High-Performance 16-Point Complex FFT/IFFT V1.0.5

July 5 2000 Product Specification

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
Phone: +1 408-559-7778
FAX: +1 408-559-7114
Email: coregen@xilinx.com
URL: http://www.xilinx.com/ipcenter

1 Features

- Drop-in module for Virtex™ and Virtex™-E FPGAs
- High-performance 16-point complex FFT and inverse FFT (IFFT)
- 16-bit complex input and output data
- 2's complement arithmetic
- Parallel architecture provides a new output sample on every clock
- Input data can be continuously streamed into the core
- Naturally ordered input and output data
- High performance and density guaranteed through Relational Placed Macro (RPM) mapping and placement technology
- Incorporates Xilinx Smart-IP technology for maximum performance
- To be used with version 2.1i or later of the Xilinx CORE Generator System

2 General Description

The vFFT16 fast Fourier transform (FFT) Core employs a radix-4 Cooley-Tukey [1] algorithm to compute the discrete Fourier transform (DFT), or inverse DFT, of a complex data vector. The input data is a vector of 16 complex values represented as 16-bit 2's complement numbers – 16-bits for each of the real and imaginary component of an input data sample.

3 Theory of Operation

The discrete Fourier transform (DFT) $X(k), k = 0, \ldots, N-1$ of a sequence $x(n), n = 0, \ldots, N-1$ is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-jnk2p/N} \quad k = 0, \ldots, N-1 \qquad (1)$$

where N is the transform size and $j = \sqrt{-1}$. The *fast Fourier transform (FFT)* is a computationally efficient algorithm for computing a DFT.

The Xilinx 16-point transform engine employs a Cooley-Tukey radix-4 decimation-in-frequency (DIF) FFT [1] to compute the DFT of a complex sequence. In general, this algorithm requires the calculation of columns or *ranks* of radix-4 butterflies. These radix-4 butterflies are sometimes referred to as *dragonflies.* Each processing rank consists of $N/4$ dragonflies. For N=16 there are 2 dragonfly ranks, with each rank comprising 4 dragonflies.

The FFT processor input-data for the Core is a vector of 16 complex samples. The real and imaginary components of each sample are represented as 16-bit 2's complement numbers. The data input and output buffers are stored internally within the FPGA. The phase factors used in the FFT calculation are generated within the Core. Like the input-data, the phase factors are kept to a precision of 16 bits. The complex output samples are also defined with 16 bits of precision for each of the real and imaginary components.

4 Finite Word Length Considerations

The radix-4 FFT algorithm processes an array of data by successive passes over the array. On each pass, the algorithm performs dragonflies, each dragonfly picking up four complex numbers and returning four complex numbers to the same addresses but in a different memory bank. The numbers returned to memory by the processor are larger than the numbers picked from memory. A strategy must be employed to accommodate this dynamic range expansion. A full explanation of scaling strategies and their implications is beyond the scope of this document, the reader is referred to several documents available in the open literature [2] [3] that discuss this topic.

The Xilinx 16-point FFT Core scales dragonfly results by a factor of 4 (or 2 bits) on each processing pass. The *SCALE_MODE* pin can be used to force an additional scaling by one bit on the first processing pass only. The scaling results in the final output sequence being modified by the factor $1/16$ when *SCALE_MODE=0* and 1/32 when *SCALE_MODE*=1. Formally, the output sequence $X'(k), k = 0,\ldots,N-1$ computed by the Core (when FWD_INV=1) is defined by Eq. (3)

$$X'(k) = \frac{1}{sN} X(k) = \frac{1}{sN} \sum_{n=0}^{N-1} x(n) e^{-jnk2p/N} \quad k = 0,\ldots,N-1 \tag{3}$$

where s=1 when *SCALE_MODE*=0 and s=2 when *SCALE_MODE*=1. The *SCALE_MODE* pin can be used for both the forward and inverse FFT modes of operation.

The vfft16 core also computes the IFFT according to the following defining equation

$$x(n) = \frac{1}{sN} \sum_{k=0}^{N-1} X(k) e^{j2pnk/N} \quad n = 0,1,\ldots,N-1 \tag{2}$$

The in-built scaling in the core accounts for the 1/N scale factor in front of the summation in Eq. (2). When *SCALE_MODE*=1, an additional scaling by a factor of ½ will be scheduled in the core. The additional scaling by 1 bit is inserted during the memory write operation of the first of the 2 processing phases.

5 Pinout

The Core symbol is shown in Figure 1.

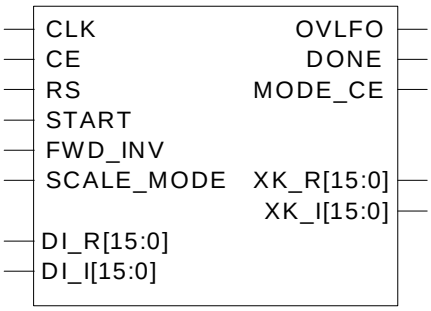

Figure 1: 16-point FFT symbol.

Table 1 defines the pin functionality.

Signal	Direction	Description	Signal	Direction	Description
CLK	Input	Clock input (active rising edge)	START	Input	FFT start control (active high)
RS	Input	Master Reset (active high)			
CE	Input	Clock enable (active high)	DONE	Output	Transform complete strobe. This signal is present for one clock cycle at the beginning of an result vector. (active high)
SCALE_MODE	Input	FFT scaling control. When SCALE_MODE=0 the FFT output vector is scaled by 1/16. When SCALE_MODE=1 the FFT output vector is scaled by 1/32.	FWD_INV	Input	Defines if a forward (FWD_INV=1) or inverse (FWD_INV=0) is performed
MODE_CE	Output	Indicates when the FWD_INV and SCALE_MODE pins are sampled (active high)	OVFLO	Output	Active high Arithmetic overflow indicator. Even when employing a 2-bit scale factor for each FFT processing phase, certain input signals can cause arithmetic overflow. This pin indicates that an internal arithmetic overflow has been generated. When additional scaling is employed by setting SCALE_MODE=1, there is no possibility of overflow occurring and this signal will not be active. OVFLO is removed when the core is reset by asserting RS, when START is asserted, or at the beginning of the next output result vector as indicated by DONE.
DI_R[15:0]	Input	Input databus – real component	XK_R[15:0]	Output	DFT result – real component
DI_I[15:0]	Input	Input databus – imaginary component	XK_I[15:0]	Output	DFT result – imaginary component

Table 1: 16-point FFT Core pin definitions.

6 Description

The 16-point FFT Core accepts naturally ordered data on the input buses *DI_R* and *DI_I* and performs a complex FFT or IFFT. These buses are respectively the real and imaginary components of the input sequence. Data must be supplied in a continuous stream to the Core. An internal input data memory controller orders the data into blocks to be presented to the FFT processor. Each block is 16 samples in length. When a new block is assembled it is immediately passed on to the FFT engine. The calculation of a complete FFT requires 16 clock cycles. This means that transforms are performed in a block-continuous fashion. The low processing latency allows data to be continuously streamed into the Core. The *START* signal is used to initiate the first transform. Once *START* has been asserted to begin the first transform there is no need to assert it again – the Core will perform transforms continuously once it has been started. There is an initial 82 clock cycle latency from the time *START* is asserted to the availability of the first valid output sample. This is illustrated in Figure 2.

Transforms can be computed back-to-back. The *MODE_CE* pin indicates when the operating mode signals *FWD_INV* and *SCALE_MODE* are sampled by the core. This signal is useful when alternating forward and inverse FFTs are performed.

The user can elect to assert *START* at any time to re-synchronize the processor to the input data. However, this causes the internal pipeline to be flushed. The result is that each time *START* is applied, the *82* clock cycle startup latency is experienced.

Just as data is continuously streamed into the Core, DFT samples are also continuously streamed out of the Core on the *XK_R* and *XK_I* buses. These buses respectively provide the real and imaginary components of the complex output samples. The *DONE* signal identifies the start of a transform output vector. Figure 2 shows the timing relationship between *DONE* and the *XK_R* and *XK_I* buses. The DFT samples appear in natural order on the output buses starting with XK_[R/I](0) as indicated in the figure.

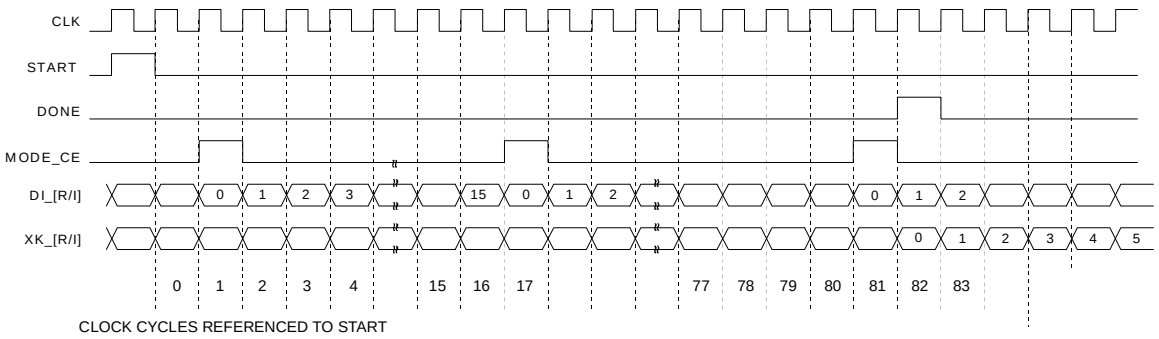

Figure 2: 16-point FFT core timing diagram.

7 Performance

The complete calculation of 1 16-point FFT requires 16 clock cycles. The transform execution time is

$$T_{\mathrm{FFT}} = \frac{16}{f_{\mathrm{CLK}}}$$

where f_{CLK} is the system clock frequency. For example, for a system clock frequency of 100 MHz, the execution time is 160 ns. When the clock frequency is increased to 120 MHz the transform time is 133.33 ns.

8 Clock-Enable Issues

There are several issues involving the clock-enable *CE* pin that designers should be familiar with when developing systems with this core. *CE* is a high fan-out signal and should be presented to the Core via a low-skew clock buffer to achieve maximum operating frequency. Refer to the Xilinx device data book for more information on these features. **The *CE* pin is a master clock enable for the entire Core**. When in the inactive state (CE=0), all core operations are stalled until *CE* is re-asserted (CE=1).

9 Core Resource Utilization

The 16-point FFT Core occupies 1386 logic slices. The geometry of the RPM requires it to be placed in a XCV300 or larger device.

10 Behavioral Simulation

Release Version 1.0 of the vfft16 core has VHDL behavioral model but does not include a verilog behavioral model.

11 Implementation

The vfft16 core is supplied as a group of edif netlists. The top level netlist is called *vfft16.edn*. All of the netlists that are delivered with the core must be present in the user's project directory. The edif netlist files are:

```
vfft16.edn
xdsp_cnt2.edn
xdsp_cnt4.edn
xdsp_cnt5.edn
xdsp_coss16.edn
xdsp_mul16x17.edn
xdsp_mux2w16.edn
xdsp_mux2w16r.edn
xdsp_mux2w4.edn
xdsp_mux4w16r.edn
xdsp_radd16.edn
xdsp_radd16c.edn
xdsp_radd17.edn
xdsp_ramd16a4.edn
xdsp_reg16.edn
xdsp_reg16b.edn
xdsp_reg4.edn
```



```
xdsp_rsub16.edn
xdsp_rsub16b.edn
xdsp_rsub16c.edn
xdsp_rsub17b.edn
xdsp_sinn16.edn
xdsp_tcompw16.edn
xdsp_tcompw16b.edn
xdsp_tcompw17.edn
xdsp_triginv.edn
```

12 References

[1] J. W. Cooley and J. W. Tukey, ``An Algorithm for the Machine Calculation of Complex Fourier Series'',
Math. Comput., Vol. 10, pp. 297-301, April 1965.

[2] W. R. Knight and R. Kaiser, ``A Simple Fixed-Point Error Bound for the Fast Fourier Transform'', *IEEE Trans. Acoustics, Speech and Signal Proc.,* Vol. 27, No. 6, pp. 615-620, Dec. 1979.

[3] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing,* Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1975.

