



OVERVIEW

This document describes RFEL's 'HyperSpeed' range of high-performance FFT cores. The cores are scalable in terms of transform length and processing parallelism and can be built to optimally process complex data streams with sample rates from 400MS/s to several GS/s. The cores use a parallel pipelined processing architecture, which allows continuous real time data to be processed with no gaps between frames.

HyperSpeed FFT cores are intended for use in applications where processing speed is critical and optimum use of available silicon is required. The cores are available for licence in netlist macro form as a component ready to be combined with customer's own IP or as part of an integrated design from RFEL.

FEATURES

- Lengths to 128K-points
 - Processing parallelism adjustable at factory
 - Continuous real time processing of complex data to several GS/s
 - Optimally targeted at Xilinx Virtex4 and Virtex5 FPGA families
 - Prime factor lengths available
 - Precision and scaling adjustable at factory
 - Fully bit-true models available
 - Memory resource tradeoffs (logic vs Block RAM) adjustable at factory
 - Multiplier resource tradeoffs (logic vs DSP48) adjustable at factory
 - Real-input processing available as option
 - Input and output data order options
 - Runtime programmable FFT and IFFT function
 - Windowing available as option, including Polyphase
 - Overlapping available
-

APPLICATIONS

- Wide-band filter banks
- Communications systems
- Electronic warfare (radar, sonar, surveillance)
- Medical instrumentation
- Test instrumentation
- Real-time spectral analysis

GENERAL DESCRIPTION

A simplified view of the HyperSpeed FFT architecture is shown in Figure 1.

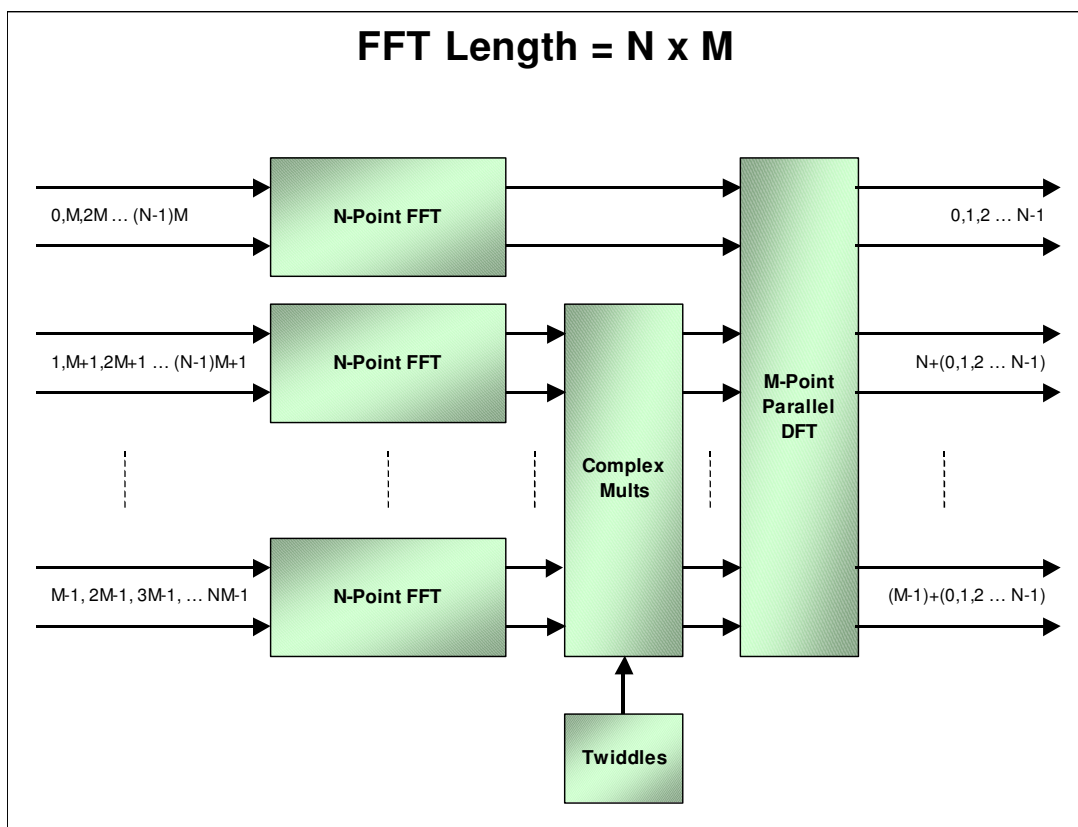


Figure 1: HyperSpeed FFT Architecture

The FFT pipeline is built from 'M' individual FFTs, each with a transform length of 'N'-points. Common resources are shared between the individual FFTs (such as twiddle factor generation, control etc). Frequency-domain outputs from the N-point FFTs are 'twiddled' and combined using an M-point DFT, which is implemented as a fully parallel pipelined FFT.

The HyperSpeed architecture can support transform lengths that are integer powers of two using existing 'off-the-shelf' building blocks. Other transform lengths can be built using RFEL's Matrix FFT prime-length building blocks.

BUILD PARAMETERS

The following standard parameters are required by RFEL in order to build a specific variant of the HyperSpeed FFT. Various examples together with estimated resource requirements are provided in the case studies section at the back of this document. Quotes for other less standard features such as windowing, overlapping, real-input transform etc are available on request

FFT Length (K)

The HyperSpeed FFT can be built to any integer power of two length, providing there are enough FPGA resources available in the target device. RFEL can implement FFTs that have factors other



sync_in	[Timing Diagram]																[Timing Diagram]															
enable_in	[Timing Diagram]																[Timing Diagram]															
I_in(1)	I ₀	I ₁₆	I ₄	I ₂₀	I ₈	I ₂₄	I ₁₂	I ₂₈	I ₃₂	I ₄₈	I ₃₆	I ₅₂	I ₄₀	I ₅₆	I ₄₄	I ₆₀	I ₆₄	I ₈₀	I ₆₈	I ₈₄	I ₇₂	I ₈₈	I ₇₆	I ₉₂								
I_in(2)	I ₁	I ₁₇	I ₅	I ₂₁	I ₉	I ₂₅	I ₁₃	I ₂₉	I ₃₃	I ₄₉	I ₃₇	I ₅₃	I ₄₁	I ₅₇	I ₄₅	I ₆₁	I ₆₅	I ₈₁	I ₆₉	I ₈₅	I ₇₃	I ₈₉	I ₇₇	I ₉₃								
I_in(3)	I ₂	I ₁₈	I ₆	I ₂₂	I ₁₀	I ₂₆	I ₁₄	I ₃₀	I ₃₄	I ₅₀	I ₃₈	I ₅₄	I ₄₂	I ₅₈	I ₄₆	I ₆₂	I ₆₆	I ₈₂	I ₇₀	I ₈₆	I ₇₄	I ₉₀	I ₇₈	I ₉₄								
I_in(4)	I ₃	I ₁₉	I ₇	I ₂₃	I ₁₁	I ₂₇	I ₁₅	I ₃₁	I ₃₅	I ₅₁	I ₃₉	I ₅₅	I ₄₃	I ₅₉	I ₄₇	I ₆₃	I ₆₇	I ₈₃	I ₇₁	I ₈₇	I ₇₅	I ₉₁	I ₇₉	I ₉₅								
Q_in(1)	Q ₀	Q ₁₆	Q ₄	Q ₂₀	Q ₈	Q ₂₄	Q ₁₂	Q ₂₈	Q ₃₂	Q ₄₈	Q ₃₆	Q ₅₂	Q ₄₀	Q ₅₆	Q ₄₄	Q ₆₀	Q ₆₄	Q ₈₀	Q ₆₈	Q ₈₄	Q ₇₂	Q ₈₈	Q ₇₆	Q ₉₂								
Q_in(2)	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁								
Q_in(3)	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂								
Q_in(4)	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃								

Figure 3: K=32, P=4, Input Order with no Input Buffer

sync_in	[Timing Diagram]																[Timing Diagram]															
enable_in	[Timing Diagram]																[Timing Diagram]															
I_in(1)	I ₀	I ₁	I ₂	I ₃	I ₄	I ₅	I ₆	I ₇	I ₈	I ₉	I ₁₀	I ₁₁	I ₁₂	I ₁₃	I ₁₄	I ₁₅	I ₁₆	I ₁₇	I ₁₈	I ₁₉	I ₂₀	I ₂₁	I ₂₂	I ₂₃								
Q_in(1)	Q ₀	Q ₁	Q ₂	Q ₃	Q ₄	Q ₅	Q ₆	Q ₇	Q ₈	Q ₉	Q ₁₀	Q ₁₁	Q ₁₂	Q ₁₃	Q ₁₄	Q ₁₅	Q ₁₆	Q ₁₇	Q ₁₈	Q ₁₉	Q ₂₀	Q ₂₁	Q ₂₂	Q ₂₃								

Figure 4: K=8, P=1, Input Order with Input Buffer

sync_in	[Timing Diagram]																[Timing Diagram]															
enable_in	[Timing Diagram]																[Timing Diagram]															
I_in(1)	I ₀	I ₄	I ₈	I ₁₂	I ₁₆	I ₂₀	I ₂₄	I ₂₈	I ₃₂	I ₃₆	I ₄₀	I ₄₄	I ₄₈	I ₅₂	I ₅₆	I ₆₀	I ₆₄	I ₆₈	I ₇₂	I ₇₆	I ₈₀	I ₈₄	I ₈₈	I ₉₂								
I_in(2)	I ₁	I ₅	I ₉	I ₁₃	I ₁₇	I ₂₁	I ₂₅	I ₂₉	I ₃₃	I ₃₇	I ₄₁	I ₄₅	I ₄₉	I ₅₃	I ₅₇	I ₆₁	I ₆₅	I ₆₉	I ₇₃	I ₇₇	I ₈₁	I ₈₅	I ₈₉	I ₉₃								
I_in(3)	I ₂	I ₆	I ₁₀	I ₁₄	I ₁₈	I ₂₂	I ₂₆	I ₃₀	I ₃₄	I ₃₈	I ₄₂	I ₄₆	I ₅₀	I ₅₄	I ₅₈	I ₆₂	I ₆₆	I ₇₀	I ₇₄	I ₇₈	I ₈₂	I ₈₆	I ₉₀	I ₉₄								
I_in(4)	I ₃	I ₇	I ₁₁	I ₁₅	I ₁₉	I ₂₃	I ₂₇	I ₃₁	I ₃₅	I ₃₉	I ₄₃	I ₄₇	I ₅₁	I ₅₅	I ₅₉	I ₆₃	I ₆₇	I ₇₁	I ₇₅	I ₇₉	I ₈₃	I ₈₇	I ₉₁	I ₉₅								
Q_in(1)	Q ₀	Q ₄	Q ₈	Q ₁₂	Q ₁₆	Q ₂₀	Q ₂₄	Q ₂₈	Q ₃₂	Q ₃₆	Q ₄₀	Q ₄₄	Q ₄₈	Q ₅₂	Q ₅₆	Q ₆₀	Q ₆₄	Q ₆₈	Q ₇₂	Q ₇₆	Q ₈₀	Q ₈₄	Q ₈₈	Q ₉₂								
Q_in(2)	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁	Q ₁								
Q_in(3)	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂	Q ₂								
Q_in(4)	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃	Q ₃								

Figure 5: K=32, P=4, Input Order with Input Buffer

Bit-reversal

The natural order of frequency domain samples that are output by the core is sliced bit-reversed order as shown in Figure 6 and Figure 7. This bit-reversed order is acceptable for some applications, but often the sliced ascending frequency order shown in Figure 8 and Figure 9 is more attractive, in which case the core must be ordered with a bit-reverser.

The waveforms shown in Figure 6 to Figure 9 relate directly to the timings of the input sequences of Figure 2 to Figure 5.

sync_out	[Timing Diagram]																[Timing Diagram]															
enable_out	[Timing Diagram]																[Timing Diagram]															
I_out(1)	I ₀	I ₄	I ₂	I ₆	I ₁	I ₅	I ₃	I ₇	I ₈	I ₁₂	I ₁₀	I ₁₄	I ₉	I ₁₃	I ₁₁	I ₁₅	I ₁₆	I ₂₀	I ₁₈	I ₂₂	I ₁₇	I ₂₁	I ₁₉	I ₂₃								
Q_out(1)	Q ₀	Q ₄	Q ₂	Q ₆	Q ₁	Q ₅	Q ₃	Q ₇	Q ₈	Q ₁₂	Q ₁₀	Q ₁₄	Q ₉	Q ₁₃	Q ₁₁	Q ₁₅	Q ₁₆	Q ₂₀	Q ₁₈	Q ₂₂	Q ₁₇	Q ₂₁	Q ₁₉	Q ₂₃								

Figure 6: K=8, P=1, Output Order with no Bit-reverser



'K' is the FFT length, 'P' is the complex parallelism

'input_buffer_present', when the core has the Input Buffer function included assign a value of 1 to include it's latency else assign a value of 0.

'bit_reverser_present' when the core has the Bit Reverser function included assign a value of 1 to include it's latency else assign a value of 0.

The HyperSpeed FFT core is a continuous pipeline, so valid data is continuously available at the output after the initial delay due to the pipeline filling up.

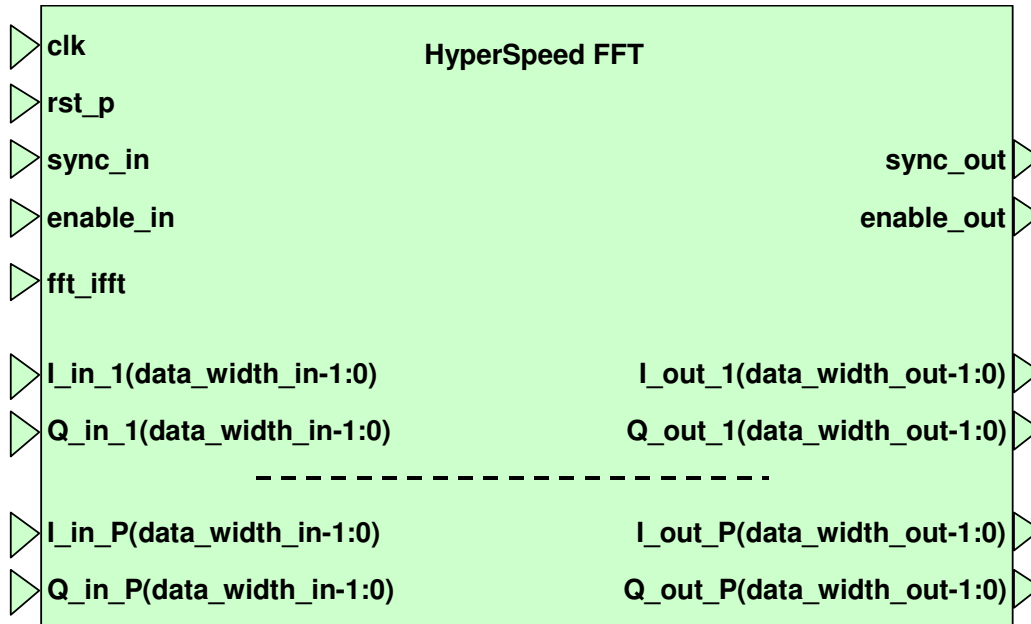
**CORE INTERFACE DESCRIPTION**

All core inputs should be synchronised to the 'clk' signal. All core outputs are synchronised by the 'clk' signal.

Signal	Direction	Type	Width	Function
clk	IN	Std logic	1 bit	The core clock rate is equal to f_s/P . Where f_s is the overall complex input sample rate and P is the complex Parallelism.
rst_p	IN	Std logic	1 bit	An active-high pulse, of duration greater than 2 core clock periods, resets the control logic, but not the data pipeline.
sync_in	IN	Std logic	1 bit	Active-high pulse marking the first sample of a new input frame. Precedes first samples of complex input data by one clock period.
enable_in	IN	Std logic	1 bit	Active-high signal asserted for a duration equal to the FFT frame length (K/P clock periods). Asserted coincident with valid input samples of complex input data.
fft_ifft	IN	Std logic	1 bit	Active-high signal to select FFT function, else an IFFT function is performed.
I_in_x, Q_in_x	IN	Std logic vector	(data_width_in - 1 downto 0)	2's complement time-domain data, ordered as described in previous text. The number parallel I and Q lines is equal to P.
sync_out	OUT	Std logic	1 bit	Active-high pulse marking the first transformed sample of a new output frame. Asserted one clock period before the first transformed sample(s) of each frame.
enable_out	OUT	Std logic	1 bit	Active-high signal asserted for a duration equal to the FFT frame length (K/P clock periods). Asserted coincident with complex output data.
I_out_x, Q_out_x	OUT	Std logic vector	(data_width_out - 1 downto 0)	2's complement frequency-domain data, ordered as described in previous text. The number parallel I and Q lines is equal to P.

Table 1: HyperSpeed FFT Interface Specification

N.B. A reduction in core power can be obtained by inhibiting the 'sync_in' signal, which will allow the pipeline to flush any frames of data that are being processed then stop most of the internal signals from toggling. This power-save mode can be used whenever the core is not required to operate in continuous mode.


Figure 10: PFFT Symbol

VERIFICATION

Verification of the core is achieved in several distinct phases in the design cycle.

Initially, a bit-true Matlab model of the core is used to provide reference output data, using a specific set of input stimuli. These models exactly match the behaviour of the hardware architecture and their outputs are bit-true representations of the actual core.

The input stimuli are used to functionally (pre-synthesis) test the VHDL code using the ModelSim simulator. The results of these simulations are automatically compared with the reference data generated by the bit-true Matlab model and pass/fail indications are reported.

After the VHDL has been synthesised and place-and-route of the core is complete, timing simulations, based on the chip vendor's simulation netlist and gate delay files for the core are performed. These also take the form of the functional tests, but being based on the compiled netlist and timing files from the FPGA vendor, they verify the expected timings of the delivered core.

The bit-true Matlab models are also included.



DELIVERABLES

Supplied Item	Description
Design	Macro netlist
Constraints File	Vendor specific
Instantiation Template	VHDL
Verification	VHDL test bench including ModelSim script and test data files. Compiled RTL VHDL Model. Bit-true Matlab model and scripts. Implementation reports.

Table 2: Items provided with each core

Electronic transfer is used to deliver the cores and supporting documentation.

Optional design support services are available to help you incorporate the core into your design.

ENGINEERING SUPPORT SERVICES AND MODELS

Accuracy and precision are system design considerations that affect data and twiddle bit-widths. RFEL can offer system-engineering advice to aid the selection of the optimal core configuration.

RFEL also offer system-engineering advice to assist in the selection of an optimal core for the system requirement. For example accuracy, precision and spurious free dynamic range/noise floor can all be modelled using Matlab bit-true models. This ensures input and output bit-widths, twiddle bit-width and arithmetic bit growth at each stage can all be optimised, ensuring the required system performance is obtained with the minimum of FPGA resource.

These models also demonstrate the excellent characteristics obtained using RFEL's fixed-point cores, without the need for resource-inefficient floating-point architectures.



CASE STUDIES

FFT length (K)	Complex Parallelism (P)	Input bit-width	Output bit-width	Twiddle bit-width	Input Buffer	Bit Reverser	Complex Throughput ¹	Logic Slices	DSP48 Slices	Block RAMs
1024	1	12	19	16	Y	Y	400MS/s	1659	20	23
1024	2	12	20	16	Y	Y	800MS/s	2946	38	23
1024	4	12	19	16	Y	Y	1.6GS/s	5672	76	35
1024	8	10	18	16	Y	Y	3.2GS/s	9260	150	56
1024	16	10	16	16	Y	Y	6.4GS/s	16233	294	92
4096	1	12	19	16	Y	Y	400MS/s	2232	24	37
4096	2	12	20	16	Y	Y	800MS/s	3952	46	37
4096	4	12	19	16	Y	Y	1.6GS/s	7116	92	46
4096	8	10	20	16	Y	Y	3.2GS/s	12146	182	66
4096	16	10	18	16	Y	Y	6.4GS/s	21703	358	113
8192	1	12	19	16	Y	Y	400MS/s	2565	26	55
8192	2	12	20	16	Y	Y	800MS/s	4448	50	55
8192	4	12	19	16	Y	Y	1.6GS/s	7938	100	64
8192	8	10	20	16	Y	Y	3.2GS/s	13620	198	77
8192	16	10	18	16	Y	Y	6.4GS/s	24706	390	122
32768	1	12	19	16	Y	Y	400MS/s	3871	30	158
32768	2	12	20	16	Y	Y	800MS/s	6080	58	158
32768	4	12	19	16	Y	Y	1.6GS/s	10222	116	167
32768	8	10	20	16	Y	Y	3.2GS/s	17208	230	165
65536	1	12	19	16	Y	Y	400MS/s	5324	32	290
65536	2	12	20	16	Y	Y	800MS/s	7696	62	290
65536	4	12	19	16	Y	Y	1.6GS/s	12164	124	299
65536	8	10	20	16	Y	Y	3.2GS/s	19802	246	280
131072	1	12	19	16	Y	N	400MS/s	7981	34	365
131072	2	12	20	16	Y	N	800MS/s	10440	66	365
131072	4	12	19	16	Y	N	1.6GS/s	15082	132	374
131072	8	10	20	16	Y	N	3.2GS/s	23068	262	345

Table 3: HyperSpeed FFT Virtex-4 Case Studies

¹ This figure assumes an FPGA clock rate of 400MHz, which is the maximum realistic rate for a medium speed grade Virtex-4 device.