³⁾. This polyphase decomposition is computationally efficient and allows analysis of multiple phases in a signal stream. This method uses aliasing to reduce resources while having the same effect as shifting to lower frequencies before sampling. Finally, the mixer operates at the output sample rate rather than the input rate, again saving resources and power

Figure 9. Aliased Polyphase Filtering⁽³⁾



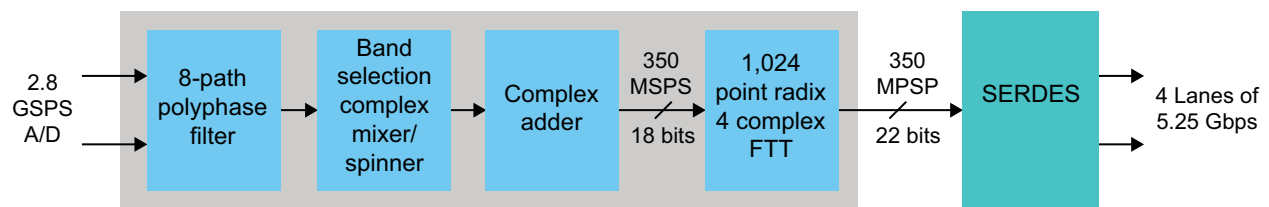
The signal flow of the radar front end design example is shown in [Figure 9](#) and [Figure 10](#). In this example the 2.8-GSPS ADC input is converted to 8 phases at 350MHz by the ALTLVDS megafunction. The design first performs 8 to 1 downsampling with the polyphase filters. Then, the spinner and adder allow selection of the desired Nyquist zone. The complex baseband signal is then ready for further processing. FFT analysis is used for conversion to spectral representation. The high speed SERDES can

be used to channel the large amount of data to another FPGA or a backplane. The polyphase filters, multi-phase NCO, band selection, complex adder, and 1K complex FFT blocks are available in the DSP Builder advanced blockset. This example design can also for the basis of a more complex design on the same chip, perhaps followed by pulse-compression, Doppler formation, or space-time adaptive processing (STAP).

Altera Stratix and Arria® series devices provide the following superior support for high-performance beamformer applications:

- High-bandwidth input connections, including LVDS and high-speed SERDES
- A simple architecture optimized for real-time parallel filtering, including pre-adders for efficient symmetric filtering, the only true floating-point capable DSP solution, and a 64-bit accumulation path for higher precision processing
- High memory and DSP density, including the Stratix V GS device, with up to 55-Mbit of on-chip memory, and over 4000 DSP elements on a single FPGA die
- For applications that require off chip data analysis, these devices offer a number of high-speed memory interface types, including DDR3, QDRII+ and RLDRAM II
- User-configurable logic to perform filtering in the time or frequency domain
- The highest performance to power ratio of any digital processing device available
- High-bandwidth SERDES, LVDS, and general purpose I/O to move data on and off-chip without delays

Figure 10. Radar Front End Beamformer Design Example



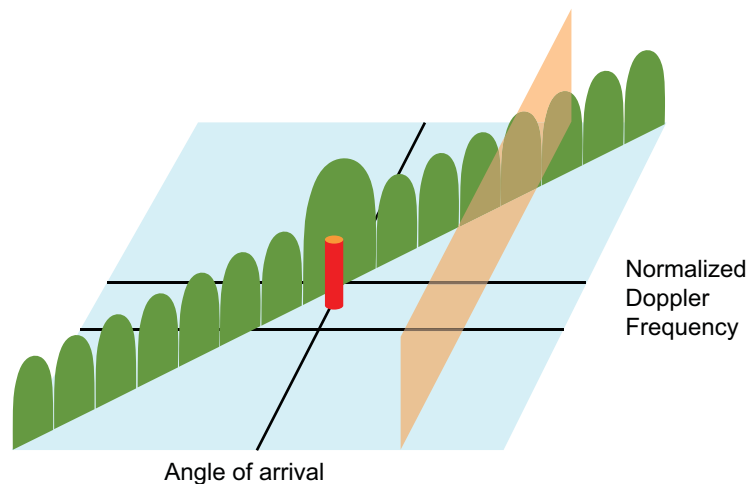
Space-Time Adaptive Processing

Radar systems use increasingly complex and high processing rate techniques, such as Space-Time Adaptive Processing (STAP). STAP is an advanced signal processing technique that is used in radar applications to suppress interference by working in both the spatial and time domains. This method improves the detection of slow moving targets that are obscured by clutter or jamming, making it particularly suitable for airborne surveillance, where the search for slow moving targets in severe clutter is a common scenario. The challenge with STAP is that it is difficult to process with low latency. With STAP algorithms, FPGAs reduce system size, weight, and power while reducing calculation latency.

Key components in the STAP algorithm are QR decomposition, as well as forward and backward substitution. QR decomposition is a floating-point matrix inversion operation. Both QR decomposition and substitution are highly iterative and sensitive to numerical effects. With the wide dynamic range requirements in radar, and rounding noise introduced in fixed point processing by other SRAM FPGAs and multi-core offerings, the use of floating-point processing is preferred.

An effective radar must discriminate targets against noise. Figure 11 illustrates three different types of noise. Receiver noise acts as a noise floor, and is indicated in light blue in the diagram. This noise level is determined by the quality of the receiver chain, including the antenna, analog processing, and digital downconversion (DDC). A second source of noise is the clutter, shown in green. Ground clutter is based on reflections off the ground from stationary or slow moving elements. Their Doppler component is thus largely defined by the platform speed. The third source of noise is from jammers. Jammers typically transmit across all frequencies. However, since any one jammer has only one specific location, only one particular angle is affected, as shown in tan color.

Figure 11. Radar with Noise from Clutter and Jamming



STAP processing filters and suppresses clutter and jammers, so that targets can be more easily identified.

There are a number of possible algorithms to perform STAP processing. ⁽⁴⁾ In selecting a STAP processing algorithm, designers must decide whether to work in the power domain or voltage domain. Both methods involve deriving a noise estimate from surrounding radar cells, and applying the inverse to the cell of interest. Calculating the inverse of the noise estimate requires matrix inversion and back substitution. Highly iterative computations like these are only possible using floating-point processing. In addition, the high number of mathematical operations required can exceed the data processing capabilities of many DSPs. These limitations lead to practical constraints in the design of modern radar applications, limiting the noise suppression performance and sensitivity of a radar system. A parallel processing floating-point FPGA can achieve superior noise suppression and sensitivity than comparable systems.

When applied correctly, STAP is a very challenging algorithm to perform. The benefit of using STAP is an order-of-magnitude sensitivity improvement in target detection. To accomplish this, a developer needs very high processing requirements, low latency, fast adaptation, and very high dynamic range. The following section describes how Altera meets or exceeds these requirements.

STAP Processing in Altera FPGAs

Altera has developed a STAP radar floating-point design example that illustrates how to implement this algorithm in Stratix series FPGAs. This design example demonstrates how high performance floating-point and vector processing are implemented. Altera provides the following support for efficient STAP floating-point processing

- Floating-point operations supported by the underlying silicon structure
- A library of efficient floating-point elements available to designers
- A design entry tool that allows efficient mapping of algorithms to the silicon structures

The Altera STAP design example demonstrates Altera's floating-point support. This design example is comprised of a realistic set of parameters, including 16 antennas, 16 Doppler bins, 64 target steering vectors, and a pulse repetition frequency of 1 kHz. This translates to a processing speed of 80 GFLOP/s. The entire design example can be implemented on the EP4SGX230 medium density Stratix IV FPGA.

The design example was created using MATLAB and Altera's DSP Builder advanced blockset. This is a standard Altera design flow that includes the following steps:

1. The entire STAP processing chain is implemented in MATLAB, including stimuli generation and plotting facilities for the results.
2. The data processing chain is implemented using the DSP Builder advanced Blockset.
3. MATLAB/ DSP Builder co-simulation verifies correct operation.

Figure 12 shows a plot generated using the example design. The upper plot shows the signals collected by the uniform linear array (ULA), before STAP is applied. The blue line indicates the location of the target. This plot shows that the target would not be recognized, as the presence of a jammer completely overcomes the system.

Figure 12. Results of STAP Processing on Range

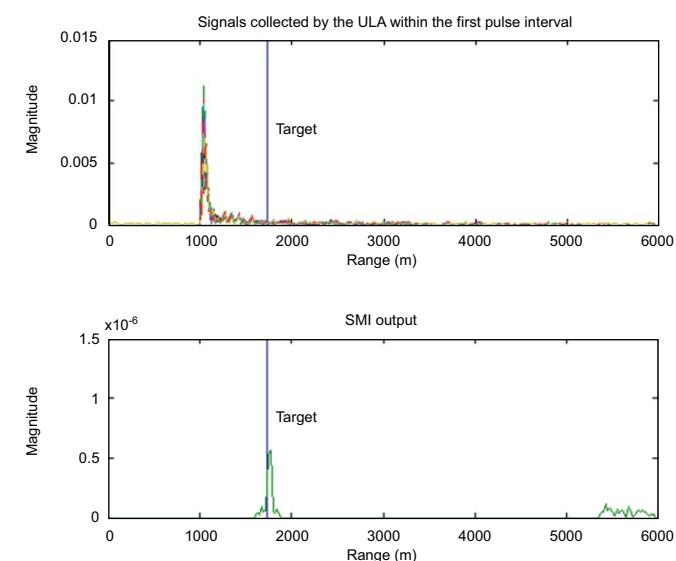


Figure 13 shows the data snapshot, which is dominated by a jammer at 60 degrees. The second subplot displays the weights that are calculated in STAP. The diagram shows that a weight of -80dB is applied at 60 degrees, suppressing the jammer. The yellow line along the clutter ridge indicates a weight of around -30dB to -40dB suppresses clutter. The jammer suppression in Figure 13 is located exactly where the jammer is indicated in Figure 11. The clutter, which was shown as a green diagonal in Figure 11, corresponds to the yellow clutter suppression line in subplot 2 of Figure 13, also a straight line along the diagonal with appropriate scaling.

Figure 13. Results of STAP Processing on Doppler

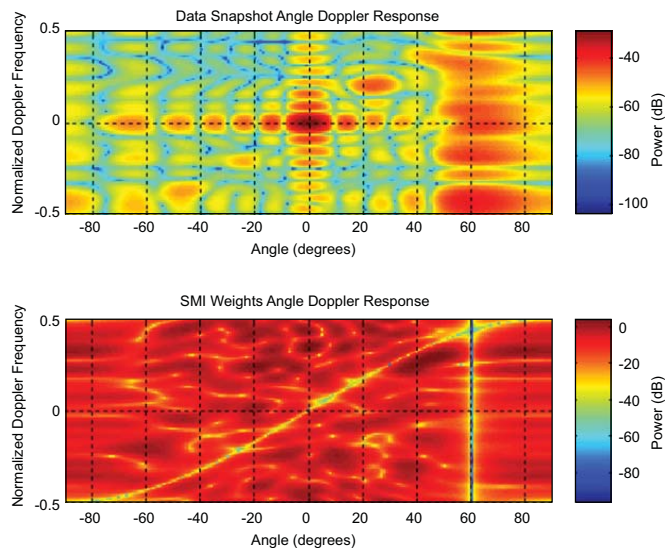


Figure 13 shows an order-of-magnitude sensitivity improvement in target detection with an Altera Stratix IV FPGA driven by the high dynamic range enabled by true floating-point processing. Radar systems built with Stratix series devices have low latency and can quickly adapt to environmental changes, with increased embedded memory and DSP element density for parallel processing.

In summary, Altera's STAP design example is a good example of how to move from a complex algorithm to a real hardware implementation. The example uses the Simulink design entry method to unlock the full potential of the underlying silicon structure. This method gives radar system designers access to hundreds of GFLOPs on a single FPGA, which raises radar system performance to new levels.

Other Processing Algorithms

There are a number of other algorithms that are of interest to the radar developer. Constant False Alarm Rate (CFAR) processing is often the first detection decision made in processing. ⁽⁵⁾ This algorithm uses an adaptive measurement of the noise in the neighboring cells and adaptively adjusts the detection threshold. CFAR maintains the probability of a false alarm at a constant level, even in the presence of noise. Designers can use floating point in conjunction with CFAR algorithms to detect targets surrounded by background clutter, such as a submarine periscope surrounded by a rough sea.

Alternatively, pulse compression is another method that reduces transmitter power while maintaining the desired range resolution. Designers can use floating-point FFTs to improve the filtering capability of the system.

Doppler filtering uses the Doppler Effect to compare the frequency shift of the return pulse with the outgoing pulse. FFT filters sort the target velocity vector toward the radar into bins. Again, floating point helps with the sensitivity of the calculation.

Table 2 and Table 3 show benchmarks for Altera's floating-point FFT IP core. This IP core implements a true floating-point format that scales each individual number without scaling blocks of numbers or causing rounding errors. Table 2 shows the resource and performance results for a single 1024-point floating-point FFT core in a Stratix IV 4SGX70 device using the Quartus II software version 10.1. Table 3 provides the resource and performance results of fourteen 1024-point FFT IP cores in a Stratix IV 4SGX530 device. Results show that even a dense, large, floating-point design clocks at over 300 MHz. These results are easily replicated using the FFT IP core, or the benchmark design is available from Altera upon request.

In summary, Altera Stratix IV devices can process floating-point operations at a similar frequency to competitive FPGA fixed-point processing.

Table 2. Resource and Performance Results: One FFT IP Core in Stratix IV 4SGX70 Device

Logic Elements	23,722	58,080	41%
M9K Blocks	89	462	19%
DSP blocks	64	384	17%
f_{MAX} (MHz)	315		

Table 3. Resource and Performance Results: 14 FFT IP Cores in Stratix IV 4SGX70 device

Logic Elements	301,308	424,960	71%
M9K Blocks	1280	1280	100%
DSP blocks	896	1024	88%
f_{MAX} (MHz)	302		

Altera's Floating-Point FPGAs

The following items are required to implement floating point in a system:

- A silicon structure that supports full floating-point processing
- Floating-point capable tools
- A complete library of efficient floating-point functions

Until recently, most silicon, tools, and IP are not integrated and designers are required to piece all of the elements together. Altera FPGAs are superior to the market leading FPGA because they process true floating-point calculations instead of block truncated floating-point calculations. Additionally, Altera has designed a tool flow and IP library which complements the superior silicon architecture. Altera FPGAs are also better than microprocessors and digital signal processors for floating point since they leverage the natural parallelism of FPGAs.

Altera FPGAs, unlike microprocessors, have thousands of high precision, hardened multiplier circuits that can be used for mantissa multiplication, and also used as high speed barrel shifters. Data shifting is required to perform the normalization to set the mantissa decimal point, and denormalization of mantissas as needed to align exponents. Use of a simple barrel shifter structure to perform this task requires very high fan-in multiplexers for each bit location, as well as the routing to connect each of the possible bit inputs. Altera devices are optimized to solve the high fan-in and routing problems that lead to device resource constraints, slow clock rates, and excessive logic usage in competitive FPGAs.

Altera FPGAs can use larger mantissas than an IEEE 754 representation. This is possible because the variable-precision DSP blocks support 27x27 and 36x36 multiplier sizes, which can be used for 23-bit single-precision floating-point datapaths. Using configurable logic, floating-point mantissa precision can be extended as necessary while still maintaining IEEE754 compliant interfaces. Using a mantissa size of a few extra bits, such as 27 bits instead of 23 bits, allows for extra precision from one operation to the next, and allows for more efficient hardware implementations. For example, a fully parallel vector dot product operation requires a bank of floating-point multipliers followed by an adder tree of floating-point adders. By carrying extra mantissa precision, the logic intensive denormalization and normalization functions associated with floating-point adders are eliminated except for the entrance and exit stage of the adder tree.

28-nm Variable-Precision Architecture

The DSP blocks in 28-nm Stratix V and Arria V FPGAs are specifically designed to meet the requirements of next generation radar and electronic warfare systems. Altera's new variable-precision DSP architecture allows designers to specify the required precision for each part of the design. This results in more efficient utilization of logic and DSP resources, and lower power consumption, while providing higher-precision DSP where it is needed.

In 18-bit precision mode, variable-precision architecture incorporates dual 18x18 multipliers, with optional hard pre-adders. The pre-adders are useful in applications like symmetric filtering, as they can add samples to be multiplied with the same coefficients. In 18 x 18 mode, variable precision supports dual integrated coefficient register banks, and the ability to efficiently implement either direct form or systolic form FIR filters. Efficient complex multiplication, essential for FFT implementation, is also supported.

An asymmetrically sized multiplier can be useful for the complex multiplications used in FFT processing because it provides for fixed-precision coefficients for the complex twiddle factors, while allowing data growth that occurs during processing. In FFTs, data growth occurs at a rate of 1 bit per each radix2 stage.

The Stratix V Variable Precision DSP block has been designed for FFT processing. Two DSP blocks can perform an 18x18 complex multiplier, three DSP blocks can perform an 18x25 complex multiplier, and four DSP blocks can perform an 18x36 complex multiplier. This allows the DSP resources to increase in proportion to the bit precision growth on the data side of the multiplier, and use fixed precision 18-bit twiddle

factors. The result is a highly efficient use of DSP resources which allows designers to trade precision for usage of DSP block resources and associated power consumption at each radix stage of the FFT. Using these modes, the Variable Precision DSP architecture is well suited to perform parallel frequency-domain processing of data from large antenna arrays.

In addition, the Variable Precision DSP block is the first to incorporate internal coefficient storage banks, in either 18-bit or 27-bit modes. This reduces usage of external memory blocks and the required routing of coefficients. It also improves timing closure at high clock rates.

The variable-precision DSP also supports native 27x27 multipliers with 64-bit accumulators, the largest in industry. This provides for higher precision and higher dynamic range signal processing, reducing fixed point numerical processing effects. Hard pre-adders, integrated coefficient register banks, and direct form or systolic form FIR filters are also supported in 27-bit mode.

Larger multiplier sizes are also supported by combining 18x18 and 27x27 multipliers in the variable-precision DSP blocks. This technique allows high performance implementation of 36x36 and 54x54 multiplier sizes. The 27x27, 36x36, and 54x54 multiplier sizes allow efficient implementation of single-precision, single-extended, and double-precision floating point.

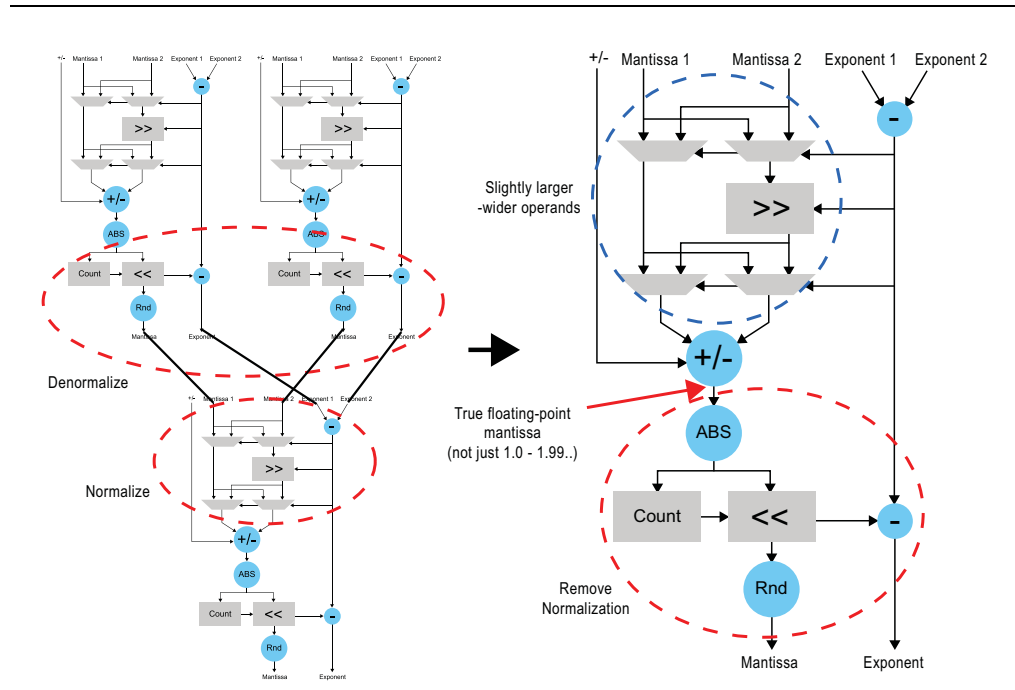
Fused Datapath Tool Flow

Altera's high-performance, low-latency, floating-point tool flow is known as "fused datapath" technology, as described in [Figure 14](#). This tool flow allows the designer to build mixed fixed- and floating-point FPGA vector signal processing datapaths. This tool analyzes normalization requirements, and inserts these stages only where necessary. This technique leads to a dramatic reduction in logic, routing, and multiplier-based shifting resources. It also results in much higher f_{MAX} , or achievable clock rates, even in the very large floating-point designs.

Because an IEEE 754 representation is required to comply with floating-point standards, all of the floating-point functions support this interface at the boundaries of each function, whether a fast Fourier transform (FFT), a matrix inversion, sine function, or a custom datapath.

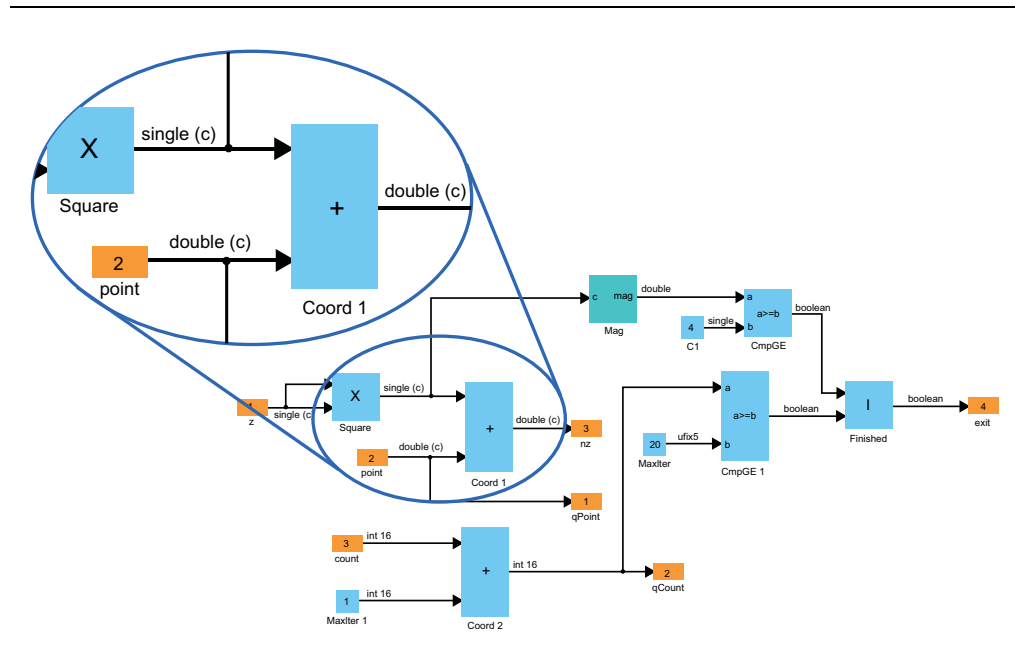
A fused-datapath tool flow is likely to produce results different from the IEEE 754 microprocessors approach. The main reason for these differences is that floating-point operations are not associative. Summing the same set of numbers in the opposite order results in various least significant bits (LSBs). To verify the fused-datapath method, the fused-datapath tools allow the designer to declare a tolerance, and to compare the hardware results output from the fused-datapath tool flow to the simulation model results. Altera analyzed the numerical precision of fused-datapath methodology and determined that it is statistically more accurate than IEEE754.

Figure 14. Fused Datapath Optimizations



The fused-datapath tool flow is integrated in Altera's DSP Builder advanced blockset, supported by MathWorks' MATLAB and Simulink. This method allows easy simulation as well as FPGA implementation of fixed and floating-point designs. Figure 15 illustrates how floating-point complex types—both single- and double-precision architecture—are used in conjunction with fixed-point types.

Figure 15. Floating-Point Design Entry Example



DSP Builder offers a single environment for building mixed floating-point and fixed-point designs. The tool also supports abstraction of complex numbers and vectors, making design description clean and easy to change. Complexity associated with mantissas, exponents, normalizations, and special conditions are abstracted away, similar to a floating-point software flow.

Floating-Point Function Library

Math.h functions are simple functions expected in a simple C library—trigonometric, log, exponent, and inverse square root, as well as basic operators such as divide. These functions are supported in the fused-datapath flow, as a floating-point library available for designers.

One of the most common functions requiring high dynamic range is matrix inversion. To accommodate this, the fused-datapath library includes linear algebra support, including the following reference designs:

- Matrix multiply
- Cholesky decomposition (used in matrix inversion algorithms)
- LU decomposition (used in matrix inversion algorithms)
- QR decomposition (used in matrix inversion algorithms)

The DSP Builder tool flow supports complex and vector representation. In addition, fixed- and floating-point operations can be easily mixed within the same design. This is essential for efficiently implementing many of the linear algebra operators in many algorithms used in the next generation radar systems. This also allows rapid design reuse and re-parameterization of vector and matrix sizes. Finally, the comprehensive library support of the fused-datapath tool flow allows customers to build large, complex, and highly optimized floating-point datapaths.

Summary

Altera's floating-point FPGAs provide a superior solution for DSP in radar applications. FPGAs offer better size, weight and power characteristics as a function of performance. This method can help reduce system latency while improving dynamic range and reducing losses. By combining highly optimized silicon features and patented library functions with a floating-point DSP methodology, radar systems developed with Altera FPGAs can achieve the new level of performance required by modern military applications.

Further Information

- *Achieving One TeraFLOPs with 28-nm FPGAs*
<http://www.altera.com/literature/wp/wp-01142-teraflops.pdf>
- *Implementing FIR Filters and FFTs with 28-nm Variable-Precision DSP Architecture*
<http://www.altera.com/literature/wp/wp-01140-fir-fft-dsp.pdf>

Acknowledgements

- Ian Land, Senior Manager, Military Business Unit, Altera Corporation
- Michael Parker, Senior Manager, Product Marketing, Altera Corporation
- Volker Mauer, Senior Manager, SSG Engineering, Altera Corporation

References

1. Bores Signal Processing, *Introduction to DSP—DSP Processors: Data Formats*, December, 2010. http://www.bores.com/courses/intro/chips/6_data.htm
2. Jeffs, Brian D. *Beamforming: A Brief Introduction*, Presentation, (Brigham Young University, October, 2004).
3. Harris, Fredric J. *Multirate Signal Processing for Communication Systems*, Chapter 6, (Prentice Hall, ISBN 0-13-146511-2).
4. Richards, Mark A. *Fundamentals of Radar Signal Processing*, Chapter 9, (McGraw-Hill, ISBN 0-07-144474-2).
5. Worsham, Richard. *Northrop Grumman Radar Notes*, et al, Presented at Radar 2010 Conference, May, 2010

