# Datasheet: Floating-Point FFT

## Features

- ***FFT size***: User chosen power-of-two or non-power-of-two
- ***Programmability***: Easy to change functionality to meet application requirements.
- ***Dynamic Range***: Real and imaginary outputs are IEEE754 single precision
- ***Scalability***: array-based architecture means arbitrarily higher throughputs are obtained by increasing array size
- ***Power***: array interconnects are entirely local, reducing parasitic routing capacitance to keep power dissipation low and clock speeds high
- ***Implementation FPGA***: Can be used in any FPGA fabric containing embedded multipliers and memories.
- ***Data I/O*:** Streaming, normal order I/O with fixed-point 2's complement input words
- ***Inverse FFT:*** Run time selectable input

## I/O

- Can be either fixed-point or IEEE754 floating point
- Output format IEEE754

## Algorithm

The transform computation is based on a new formulation[1] of the discreet Fourier transform (DFT), different than any other FFT implementation, which decomposes it into structured sets of small, matrix-based DFTs. In particular the locality, simplicity and regularity of the processing core keeps interconnect delays lower than cell delays, leading to clock speeds that can approach the FPGA fabric limitations, e.g., "worst case" Fmax speeds determined by embedded elements. Short critical-path lengths with most of the delay in cells rather than interconnects also lowers power dissipation. Additionally, a novel "base-4" algorithm reduces the number of cycles needed per FFT to less than the transform size value *N.*

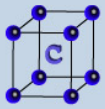
## Architecture

The architecture consists of two small arrays of pipelined, fine-grained, locally connected, simple processing elements, each containing a complex adder, a few registers and multiplexors. By cycling data through this array in a programmable way any size transform can be supported as well as desired variations from the standard DFT calculation.

## Precision

Unlike traditional pipelined FFTs, all additions are performed at full precision so that the round-off errors occur only in the twiddle multiplication steps. Consequently, the resulting precision is high. In the table below are accuracy measurements for 256/1024-pt streaming floating-point single-precision FFTs calculated based on 500 blocks of random 24-bit real and imaginary input

[1] J. Greg Nash, "Computationally Efficient Systolic Array for Computing the Discreet Fourier Transform, IEEE Trans. Signal Processing, Vol. 53, No.12, December 2005, pp.4640-4641.



data (random phase) obtained from Matlab simulations (Intel's model is generated by IP v17). The comparison reference is a double precision Matlab calculation.

The "mean absolute error" numbers are obtained by subtracting each reference output from each circuit output, taking the magnitude of this and then dividing by the magnitude of the reference value for that output point. The "maximum absolute error" is the largest of these errors computed over all 500 blocks of input data. (In terms of the signal-to-noise-quantization-ratio both Centar designs in the table below have values >150db.)

| | Intel (256pt) | Centar (256pt) | Intel (1024pt) | Centar (1024pt) |
|---|---|---|---|---|
| **Mean Absolute Error** | 2.4e-07 | 3.1e-08 | 2.9e-07 | 4.2e-08 |
| **Std Deviation Absolute Error** | 2.5e-07 | 3.1e-08 | 3.4e-07 | 7.7e-08 |
| **Maximum Absolute Error** | 2.4e-05 | 4.3e-06 | 9.1e-05 | 2.7e-05 |

## Performance and Resources

Here example performance and resource usage data is provided on a 256/1024-point, streaming FFT. The circuits were compiled using Intel's software tools (Quartus II v17) using a Stratix IV EP4SE360H29C2 FPGA (40nm technology) device. The TimeQuest static timing analyzer was used to determine maximum clock frequencies (Fmax) at 85C (worst case settings). The same Quartus settings were used for both the Centar and Intel designs (IP v17).

| | Intel | Centar v1 | Centar v2 | Intel | Centar |
|---|---|---|---|---|---|
| **Transform Size** | 256 points | | | 1024 points | |
| **ALMs** | 10834 | 7137 | 7834 | 13559 | 7186 |
| **ALUTs** | 16519 | 11050 | 12006 | 21801 | 11193 |
| **Registers** | 15545 | 10431 | 12535 | 18169 | 10495 |
| **M9Ks** | 54 | 62 | 30 | 87 | 62 |
| **Multipliers (18-bits)** | 48 | 129 | 129 | 64 | 129 |
| **Fmax (data rate, MHz)** | 299 | 456 | 426 | 285 | 386 |

(The larger word lengths in the SA lead to critical paths in the much larger multipliers, which limits Fmax compared to the smaller fixed-point versions.)

In the table above the adaptive logic module (ALM) is the basic unit of a Stratix IV FPGA (one 8-input LUT, two registers plus other logic). Comparison with Xilinx Virtex 6 devices (also 40nm) can be made by noting that two M9K memories are equivalent to a Xilinx BRAM and that an ALM is equivalent to between 1.2 and 1.8 LEs. These numbers come from benchmark studies which show 1 ALM=1.2 LEs (Xilinx white paper WP284 v1.0, December 19, 2007) and 1 ALM=1.8 LEs (Altera white paper), respectively. These papers actually compare Stratix III and Virtex 5 FPGAs; however, the Stratix IV/Virtex 6 architectures are essentially the same so the comparisons should still be valid.

## Device Family Support

Intel Device Families Supported

- Stratix
- Aria
- Cyclone
- Hardcopy

## Xilinx Device Families Supported

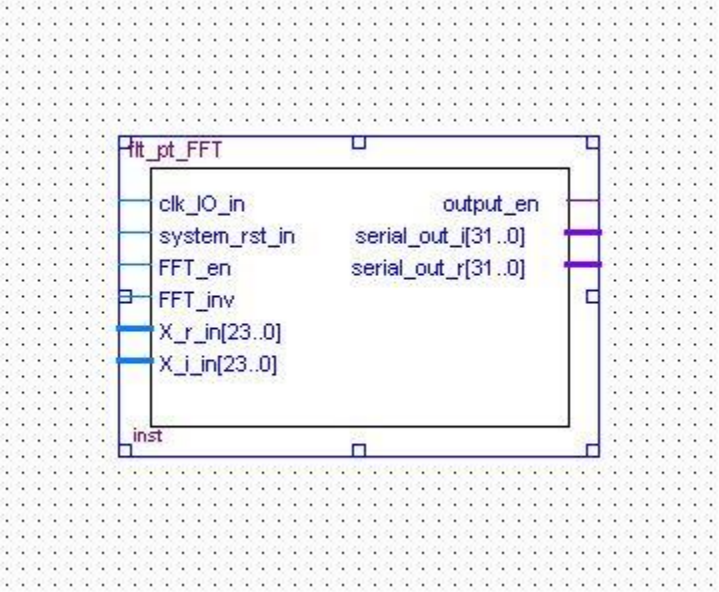
- Virtex 4-7
- Spartan
- Artix-7

## Deliverables

- Netlist (e.g., for Intel FPGAs a *.qxp file for synthesis or a *.vo or lib file for simulation)
- Synthesis constraints (e.g., for Intel FPGA's an *.sdc file)
- Modelsim Testbench (*.vo file for DFT circuit plus verilog testbench for control). Matlab verification utilities also available.
- Intel Stratix III FPGA board development kit testbench
- Matlab behavioral bit-accurate model (p-code)
- Documentation

## Pin-outs

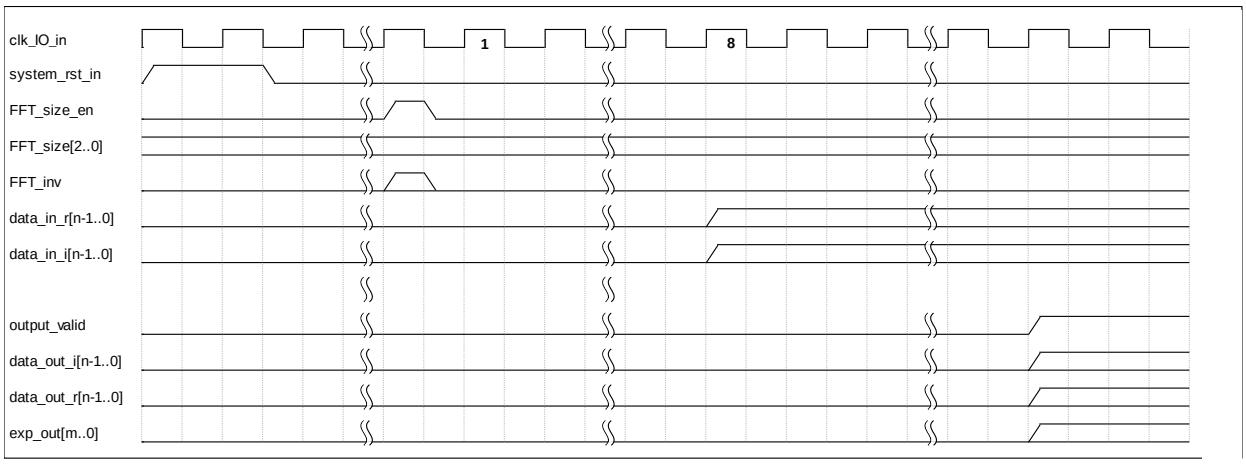
A symbol list corresponding to the pin-outs shown below are provided in the accompanying table. Depending upon the desired interfaces, some signals could change.



| Name | Signal | Description |
|---|---|---|
| clk_IO_in | clock | Can be either a low frequency board oscillator clock output in which case circuit clock is derived from a PLL or actual data I/O clock |
| system_rst_in | control | Resets circuit; active high |
| FFT_en | control | Registers FFT_inv signal; active high |
| FFT_inv | control | High->forward; low->inverse |
| data_in_r/i | 24-bit signed | Real and imaginary inputs (can be IEEE754) |
| output_en | control | High during data output |
| data_out_r/i | 32-bit signed | Real and imaginary outputs IEEE754 |

## Timing Diagram

A variety of timing possibilities exist depending upon the desired interface. That shown here is applicable to a streaming, normal order input/output scheme.




At any time after system_rst_in goes low, FFT_ en is used to latch the direction of the transform (FFT_inv=0/1 for forward/inverse). Following this, the circuit expects to see continuous data appearing on the 8th subsequent cycle. Valid data out occurs when output_valid (output_en) goes high. For streaming operation output is continuous from this point on.